Forgery Attacks on ++AE Authenticated Encryption Mode

Hassan Qahur Al Mahri
hassan.mahri@hdr.qut.edu.au

Leonie Simpson
lr.simpson@qut.edu.au

Harry Bartlett
h.bartlett@qut.edu.au

Ed Dawson
e.dawson@qut.edu.au

Kenneth Koon-Ho Wong
kk.wong@qut.edu.au

Queensland University of Technology
2 George St, Brisbane 4000, Australia

ABSTRACT
In this paper, we analyse a block cipher mode of operation submitted in 2014 to the cryptographic competition for authenticated encryption (CAESAR). This mode is designed by Recacha and called ++AE (plus-plus-ae). We propose a chosen plaintext forgery attack on ++AE that requires only a single chosen message query to allow an attacker to construct multiple forged messages. Our attack is deterministic and guaranteed to pass ++AE integrity check. We demonstrate the forgery attack using 128-bit AES as the underlying block cipher. Hence, ++AE is insecure as an authenticated encryption mode of operation.

CCS Concepts
• Security and privacy~Cryptanalysis and other attacks

Keywords
Authenticated encryption; ++AE; confidentiality; integrity; block cipher; forgery attack; symmetric encryption; CAESAR; AEAD.

1. INTRODUCTION
Authenticated encryption (AE) is a scheme that provides both data confidentiality and integrity [1]. Confidentiality ensures that only the intended recipient can read the contents of a transmitted message, while data integrity assures that the contents have not been altered during transmission by unauthorised means [4]. Some security protocols do not require all data to be encrypted, such as protocol headers that need to be in clear text in order to be identified by different communicating parties. Therefore, Rogaway [9] singles out a new version of AE schemes, called authenticated encryption with associated data (AEAD), where the associated data are authenticated only.

One common approach to provide AE and AEAD is through block cipher modes. One of these proposed modes is ++AE (plus-plus-ae) [8]; a candidate submitted to the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) [2].

++AE is an authenticated encryption with associated data (AEAD) mode of operation for a block cipher of arbitrary block length and key size. Recacha designed ++AE [8] based on his previously proposed AE schemes, namely IOBC [6] and IOC [7]. All Recacha's schemes, IOBC, IOC and ++AE, use a block cipher in ECB mode with cross block chaining and a detectable redundancy paradigm for integrity check rather than computing a message authentication code (MAC). Compared to other similar modes, ++AE has some sophisticated features, such as it is parallelisable, has minimal overhead and supports the authentication of associated data [8].

In Recacha's earlier schemes [6], the communicating parties agree on a predefined value, called Integrity Check Vector (ICV). This is appended to the end of the plaintext message before encryption (see Figure 1). The encrypted ICV block is referred to as a Modification Detection Code (MDC). During decryption, errors in ciphertext are propagated to the last block (MDC). The altered MDC yields the decryption of an incorrect ICV. Only messages with the expected ICV are accepted as authentic. As will be discussed later, ++AE uses a different method for determining the value of the ICV, although the verification process follows the same concept described above.

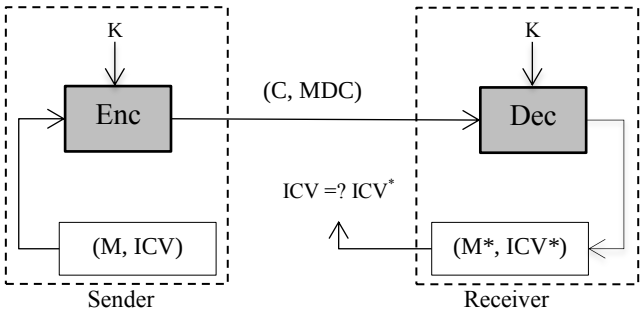

Figure 1. Integrity mechanism [6]

Weaknesses in previous designs using an ICV have been identified. Mitchell [5] showed that IOBC is vulnerable to a known plaintext forgery attack with a complexity of around $2^{n/3}$, where n is the underlying block cipher block length. Bottinelli et al. [3] breach the integrity of IOC with a probability of $1 - 3 \times 2^{-n}$.

This paper identifies a weakness in the integrity assurance mechanism of the ++AE mode. This weakness can be exploited in a chosen plaintext forgery attack similar to the attack approach by Bottinelli, Reyhanitabar and Vaudenay [3] in breaking the IOC authenticated encryption mode. For ++AE, if an attacker can obtain the ciphertext message corresponding to a chosen plaintext message containing two groups of four consecutive blocks anywhere before the last block, the attacker can construct multiple

ACE '16 Canberra, ACT Australia

http://dx.doi.org/10.1145/2843043.2843355

forged messages in various ways. The forged ciphertext messages are constructed either by deleting the ciphertext blocks corresponding to one group and repeating the second group of ciphertext blocks, or swapping the ciphertext blocks of the two groups. The forged messages are guaranteed to decrypt to plaintexts that have an ICV identical to the ICV of the original plaintext message. Therefore, we break the integrity of ++AE as an AE scheme.

This paper is arranged as follows: Section 2 briefly describes the ++AE scheme. In Section 3 we outline a weakness in the ++AE structure related to the chaining mechanism. We also provide a mathematical proof of this. Section 4 proposes attack procedures using the weakness in the ++AE chaining mechanism. We demonstrate the forgery attacks using 128-bit AES as the underlying block cipher in Section 5. The last section draws a conclusion.

2. DESCRIPTION OF ++AE

This section describes the native encryption and decryption processes of ++AE as an AE mode of operation. Our description is based on Recacha's paper [8] submitted to the CAESAR competition. Recacha submitted two versions: v1.0 in March 2014 and the revised version v1.1 in April 2014. The encryption and decryption baseline procedures for ++AE mode are identical in both versions.

