A Forgery Attack on the Candidate LTE Integrity Algorithm 128-EIA3
(updated version)

Thomas Fuhr, Henri Gilbert, Jean-René Reinhard, and Marion Videau
ANSSI, France

Abstract

In this note we show that the message authentication code 128-EIA3 considered for adoption as a third integrity algorithm in the emerging mobile standard LTE is vulnerable to a simple existential forgery attack. This attack allows, given any message and the associated MAC value under an unknown integrity key and an initial vector, to predict the MAC value of a related message under the same key and the same initial vector with a success probability 1/2.

1 Introduction

A set of two cryptographic algorithms is currently considered for inclusion of the emerging mobile communications standard LTE of the 3rd Generation Partnership Project 3GPP. It consists of an encryption algorithm and an integrity algorithm named 128-EEA3 and 128-EIA3 – that are both derived from a core stream cipher named ZUC.[1] ZUC, 128-EEA3, and 128-EIA3 were designed by DACAS in China, and preliminary drafts of their specification [2, 3] are currently open for public evaluation. After its adoption by 3GPP, 128-EEA3/EIA3 will represent the third LTE encryption and integrity algorithm set, in addition to the already adopted sets 128-EEA1/EIA1 based on the stream cipher SNOW 3G and 128-EEA2/EIA2 based on AES.

The integrity algorithm 128-EIA3 is an IV-dependent MAC that on input (1) a 128-bit key, (2) various public parameters that together determine a 128-bit initial vector, and (3) an input message of length between 1 and 20000 bits produces a 32-bit MAC value. 128-EIA3 uses a universal hash function-based construction and has therefore many features in common with the algorithms of the well known Carter-Wegman family of message authentication codes [1].

[1] EEA stands for "EPS Encryption Algorithm" and EIA stands for "EPS Integrity Algorithm". EPS (Evolved Packet System) is an evolution of the third generation system UMTS that consists of new radio access system named LTE (Long Term Evolution) and a new core network named SAE (System Architecture Evolution).

In this note we show that 128-EIA3 is vulnerable to a simple existential forgery attack that allows, given any known message M, any known (or even unknown) initial vector, and the associated MAC under an unknown key, to predict with a success probability 1/2 the MAC value associated with a new message $M' \neq M$ derived from M, the same initial vector, and the same unknown key. This attack is generic, it does not rely on any specific feature of ZUC and works with any underlying stream cipher. It exploits a subtle deviation of 128-EIA3 from the requirements of the Wegmann-Carter paradigm: the mechanism used to generate masking values applied to the output of the universal hash function underlying 128-EIA3 does not match the model used in the proofs. While the latter requirements can be informally summarised in saying that mask values must behave as one-time masks, 128-EIA3 masks do not behave this way. As will be shown in the sequel, distinct 128-EIA3 mask values are not necessarily independent. Consequently, the arguments from [6] and [7] that are invoked in [4] to infer bounds on the success probability of forgery attacks from the properties of the universal function of 128-EIA3 are not applicable.

Outline of this note. In Section 2, we give a short description of the 128-EIA3 algorithm. We then describe the attack in Section 3. In Section 4 we discuss the reasons why the security proofs for related constructions by Krawczyk [6] and Shoup [7] do not guarantee the security of 128-EIA3 and we give some evidence that a simple modification of the specification of 128-EIA3 might suffice to thwart the attack.

Notation. Throughout this note we are using the following notation.

- $\mathcal{S}$ is a stream cipher
- $W^{(i)}$ denotes the i-th bit of a 32-bit word W
- For a 32-bit word $W = (W^{(0)}, \ldots, W^{(31)})$ and integers a and b between 1 and 31, $W \ll a$ and $W \gg b$ denote the $(32-a)$-bit word, resp. the $(32-b)$-bit word that results from a left shift of W by a positions, resp. the right shift of W by b positions.[2] More in detail $W \ll a = (W^{(a)}, \ldots, W^{(31)})$.

