Message Extension Attack against Authenticated Encryptions: Application to PANDA

Sasaki, Yu; Wang, Lei

2016

Sasaki, Y., & Wang, L. (2016). Message Extension Attack against Authenticated Encryptions: Application to PANDA. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E99.A(1), 49-57.

<https://hdl.handle.net/10356/82092>

<https://doi.org/10.1587/transfun.E99.A.49>

© 2016 Institute of Electronics, Information and Communication Engineers. This paper was published in IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences and is made available as an electronic reprint (preprint) with permission of Institute of Electronics, Information and Communication Engineers. The published version is available at: [<http://dx.doi.org/10.1587/transfun.E99.A.49>]. One print or electronic copy may be made for personal use only. Systematic or multiple reproduction, distribution to multiple locations via electronic or other means, duplication of any material in this paper for a fee or for commercial purposes, or modification of the content of the paper is prohibited and is subject to penalties under law.

Downloaded on 26 Apr 2023 22:59:24 SGT

Message Extension Attack against Authenticated Encryptions: Application to PANDA*

Yu SASAKI^{†a)}, Member and Lei WANG^{††b)}, Nonmember

SUMMARY We present a new cryptanalysis approach to analyze the security of a class of authenticated encryption schemes, which shares similarity with the previous length extension attack against hash-function-based MACs. Hence we name our approach by message extension attack. For an authenticated encryption from the target class, it consists of three phases; initialization with nonce and key as input, state update function with associated data and message as input and tag generation with updated state as input. We will show how to mount a forgery attack in the nonce-repeating model under the chosen-plaintext scenario, when both state update function and tag generation is built based on the same function. To demonstrate the effectiveness of our message extension attack approach, we apply it to a dedicated authenticated encryption called PANDA, which is a candidate of the ongoing CAESAR cryptographic competition. We successfully found an existential forgery attack on PANDA with 2^5 chosen plaintexts, 2^{64} computations, and a negligible memory, and it breaks the claimed 128-bit security for the nonce-repeating model. We note that this is the first result that breaks the security claim of PANDA, which makes it withdrawn from the CAESAR competition by its designer.

key words: message extension attack, internal state recovery, existential forgery, nonce misuse, CAESAR, PANDA

1. Introduction

Authenticated encryption is an all-in-one symmetric-key primitive to provide both of the confidentiality and the integrity of a message at the same time. Such a primitive has several advantages compared to the combinations of encryption schemes and authentication schemes, which are designed independently. For instance, the security evaluation can be carried out inside one scheme. Also it simplifies the proper implementation of the scheme. Moreover, the better performance can be achieved because a part of computations for encryption and authentication can be shared. Therefore, it is both theoretical and practical interesting to study the design of authenticated encryption.

Various design methods of authenticated encryption have been proposed, which is partially due to its generic purpose and complicated security goal. Particularly, the cryptographers are conducting a competition named CAESAR [2], which aims to determine new portfolio of the authenticated encryption. There were 58 candidates submitted

in March 2014, and the final portfolio will be selected in about 4 years through elaborative discussion from various points of view.

As a cryptographic primitive, security is definitely one among the most important criteria of authenticated encryptions. While some of the proposals submitted to CAESAR follows mode-of-operation approach and manage to obtain the security proof, there are quite a portion of proposals, which are mainly of dedicated constructions, that do not have the proof. Instead the designers claim a certain number of bits of security based on their intuition. Such claims must be carefully and continuously analyzed by cryptanalysts. However, cryptanalytic techniques (and proving techniques) for authenticated encryptions do not seem to be developed enough. Hence, we can expect more generic approaches will be discussed.

Among various security issue, *misuse resistance* is an important one, as it aims to provide a fail safe mechanism for incorrect implementations. In details, many authenticated encryptions are nonce-based, and rely the security on a requirement that each nonce value must be used for at most one message under the same key. On the other hand, nonce-misuse resistant schemes guarantee a certain level of security even if the nonce value is incorrectly repeated for multiple messages. Considering the recent incidents such as Lucky Thirteen [3], cryptographers cannot always expect the perfect implementation with very careful security analysis, and thus it is important to provide the fail safe mechanism from the primitive level.