++AE uses a k-bit key K for encryption, decryption and authentication operations and vector initialisation. Let $\varepsilon_K(.)$ denote the block cipher encryption function under the key K, and $\mathrm{d_K}(.)$ denote the decryption function under the same key. Let the block length of the block cipher be b bits. Let $P = \{P_1, P_2, \dots, P_N\}$ be a plaintext message that consists of N blocks of length b bits. Suppose $C = \{C_1, C_2, \dots, C_N\}$ is the ciphertext obtained from encrypting the message P. Addition $(+)$ and subtraction $(-)$ are regular arithmetic operations modulo 2^b, while bitwise XOR operation is denoted as $\oplus$.

Each message P gets a secret and random value S. S is used to generate a pair of secret and random initial vectors denoted by IV_a and IV_b each of length b bits, such that $IV_a = \varepsilon_K(S)$ and $IV_b = \varepsilon_K(IV_a)$. For each plaintext message, the integrity check vector (ICV) appended to the message is calculated as follows:

$$ICV = (IV_a \oplus S) + \big(IV_b \oplus (N + M)\big)$$

where M denotes the number of blocks of associated data. Defining the ICV in this way provides some protection against forgery attacks in which the message length is changed, such as only deleting or inserting blocks. The ++AE encryption and decryption operations are described in Algorithm 1 and Algorithm 2 and illustrated in Figure 1 and Figure 2.

Algorithm 1: ++AE Encryption

1: $\mathrm{Encrypt}_K(S, P)$
2: $Q_0 \leftarrow IV_a$
3: $I_0 \leftarrow IV_b$
4: **for** $i \leftarrow 1$ **to** N **do**
5: $\quad I_i \leftarrow P_i \oplus Q_{i-1}$
6: $\quad Q_i \leftarrow I_i + I_{i-1} + Q_{i-1}$
7: $\quad X_i \leftarrow Q_i \oplus I_{i-1}$
8: $\quad C_i \leftarrow \varepsilon_K(X_i)$
9: **end for**
10: $MDC \leftarrow \varepsilon_K(((ICV \oplus Q_N) + I_N + Q_N) \oplus I_N)$
11: **return** $C||MDC = \{C_1, \dots, C_N\}||MDC$

Algorithm 2: ++AE Decryption

1: $\mathrm{Decrypt}_K(S, C||MDC)$
2: $Q_0 \leftarrow IV_a$
3: $I_0 \leftarrow IV_b$
4: **for** $i \leftarrow 1$ **to** N **do**
5: $\quad X_i \leftarrow d_K(C_i)$
6: $\quad Q_i \leftarrow X_i \oplus I_{i-1}$
7: $\quad I_i \leftarrow Q_i - (I_{i-1} + Q_{i-1})$
8: $\quad P_i \leftarrow I_i \oplus Q_{i-1}$
9: **end for**
10: $ICV' \leftarrow ((d_K(MDC) \oplus I_N) - (I_N + Q_N)) \oplus Q_N$
11: **if** $ICV' = ICV$ **then**
12: $\quad$ **return** $P = \{P_1, \dots, P_N\}$
13: **else**
14: $\quad$ **return** INVALID
15: **end if**

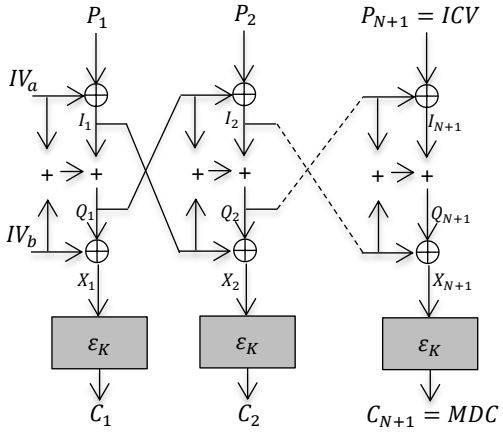

Figure 2. ++AE encryption [8]

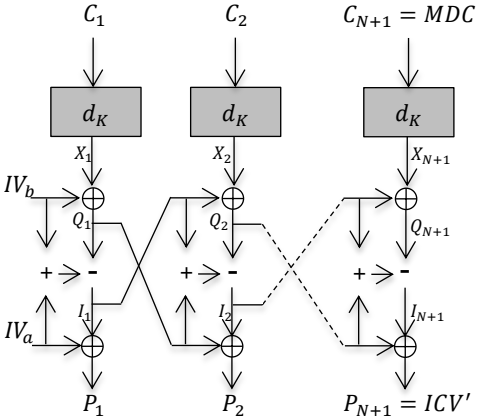

Figure 3. ++AE decryption [8]

3. FLAW IN ++AE STRUCTURE

In this section, we first make some observations on the basic operations of ++AE scheme. Then, we apply the observations to point out a weakness in the ++AE structure.

Note that the four summation operations and cross block chaining mechanism in ++AE occur prior to the block cipher encryption function in the encryption operation, or after the block cipher decryption function in the decryption operation. Therefore, the effect of the chaining propagates independently of the block cipher functions. We look at special cases of particular plaintext blocks that permit repetition of the internal chaining values, Q and I.

3.1 Plaintext blocks of value $2^b - 1$

In this case, we consider plaintext blocks represented by the integer $(2^b - 1)$ or $(-1 \pmod{2^b})$. We first start with the following lemma before explaining the flaw in ++AE mode.

Lemma 1: Let (a) be a b-bit binary number. Then, $(2^b - 1) \oplus a \equiv 2^b - 1 - a \pmod{2^b}$.