2 Description of the 128-EIA3 Integrity Algorithm

128-EIA3 makes a black box use of a stream cipher to generate a keystream from a key and an initial value. A stream cipher $\mathcal{S}$ is an algorithm that takes as input a k-bit key IK and an n-bit initialisation value IV and outputs a binary sequence $b_0, \ldots, b_i, \ldots$ named the keystream. The keystream is used to compute

[2] We are thus using the same somewhat unusual convention as a in [3] for defining the symbol "$\gg$" as a "shift and truncate" rather a mere shift. This is motivated by the fact that this convention is more convenient for presenting the attack of Section 2.

a 32-bit MAC value Tag according to the following procedure:

Algorithm 1 The 128-EIA3 MAC algorithm

Input: $IK \in \{0,1\}^k$, $IV \in \{0,1\}^n$, $\ell \in \mathbb{N}^*$
Input: $M = (m_0, \ldots, m_{\ell-1}) \in \{0,1\}^\ell$
  $(b_0, \ldots, b_{\ell+63}) \leftarrow \mathcal{S}(IK, IV)|_{\ell+63}$
  $Tag = 0$
  for $i = 0$ to $\ell - 1$ **do**
    $W_i \leftarrow (b_i, \ldots, b_{i+31})$
    if $m_i = 1$ **then**
      $Tag \leftarrow Tag \oplus W_i$
    end if
  end for
  $W_\ell \leftarrow (b_\ell, \ldots, b_{\ell+31})$
  $Tag \leftarrow Tag \oplus W_\ell$
  $W_{mask} \leftarrow (b_{\ell+32}, \ldots, b_{\ell+63})$
  $Tag \leftarrow Tag \oplus W_{mask}$
  return Tag

In other words the MAC value T associated with IK, IV, and an ℓ-bit message M is derived by accumulating (for a set of positions i determined by the message bits and the message length) 32-bit words $W_i = (b_i, \ldots, b_{i+31})$ extracted from the keystream by applying it a 32-bit "sliding window":

$$T = (\bigoplus_{i=0}^{\ell} m_i W_i) \oplus W_\ell \oplus W_{mask},$$

where $W_{mask} = W_{\ell+32}$. The parameter lengths used in 128-EIA3 are the following: $k = n = 128$; l is between 1 and 20000.

3 An Existential Forgery

128-EIA3 has some specific properties that we will now exploit to transform a valid MAC Tag for a message M into a valid MAC for a message M' related to M.

First, we can notice that the keystream words W_i corresponding to message bits m_i are not independent from each other. More precisely, we have $W_{i+1} = ((W_i \ll 1), b_{i+32})$.

Moreover the "one-time masks" W_{mask} associated with identical values of IV but different message lengths are also related. Let us suppose that W_{mask} is the one-time mask generated for the input (M, IV) and W'_{mask} is the one-time mask generated for the input (M', IV). If $length(M') - length(M) = \Delta\ell$ with

$0 < \Delta\ell < 32$, we have $W'_{mask} = (W_{mask} \ll \Delta\ell, \beta_0, \ldots, \beta_{\Delta\ell-1})$, for some bit values β_i.

We can now describe our forgery attack. Let us suppose that the adversary knows a valid MAC value T for a given message $M = (m_0, \ldots, m_{\ell-1})$ of length ℓ bits and a given IV value IV. This MAC can be transformed into a candidate value MAC T' for the $\ell + 1$-bit message $M' = (0, m_0, \ldots, m_{\ell-1})$, the same IV value IV, and the same key IK that is valid with probability 1/2 as detailed hereafter.

Let us analyse what happens during the computation of the MAC for M' and IV. The generated keystream $b_0, \ldots, b_{\ell+64}$ is the same as the keystream that was used to compute T, with one extra bit $b_{\ell+64}$. As a consequence, the words $W_i, 0 \leq i \leq \ell$ are identical. The one-time mask used is $W'_{mask} = (b_{\ell+33}, \ldots, b_{\ell+64}) = ((W_{mask} \ll 1), b_{\ell+64})$. T' is then given by the following formula :

$$\begin{aligned}
T' &= (\bigoplus_{i=0}^{\ell} m'_i W_i) \oplus W_{\ell+1} \oplus W'_{mask} \\
&= (\bigoplus_{i=0}^{\ell-1} m_i W_{i+1}) \oplus W_{\ell+1} \oplus W'_{mask} \\
&= (\bigoplus_{i=0}^{\ell-1} m_i((W_i \ll 1), b_{i+32})) \oplus (W_\ell \ll 1, b_{\ell+32}) \oplus ((W_{mask} \ll 1), b_{\ell+64}) \\
&= (((\bigoplus_{i=0}^{\ell-1} m_i W_i) \oplus W_\ell \oplus W_{mask}) \ll 1, \beta) \\
&= (T \ll 1, \beta)
\end{aligned}$$

$(T \ll 1, \beta)$ is thus a valid MAC for M' and IV. Knowing T, the adversary only needs to guess the value of bit β, which happens with probability 1/2.

This attack can then be generalized by recurrence to generate a valid MAC for $(0^r||M)$, with probability 2^{-r}, when $r < 32$: the corresponding tag is then $T_r = ((T \ll r), \beta_0, \ldots, \beta_{r-1})$ for some value of bits $(\beta_0, \ldots, \beta_{r-1})$.

Equivalently, we have that $T = (\alpha_0, \ldots, \alpha_{r-1}, T_r \gg r)$. This equation enables an adversary to transform a valid MAC IV, T_r for $(0^r||M)$ into a valid MAC for M with probability 2^{-r}.

The attack was checked for $r = 1$ and larger values of r on a few example values, using the implementation programs provided in annexes of the specification documents [2, 3].

4 Partial Flaw in 128-EIA3 Security Arguments

The Design and Evaluation Report [4] erroneously invokes the security proofs of [7] to infer that in the case of 128-EIA3, no forgery of a new message can succeed

with probability higher than 2^{-32} from the fact that the algorithm makes use of an ε almost xor universal (ε-AXU) family of hash functions with $\varepsilon = 2^{-32}$.

A family of hash functions $\{H_K\}_{K\in\{0,1\}^k}$ of range $\{0,1\}^t$ is ε-AXU if for any two distinct messages M, M' in $\{0,1\}^*$ and any $c \in \{0,1\}^t$

$$Pr_{K\in\{0,1\}^k}[H_K(M) \oplus H_K(M') = c] \leq \varepsilon.$$

In [4], a proof is given that for any value of IV, the family of hash functions used in 128-EIA3, i.e. the intermediate value obtained in the MAC computation associated with key K just before before the exclusive or with W_{mask} is ε-AXU with $\varepsilon = 2^{-32}$.

As far as we know, the first construction of a secure MAC using ε-AXU hash functions has been issued by Krawczyk [6], who proves that given $H_K(M) \oplus r$ for secret uniformly drawn values of K and r, an adversary cannot determine $H_K(M') \oplus r$ with probability higher than ε. What makes our attack work in the case of 128-EIA3 is that the one-time masks used for messages M and M' of distinct lengths are different but related. In [6] Krawczyk does not address the one-time mask generation issue.

In [7], Shoup gives a security proof for a MAC algorithm where $MAC(m) = (r, F(r) \oplus H_K(M))$, where F is a pseudo-random function. In the case of 128-EIA3, r can be viewed as the IV of the stream cipher $\mathcal{S}$ (up to the minor difference that IV is not a fully random value). It seems reasonable to assume that the keystream is a pseudo-random function of the IV. However, the mask computation also involves the message length and leads to distinct, but related mask values, for identical IVs and different message lengths. Therefore the proof does not apply.

A slight variant of 128-EIA3. Let us then consider the slightly modified MAC described in Algorithm 2, that is quite similar to 128-EIA3 and requires the same number of keystream bits and the same amount of computation – the single differences being that the mask value consists of the first keystream bits and the universal hash function output value is derived from the subsequent keystream bits.