Proof:

Let $a = \sum_{j=0}^{b-1} a_j . 2^j$ be the binary representation of (a).

Then, $(2^b - 1) \oplus a = (2^b - 1) \oplus \sum_{j=0}^{b-1} a_j . 2^j$

$= \sum_{j=0}^{b-1} 2^j \oplus \sum_{j=0}^{b-1} a_j . 2^j$

$= \sum_{j=0}^{b-1} (1 \oplus a_j) 2^j$

$\equiv \sum_{j=0}^{b-1} (1 - a_j) 2^j \pmod{2^b}$

$\equiv (\sum_{j=0}^{b-1} 2^j - \sum_{j=0}^{b-1} a_j . 2^j) \pmod{2^b}$

$\equiv 2^b - 1 - a \pmod{2^b}$ □

Theorem 1: Suppose $P = P_1, P_2, .., P_{N+1}$ (where $N \geq 4$) is a plaintext message submitted for ++AE encryption. If $P_i = P_{i+1} = P_{i+2} = P_{i+3} = 2^b - 1$ $(i.e. -1 \ (mod\ 2^b))$ (where $1 \leq i \leq N - 3$), then the inner vectors I_{i+3} and Q_{i+3} will be equal to I_{i-1} and Q_{i-1} respectively.

Proof:

From Lemma 1, we have:

$I_j = P_j \oplus Q_{j-1}$
$= (2^b - 1) \oplus Q_{j-1}$
$= 2^b - 1 - Q_{j-1}$
, for $j = i, i+1, i+2, i+3$.

$Q_j = I_j + I_{j-1} + Q_{j-1}$
$= 2^b - 1 + I_{j-1}$
, for $j = i, i+1, i+2, i+3$.

Therefore:

$I_i = 2^b - 1 - Q_{i-1}$
$Q_i = 2^b - 1 + I_{i-1}$

$I_{i+1} = 2^b - 1 - Q_i = -I_{i-1}$
$Q_{i+1} = 2^b - 1 + I_i = 2(2^b - 1) - Q_{i-1}$

$I_{i+2} = 2^b - 1 - Q_{i+1} = Q_{i-1} - (2^b - 1)$
$Q_{i+2} = 2^b - 1 + I_{i+1} = 2^b - 1 - I_{i-1}$

$I_{i+3} = 2^b - 1 - Q_{i+2} = I_{i-1}$
$Q_{i+3} = 2^b - 1 + I_{i+2} = Q_{i-1}$ □

Theorem 1 proves that the value of the inner vectors I_{i+3} and Q_{i+3}, after processing four consecutive plaintext blocks each of value $(2^b - 1)$, will be the same as the values of I_{i-1} and Q_{i-1}. The repetition of the inner vectors is a weakness that can be exploited for forgery attacks. Note that this flaw is independent of the encryption key, the secret value S, the block cipher used and the block length.

3.2 Plaintext blocks of value $2^{b-1} - 1$

We next consider the case when the plaintext blocks are equal to the integer $(2^{b-1} - 1)$.

Lemma 2: Let (a) be a b-bit binary number. Then, $(2^{b-1} - 1) \oplus a \equiv 2^{b-1} - 1 - a \pmod{2^b}$.

Proof:

Let $a = \sum_{j=0}^{b-1} a_j . 2^j$ be the binary representation of (a).

Since $2^{b-1} \equiv -2^{b-1} \pmod{2^b}$, then:

$(2^{b-1} - 1) \oplus a = (2^{b-1} - 1) \oplus \sum_{j=0}^{b-1} a_j . 2^j$

$= \sum_{j=0}^{b-2} 2^j \oplus \sum_{j=0}^{b-1} a_j . 2^j$

$= (\sum_{j=0}^{b-2} 2^j \oplus \sum_{j=0}^{b-2} a_j . 2^j) + (0 \oplus a_{b-1} . 2^{b-1})$

$= (\sum_{j=0}^{b-2} (1 \oplus a_j) 2^j) + (a_{b-1} . 2^{b-1})$

$\equiv (\sum_{j=0}^{b-2} (1 - a_j) 2^j + a_{b-1} . 2^{b-1}) \pmod{2^b}$

$\equiv (\sum_{j=0}^{b-2} 2^j - \sum_{j=0}^{b-2} a_j . 2^j - a_{b-1} . 2^{b-1}) \pmod{2^b}$

$\equiv 2^{b-1} - 1 - a \pmod{2^b}$ □

Theorem 2: Suppose $P = P_1, P_2, .., P_{N+1}$ (where $N \geq 4$) is a plaintext message submitted for ++AE encryption. If $P_i = P_{i+1} = P_{i+2} = P_{i+3} = 2^{b-1} - 1$ (where $1 \leq i \leq N - 3$), then the inner vectors I_{i+3} and Q_{i+3} will be equal to I_{i-1} and Q_{i-1} respectively.

Proof:

The proof of Theorem 2 directly follows the proof of Theorem 1, except that Lemma 1 is replaced by Lemma 2 and 2^b is replaced by 2^{b-1}.

3.3 Other plaintext blocks

It is straightforward to show that the approach taken in Theorems 1 and 2 can also be applied to the values: $2^b - 2$ and $2^{b-1} - 2$ for any block size $b \geq 2$.

Therefore, $2^b - 1$, $2^{b-1} - 1$, $2^b - 2$ and $2^{b-1} - 2$ are the list of values from which we can choose a value to form a group of four identical consecutive plaintext blocks, so that the inner vectors I and Q before and after processing the group of blocks are the same.

3.4 Mixing plaintext blocks

Instead of choosing four identical consecutive plaintext blocks using one of the values listed in section 3.3, we can alternate the values of the plaintext blocks.