The attack of Section 3 does no longer work against this "tweaked" version of 128-EIA3, and the main reason why Shoup's proof is not applicable also disappears. However, the ε-AXU hash function also depends on the IV, which is not the case in Shoup's proof - that would thus need being adapted to take this property into account. Moreover, this proof does not guarantee what happens when the number of verification requests exceeds 2^{32}, or when IVs can be reused - for instance because a message M and the associated 32-bit MAC T transmitted by a sending entity is observed and a trial substitution by an adversary of (M, T) by a distinct pair (M', T') happens to be accepted at the receiving side, following an attack scenario considered in [5]. Thus some unwanted properties might well still hold for the "tweaked" 128-EIA3. Therefore, while the above example suggests that minor modifications to 128-EIA3 might

Algorithm 2 A modified version of 128-EIA3

Input: $IK \in \{0,1\}^k$, $IV \in \{0,1\}^n$, $\ell \in \mathbb{N}^*$
Input: $M = (m_0, \ldots, m_{\ell-1}) \in \{0,1\}^\ell$
  $(b_0, \ldots, b_{\ell+63}) \leftarrow \mathcal{S}(IK, IV)|_{\ell+63}$
  $Tag = 0$
  $W_{mask} \leftarrow (b_0, \ldots, b_{31})$
  for $i = 0$ to $\ell - 1$ **do**
    $W_i \leftarrow (b_{i+32}, \ldots, b_{i+63})$
    if $m_i = 1$ **then**
      $Tag \leftarrow Tag \oplus W_i$
    end if
  end for
  $W_\ell \leftarrow (b_{\ell+32}, \ldots, b_{\ell+63})$
  $Tag \leftarrow Tag \oplus W_\ell$
  $Tag \leftarrow Tag \oplus W_{mask}$
  return Tag

suffice to thwart the attack presented above and remove features that deviate from the Carter-Wegman paradigm, we do not claim that it is fit for purpose in the context of LTE security - since any candidate EIA algorithm obviously has to be studied deeply in the context it is supposed to be used, not only by applying proofs that restrict the attack scenario.

5 Conclusion

The attack presented in this note shows that the algorithm 128-EIA3 specified in the preliminary draft [2] does not offer an adequate integrity protection. The identified weakness does neither relate to the core of 128-EIA3 nor to the underlying stream cipher ZUC, but only to the way the mask values are derived from the keystream.[3]

References

[1] J.L. Carter and M.N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Science*, 128:143–154, 1979.

[2] ETSI/SAGE. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 1: 128-EEA3 and 128-EIA3 Specification. Version 1.4, 30th July 2010. Technical report, ETSI, 2010. `http://www.gsmworld.com/documents/EEA3_EIA3_specification_v1_4.pdf`.

[3] An early version of this note was forwarded to the designers of 128-EIA3, who plan to introduce some modifications in the specification of the mask values to solve the issue.

[3] ETSI/SAGE. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification. Version 1.4, 30th July 2010. Technical report, ETSI, 2010. `http://www.gsmworld.com/documents/EEA3_EIA3_ZUC_v1_4.pdf`.

[4] ETSI/SAGE. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 4: Design and Evaluation Report. Version 1.1, 11th August 2010. Technical report, ETSI, 2010. `http://www.gsmworld.com/documents/EEA3_EIA3_Design_Evaluation_v1_1.pdf`.

[5] H. Handschuh and B. Preneel. Key Recovery Attacks On Universal Hash Function Based MAC Algorithms. In *Advances in Cryptology - CRYPTO 2008*, pages 144–161. Springer, 2008.

[6] Hugo Krawczyk. LFSR-based Hashing and Authentication. In *Advances in Cryptology - CRYPTO' 94*, pages 129–139. Springer, 1994.

[7] Victor Shoup. On Fast and Provably Secure Message Authentication Based on Universal Hashing. In *Advances in Cryptology - CRYPTO' 96*, pages 313–328. Springer, 1996.