Theorem 3: Suppose $P = P_1, P_2, .., P_{N+1}$ (where $N \geq 4$) is a plaintext message submitted for ++AE encryption. If

$P_i, P_{i+1}, P_{i+2}, P_{i+3} = (2^b - 1, 2^{b-1} - 1, 2^b - 1, 2^{b-1} - 1),$
$(2^{b-1} - 1, 2^b - 1, 2^{b-1} - 1, 2^b - 1),$
$(2^b - 1, 2^b - 1, 2^{b-1} - 1, 2^{b-1} - 1),$
$(2^{b-1} - 1, 2^{b-1} - 1, 2^b - 1, 2^b - 1),$

$(2^b - 1, 2^{b-1} - 1, 2^{b-1} - 1, 2^b - 1)$,
or $(2^{b-1} - 1, 2^b - 1, 2^b - 1, 2^{b-1} - 1)$

(where $1 \le i \le N - 3$), then the inner vectors I_{i+3} and Q_{i+3} will be equal to I_{i-1} and Q_{i-1} respectively.

Proof:

From Lemma 1&2:

For $P_j = 2^b - 1$:

$I_j = 2^b - 1 - Q_{j-1}$, for $j = i, i+1, i+2, i+3$.
$Q_j = 2^b - 1 + I_{j-1}$, for $j = i, i+1, i+2, i+3$.

For $P_j = 2^{b-1} - 1$:

$I_j = 2^{b-1} - 1 - Q_{j-1}$, for $j = i, i+1, i+2, i+3$.
$Q_j = 2^{b-1} - 1 + I_{j-1}$, for $j = i, i+1, i+2, i+3$.

If $P_i, P_{i+1}, P_{i+2}, P_{i+3} = (2^b - 1, 2^{b-1} - 1, 2^b - 1, 2^{b-1} - 1)$:

$$I_i = 2^b - 1 - Q_{i-1}$$
$$Q_i = 2^b - 1 + I_{i-1}$$
$$I_{i+1} = 2^{b-1} - 1 - Q_i = -2^{b-1} - I_{i-1}$$
$$Q_{i+1} = 2^{b-1} - 1 + I_i = 3(2^{b-1}) - 2 - Q_{i-1}$$
$$I_{i+2} = 2^b - 1 - Q_{i+1} = Q_{i-1} - (2^{b-1} - 1)$$
$$Q_{i+2} = 2^b - 1 + I_{i+1} = 2^{b-1} - 1 - I_{i-1}$$
$$I_{i+3} = 2^{b-1} - 1 - Q_{i+2} = I_{i-1}$$
$$Q_{i+3} = 2^{b-1} - 1 + I_{i+2} = Q_{i-1}$$

For other cases, the essential difference from the previous one is that we change the position of 2^b and 2^{b-1}. Therefore, the proofs of these cases exactly follow the above proof. □

Note that if we change $2^b - 1$ and $2^{b-1} - 1$ to $2^b - 2$ and $2^{b-1} - 2$, the above approach still gives the same results. Combining the cases presented in this section with those listed in section 3.3, there are 16 possible combinations of four consecutive plaintext blocks that result in the inner vectors I and Q being repeated, for any block size $b \ge 2$.

4. PROPOSED FORGERY ATTACKS

The forgery attacks we propose only require a single chosen plaintext message to be encrypted using ++AE. The attacker can use the corresponding ciphertext and modify it in various ways without invalidating the MDC or its decryption to the original ICV provided the length of the ciphertext is unchanged. Possible modifications include inserting and deleting certain ciphertext blocks, or reordering them. The attack is deterministic and the modified ciphertext is guaranteed to pass the ++AE integrity test during decryption. These forgeries are applicable to both versions of ++AE.

4.1 Forgery attack using insertion and deletion

For this proposed attack, suppose a plaintext message P contains two separate groups of four identical consecutive blocks whose values are chosen from the list in section 3.3. The value chosen for one group may be different from the second group value. We denote these two values as r_1 and r_2. The message is chosen as follows:

$$P = P_1, P_2, .., P_{N+1} \ (N \ge 9)$$

where

$$P_i = P_{i+1} = P_{i+2} = P_{i+3} = r_1 \ (1 \le i \le N - 8)$$

and

$$P_j = P_{j+1} = P_{j+2} = P_{j+3} = r_2 \ \ (i + 5 \le j \le N - 3)$$

as shown in Figure 4. This message is encrypted using ++AE to produce a ciphertext message $C = C_1, C_2, .., C_{N+1}$, such that $C_i, C_{i+1}, C_{i+2}, C_{i+3}$ correspond to $P_i, P_{i+1}, P_{i+2}, P_{i+3}$ and $C_j, C_{j+1}, C_{j+2}, C_{j+3}$ correspond to $P_j, P_{j+1}, P_{j+2}, P_{j+3}$.

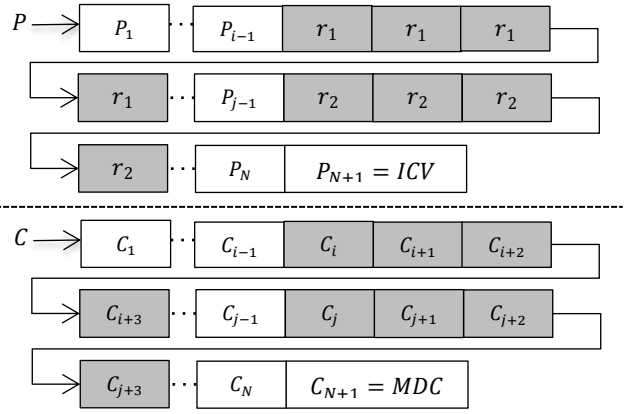

Figure 4. The chosen message

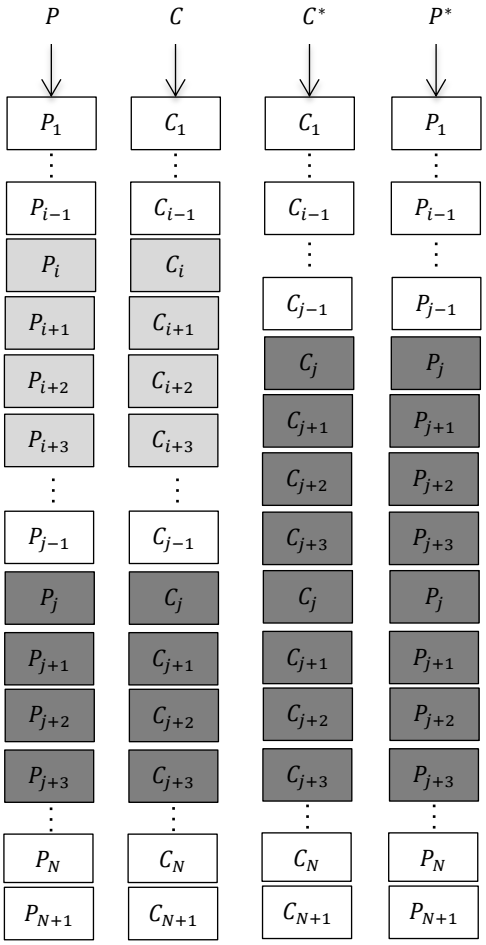

Figure 5. Forgery attack using insertion and deletion

One way in which we can construct a forged ciphertext message C^* from the obtained ciphertext C, is by deleting the four consecutive ciphertext blocks $(C_i, C_{i+1}, C_{i+2}, C_{i+3})$, and repeating

the four consecutive ciphertext blocks $(C_j, C_{j+1}, C_{j+2}, C_{j+3})$. The forged ciphertext message is established as follows:

$$C_k^* = C_k \ (1 \le k \le i-1),$$
$$C_k^* = C_{k+4} \ (i \le k \le j-1),$$
$$C_k^* = C_k \ (j \le k \le N+1).$$

Such a forged message C*, as illustrated in Figure 5, will not cause any change in the number of blocks N. It will also produce the same MDC and decrypt to give the same ICV, so it will be accepted as genuine by the receiver.

Another way to construct a forged ciphertext message from C is by inserting the $(C_i, C_{i+1}, C_{i+2}, C_{i+3})$ blocks after the original ones and deleting the $(C_j, C_{j+1}, C_{j+2}, C_{j+3})$ blocks. That is, the forged message will be as follows:

$$C_k^* = C_k \ (1 \le k \le i+3),$$
$$C_k^* = C_{k-4} \ (i+4 \le k \le j+3),$$
$$C_k^* = C_k \ (j+4 \le k \le N+1).$$

Note that this approach can be used for any of the four plaintext-block values mentioned in Section 3.3. Thus, there are multiple chosen plaintext messages that can be used to obtain ciphertext messages, each of which can be manipulated in various ways to obtain multiple forgeries.

4.2 Forgery attack using swapping

Instead of insertion and deletion using two separate groups of four identical consecutive plaintext messages, we can also use two consecutive groups where each group of four identical consecutive plaintext blocks uses a different value chosen from the list of values mentioned in Section 3.3. One way in which the attacker can construct a forged ciphertext message is by swapping the ciphertext blocks of the two groups. These modifications do not cause any change in the number of blocks N, or in the internal chaining values, Q and I. Hence, The MDC and its decryption to the original ICV will remain unchanged.

To illustrate the proposed attacks, suppose a plaintext message P contains two consecutive groups of four identical consecutive plaintext blocks whose values are chosen from the list in section 3.3. In the first group the plaintext blocks all have the value r_1 and in the second group the plaintext blocks all have the value r_2 provided that r_1 is different form r_2. These two groups can be anywhere in the plaintext before the last block:

$$P = P_1, P_2, .., P_{N+1} \ (N \ge 8)$$

where

$$P_i = P_{i+1} = P_{i+2} = P_{i+3} = r_1 \ (1 \le i \le N-7)$$

and

$$P_{i+4} = P_{i+5} = P_{i+6} = P_{i+7} = r_2 \ (1 \le i \le N-7).$$

This message is encrypted using ++AE to produce a ciphertext message $C = C_1, C_2, .., C_{N+1}$ where $C_i, C_{i+1}, C_{i+2}, C_{i+3}$ correspond to $P_i, P_{i+1}, P_{i+2}, P_{i+3}$ and $C_{i+4}, C_{i+5}, C_{i+6}, C_{i+7}$ correspond to $P_{i+4}, P_{i+5}, P_{i+6}, P_{i+7}$.

We can construct a forged ciphertext message C^* from the obtained ciphertext C, by swapping the four consecutive ciphertext blocks $(C_i, C_{i+1}, C_{i+2}, C_{i+3})$ with the ciphertext blocks $(C_{i+4}, C_{i+5}, C_{i+6}, C_{i+7})$. The forged ciphertext message is established as follows:

$$C_k^* = C_k \ (1 \le k \le i-1),$$
$$C_k^* = C_{k+4} \ (i \le k \le i+3),$$
$$C_k^* = C_{k-4} \ (i+4 \le k \le i+7),$$
$$C_k^* = C_k \ (i+8 \le k \le N+1).$$

Such a forged message, C*, as illustrated in Figure 6, will produce the same MDC and decrypt to give the same ICV, so it will be accepted as genuine by the receiver.

Instead of swapping, we can construct other forged messages from C by replacing the ciphertext blocks of one group with the ciphertext blocks of the second group. If the ciphertext blocks of the first group are replaced by the ciphertext blocks of the second group, the forged message is constructed as follows:

$$C_k^* = C_k \ (1 \le k \le i-1),$$
$$C_k^* = C_{k+4} \ (i \le k \le i+3),$$
$$C_k^* = C_k \ (i+4 \le k \le N+1).$$

However, if the ciphertext blocks of the second group are replaced by the ciphertext blocks of the first group, the forged message is established as follows:

$$C_k^* = C_k \ (1 \le k \le i+3),$$
$$C_k^* = C_{k-4} \ (i+4 \le k \le i+7),$$
$$C_k^* = C_k \ (i+8 \le k \le N+1).$$

Note that the same procedure is also valid for any two different values mentioned in Section 3.3. Hence, more plaintext messages can be chosen to obtain ciphertext messages, each of which can be manipulated in various ways to obtain multiple forgeries.

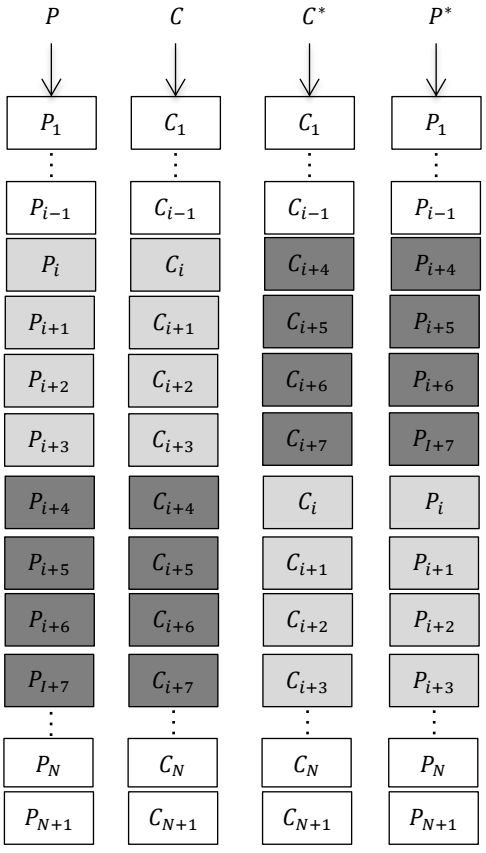

Figure 6. Forgery attack using swapping

4.3 Forgery attack with mixed value blocks

For the forgeries described in sections 4.1 and 4.2, we assume that the four consecutive plaintext blocks in each group are all of the same value chosen from the list in Section 3.3. Note that using Theorem 3 these four blocks can have mixed values and still do not change the internal chaining values, Q and I. Thus, the

forgeries presented in sections 4.1 and 4.2 are also valid for the twelve groups of four consecutive mixed-value plaintext blocks listed in Section 3.4.

5. EXPERIMENTAL VERIFICATION

In this section, we verify the proposed forgery attacks on ++AE experimentally. The underlying block cipher used in this experiment is AES with a 128-bit key. The experiment is implemented using C language and the implementation is run on a desktop computer with 64-bit Windows 7 Enterprise operating system and Intel Core i7-4790 3.6GHz processor.

5.1 Performing a forgery attack using insertion and deletion

The experiment is performed for a plaintext message of eleven 128-bit blocks as follows:

1. An attacker chooses a plaintext message containing two separate groups such that each group contains four consecutive identical blocks.
2. The eleven-block plaintext is encrypted using ++AE mode with 128-bit AES as the underlying block cipher.
3. The attacker can construct a forged message by deleting the ciphertext blocks correspond to one group and repeating the ciphertext blocks correspond to the second group, as shown in Figure 5.
4. The modified ciphertext is decrypted using ++AE mode with 128-bit AES as the underlying block cipher.

We demonstrated the above forgery attack with a specific example, as follows:

1. Form a plaintext of eleven 128-bits blocks considering the last block as the ICV, as shown in Table 1. The message has the blocks $(P_2, P_3, P_4, P_5) = (2^{128} - 1, 2^{128} - 1, 2^{128} - 1, 2^{128} - 1)$ as one group and the blocks $(P_7, P_8, P_9, P_{10}) = (2^{127} - 2, 2^{127} - 2, 2^{127} - 2, 2^{127} - 2)$ as the second group. The two groups are shown in **bold**. The remaining plaintext blocks have been chosen randomly.

Table 1. The chosen plaintext message

6bc1bee22e409f96e93d7e117393172a
ffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
30c81c46a35ce411e5fbc1191a0a52ea
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
f69f2445df4f9b17ad2b417be66c3710

2. Table 2 shows the ciphertext obtained from the encryption of the message in Table 1. The corresponding ciphertext blocks (C_2, C_3, C_4, C_5) and (C_7, C_8, C_9, C_{10}) are shown in **bold**.

Table 2. The obtained ciphertext message

b9501472308964e206f3652aa72ccdd6
8b6af01acb7464cb68c4a3548aaf95a6
469c7fcb75d5d9a1b418cb997b09a185
8b6af01acb7464cb68c4a3548aaf95a6
469c7fcb75d5d9a1b418cb997b09a185
c213b40c852738f6d446d79eb8f79b03
4106dc419a06db149dd22bfb583ffeee
f6c71eedc3d99bb183cb5b8d1568e606
4106dc419a06db149dd22bfb583ffeee
f6c71eedc3d99bb183cb5b8d1568e606
3758893b632fd6857c89f9b5aeb1427d

3. We modify the ciphertext by deleting the ciphertext blocks (C_2, C_3, C_4, C_5) and repeating the ciphertext blocks (C_7, C_8, C_9, C_{10}), as shown in Table 3.

Table 3. The forged ciphertext message

b9501472308964e206f3652aa72ccdd6
c213b40c852738f6d446d79eb8f79b03
4106dc419a06db149dd22bfb583ffeee
f6c71eedc3d99bb183cb5b8d1568e606
4106dc419a06db149dd22bfb583ffeee
f6c71eedc3d99bb183cb5b8d1568e606
4106dc419a06db149dd22bfb583ffeee
f6c71eedc3d99bb183cb5b8d1568e606
4106dc419a06db149dd22bfb583ffeee
f6c71eedc3d99bb183cb5b8d1568e606
3758893b632fd6857c89f9b5aeb1427d

4. We decrypt the modified ciphertext to obtain the message shown in Table 4. Note that the obtained plaintext message is different to the original chosen plaintext shown in Table 1, but both have the same last block.

Table 4. The decrypted forged message

6bc1bee22e409f96e93d7e117393172a
30c81c46a35ce411e5fbc1191a0a52ea
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
f69f2445df4f9b17ad2b417be66c3710

5.2 Performing a forgery attack using swapping

The experiment is performed for a plaintext message of ten 128-bit blocks as follows:

1. An attacker chooses two consecutive groups of four consecutive identical blocks. The value chosen for one group should be different from the second group value
2. The ten-block plaintext is encrypted using ++AE mode with 128-bit AES as the underlying block cipher.
3. The attacker can construct a forged message by swapping the ciphertext blocks of the two groups, as shown in Figure 6.
4. The modified ciphertext is decrypted using ++AE mode with 128-bit AES as the underlying block cipher.

We demonstrated the above forgery attack with a specific example as follows:

1. Form a plaintext of ten 128-bits blocks considering the last block as the ICV, as shown in Table 5. The message has the blocks $(P_2, P_3, P_4, P_5) = (2^{128}-1, 2^{128}-1, 2^{128}-1, 2^{128}-1)$ as one group and the blocks $(P_6, P_7, P_8, P_9) = (2^{127}-2, 2^{127}-2, 2^{127}-2, 2^{127}-2)$ as the second group. The two groups are shown in **bold**. The remaining plaintext blocks have been chosen randomly.

Table 5. The chosen plaintext message

6bc1bee22e409f96e93d7e117393172a
ffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
f69f2445df4f9b17ad2b417be66c3710

2. Table 6 shows the ciphertext obtained from the encryption of the message in Table 5. The corresponding ciphertext blocks (C_2, C_3, C_4, C_5) and (C_6, C_7, C_8, C_9) are shown in **bold**.

Table 6. The obtained ciphertext message

b9501472308964e206f3652aa72ccdd6
8b6af01acb7464cb68c4a3548aaf95a6
469c7fcb75d5d9a1b418cb997b09a185
8b6af01acb7464cb68c4a3548aaf95a6
469c7fcb75d5d9a1b418cb997b09a185
f6c71eedc3d99bb183cb5b8d1568e606
4106dc419a06db149dd22bfb583ffeee
f6c71eedc3d99bb183cb5b8d1568e606
4106dc419a06db149dd22bfb583ffeee
e0cd0b77eed4f7041fde256038dbf51f

3. We modify the ciphertext by swapping the ciphertext blocks (C_2, C_3, C_4, C_5) and the ciphertext blocks (C_6, C_7, C_8, C_9) , as shown in Table 7.

Table 7. The forged ciphertext message

b9501472308964e206f3652aa72ccdd6
f6c71eedc3d99bb183cb5b8d1568e606
4106dc419a06db149dd22bfb583ffeee
f6c71eedc3d99bb183cb5b8d1568e606
4106dc419a06db149dd22bfb583ffeee
8b6af01acb7464cb68c4a3548aaf95a6
469c7fcb75d5d9a1b418cb997b09a185
8b6af01acb7464cb68c4a3548aaf95a6
469c7fcb75d5d9a1b418cb997b09a185
e0cd0b77eed4f7041fde256038dbf51f

4. We decrypt the modified ciphertext to obtain the message shown in Table 8. Note that the obtained plaintext message is different to the original chosen plaintext shown in Table 5, but both have the same last block.

Table 8. The decrypted forged message

6bc1bee22e409f96e93d7e117393172a
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffffe
ffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
f69f2445df4f9b17ad2b417be66c3710

5.3 Performing a forgery attack with mixed value blocks

In this experiment we perform a forgery attack using insertion and deletion for a plaintext message of eleven 128-bit blocks exactly as described in Section 5.1 except that the plaintext blocks have mixed values.

We demonstrated this forgery attack with a specific example as follows:

1. Form a plaintext of eleven 128-bits blocks considering the last block as the ICV, as shown in Table 9. The message has the blocks $(P_2, P_3, P_4, P_5) = (2^{128}-1, 2^{127}-1, 2^{128}-1, 2^{127}-1)$ as one group and the blocks $(P_7, P_8, P_9, P_{10}) = (2^{128}-2, 2^{127}-2, 2^{128}-2, 2^{127}-2)$ as the second group. The two groups are shown in **bold**. The remaining plaintext blocks have been chosen randomly.

Table 9. The chosen plaintext message

6bc1bee22e409f96e93d7e117393172a

ffffffffffffffffffffffffffffffff
7fffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
7fffffffffffffffffffffffffffffff
30c81c46a35ce411e5fbc1191a0a52ea
ffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffe
ffffffffffffffffffffffffffffffe
7fffffffffffffffffffffffffffffe
f69f2445df4f9b17ad2b417be66c3710

2. Table 10 shows the ciphertext obtained from the encryption of the message in Table 9. The corresponding ciphertext blocks (C_2, C_3, C_4, C_5) and (C_7, C_8, C_9, C_{10}) are shown in **bold**.

Table 10. The obtained ciphertext message

b9501472308964e206f3652aa72ccdd6
8b6af01acb7464cb68c4a3548aaf95a6
e2c75fad134f621d9223df0a9c2793d8
8b6af01acb7464cb68c4a3548aaf95a6
e2c75fad134f621d9223df0a9c2793d8
c213b40c852738f6d446d79eb8f79b03
973f2ef34879e2027f1734303ff21f89
f6c71eedc3d99bb183cb5b8d1568e606
973f2ef34879e2027f1734303ff21f89
f6c71eedc3d99bb183cb5b8d1568e606
3758893b632fd6857c89f9b5aeb1427d

3. We modify the ciphertext by deleting the ciphertext blocks (C_7, C_8, C_9, C_{10}) and repeating the ciphertext blocks (C_2, C_3, C_4, C_5), as shown in Table 11.

Table 11. The forged ciphertext message

b9501472308964e206f3652aa72ccdd6
8b6af01acb7464cb68c4a3548aaf95a6
e2c75fad134f621d9223df0a9c2793d8
8b6af01acb7464cb68c4a3548aaf95a6
e2c75fad134f621d9223df0a9c2793d8
8b6af01acb7464cb68c4a3548aaf95a6
e2c75fad134f621d9223df0a9c2793d8
8b6af01acb7464cb68c4a3548aaf95a6
e2c75fad134f621d9223df0a9c2793d8
c213b40c852738f6d446d79eb8f79b03
3758893b632fd6857c89f9b5aeb1427d

4. We decrypt the modified ciphertext to obtain the message shown in Table 12. Note that the obtained plaintext message is different to the original chosen plaintext shown in Table 9, but both have the same last block.

Table 12. The decrypted forged message

6bc1bee22e409f96e93d7e117393172a
ffffffffffffffffffffffffffffffff
7fffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
7fffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
7fffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffff
7fffffffffffffffffffffffffffffff
30c81c46a35ce411e5fbc1191a0a52ea
f69f2445df4f9b17ad2b417be66c3710

6. CONCLUSION

The ++AE proposal has been submitted to the CAESAR competition for authenticated encryption. In this paper, we reviewed the internal structure of ++AE mode of operation. We found ++AE has a flaw that can be exploited in a chosen plaintext forgery attack. The flaw is due to the fact that for certain input block values, the encryption of four consecutive plaintext blocks will result in the values of the internal chaining values, Q and I being repeated. Such repetition occurs for several different sets of four consecutive plaintext blocks.

This flaw enables an attacker to choose a plaintext and use the corresponding ciphertext to construct multiple forged messages that will be accepted as authentic messages during decryption. An attacker can either delete and insert particular ciphertext blocks in the original cipher message or reorder them. The forged messages are guaranteed to pass ++AE integrity test regardless of the encryption key, the block cipher used and the block length. That is, the success rate for this attack is 100%.

Given a chosen plaintext scenario, we performed the proposed forgery attacks for a selection of chosen plaintext messages and for both the insertion and deletion approach and the swapping approach. The experiments were conducted using an implementation of ++AE with 128-bit AES as the underlying block cipher. In every case, the forgery resulted in the correct ICV, and so would not be detected as modified by a receiver.

Based on these results, we conclude that ++AE has a serious flaw in the integrity assurance mechanism. The mode should not be considered suitable for authenticated encryption.

7. REFERENCES

[1] Bellare, M. and Namprempre, C., 2008. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J CRYPTOL 21*, 4 (Oct. 2008), 469-491. DOI= http://dx.doi.org/10.1007/s00145-008-9026-x.

[2] Bernstein, D.J., 2014. Cryptographic competitions: CAESAR. http://competitions.cr.yp.to/caesar-call.html.

[3] Bottinelli, P., Reyhanitabar, R., and Vaudenay, S., 2014. Breaking the IOC authenticated encryption mode. In *Proceedings of the 7th International Conference on Cryptology* (Marrakesh, Morocco, May 28-30, 2014). AFRICACRYPT 2014. Springer International Publishing, Cham, ZG, 126-135. DOI= http://dx.doi.org/10.1007/978-3-319-06734-6_8.

[4] Menezes, A.J., Van Oorschot, P.C., and Vanstone, S.A., 1996. *Handbook of Applied Cryptography*. CRC press, Boca Raton, FL.

[5] Mitchell, C.J., 2013. Analysing the IOBC authenticated encryption mode. In *Proceedings of the 18th Australasian Conference on Information Security and Privacy* (Brisbane, Australia, July 1-3, 2013). ACISP 2013. Springer, Heidelberg, Berlin, 1-12. DOI= http://dx.doi.org/10.1007/978-3-642-39059-3_1.

[6] Recacha, F., 1996. IOBC: Un nuevo modo de encadenamiento para cifrado en bloque. In *Proceedings: IV Reunion Espanola de Criptologia, Valladolid*, 85-92.

[7] Recacha, F., 2013. IOC: The most lightweight authenticated encryption mode? In *Submission to National Institute of Standards and Technology*. http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html.

[8] Recacha, F., 2014. ++AE. http://competitions.cr.yp.to/caesar-submissions.html.

[9] Rogaway, P., 2002. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM conference on Computer and Communications Security* (Washington, DC, November 18 - 22, 2002). CCS '02. ACM, New York, NY, 98-107. DOI= http://dx.doi.org/10.1145/586110.586125.