There is one popular approach to design authenticated encryptions: at first use a large state and extract several bits from the state as a key stream, then, generate the ciphertext by XORing the key stream and the message, and finally generate the tag value from the updated state value. The approach has been shown in Fig. 1. Its encryption process shares similarity with stream ciphers. Firstly initialize the state by the key K and nonce N , which will be then updated by associated data A . After that, start to generate the key

Fig. 1 Generic computation class of our target.

Manuscript received March 23, 2015.

Manuscript revised June 12, 2015.

[†]The author is with NTT Secure Platform Laboratories, NTT Corporation, Musashino-shi, 180-8585 Japan.

^{††}The author is with the Nanyang Technological University, Singapore.

*The preliminary version was presented in CANS 2014 [1].

a) E-mail: sasaki.yu@lab.ntt.co.jp

b) E-mail: wang.lei@ntu.edu.sg

DOI: 10.1587/transfun.E99.A.49

stream. Finally produce the tag by processing the length of the associated data, A_{len} , and the length of the message, M_{len} . Additionally, some values may be computed from A and M denoted by A^* and M^* , e.g. the checksum of A or M , and then be input to tag generation function. Using the empty message or a pre-specified fixed message to update the state can be regarded as M^* . In order to optimize the implementation cost, a function to update the state can be shared among the associated data processing part, the message processing part, and the tag generation part.

As shown in Fig. 1, the target class reuses the updated state value by the encryption for the authentication, and is thus very efficient. Besides, the class can achieve the *length optimality*, i.e. the ciphertext length can be exactly the same as the input message length. This class is useful and deserves careful analysis. Indeed, many designs presented so far belong to this class. The duplex sponge mode [4] is an example, and thus all the designs following the duplex sponge mode also belong to this class. Several dedicated designs submitted to CAESAR also belong to this class, e.g. AEGIS [5], [6], LAC [7], PAES [8] and PANDA [9]. Besides, several designs outside CAESAR belong to it as well, e.g. ASC-1 [10], ALE [11], and FIDES [12][†].

We mainly deal with the nonce-misuse resistance of this class. On one hand, the confidentiality of the ciphertext is trivially broken by repeating the same K, N, A . Because the first key stream stays unchanged for the same K, N, A , the plaintext recovery attack will be mounted easily. On the other hand, the internal state value still remains secret. Thus, the integrity may be ensured even for the repetition of the same nonce value. The CAESAR candidate PANDA aims to achieve this goal. PANDA-s, which is one of the members of the PANDA-family, is claimed to be nonce-misuse resistant with respect to the integrity.

Our Contributions

In this paper, we present a new cryptanalytic approach to recover the internal state value or to break the integrity of authenticated encryptions belonging to the class in Fig. 1 in the nonce-repeating model. The overall idea is inspired by the length extension attack for hash function based MACs specified in [13], thus we call it the *message extension attack*.

The message extension attack firstly aims to recover the internal state value between the message processing part and the tag generation part. Let s and n be the size of the internal state and the tag, respectively. The attacker first observes an n -bit tag T for any pair of (A, M) . The attacker can obtain n bits out of s bits of the state after the tag generation function from T . Here, we have two observations: 1) input values to the tag generation function, i.e. $A_{len}, M_{len}, A^*, M^*$, are usually derived only from (A, M) , and 2) functions to update the internal state are usually identical in the message processing

part and in the tag generation part in order to optimize the implementation cost. Whenever these observations apply, the attacker can extend the message M so that the input to the tag generation function is appended to the end of M . For this new message under the same (K, N, A) , the attacker can obtain n bits of the target state. Once the internal state is fully recovered, the tag generation function can be simulated offline and thus the existential forgery, which produces a valid pair of (N, A, M) and (C, T) , can be performed only by accessing the encryption oracle.

Note that several cryptanalytic results have already been proposed for ALE [14], [15], FIDES [16] and PAES [17]–[20] that belong to the same class. We stress that the message extension attack is different from those attacks that are differential based approach mainly accessing to the decryption oracle.

We then apply the message extension attack to PANDA-s, which belongs to the class in Fig. 1 and claims 128-bit security in the nonce-repeating model. The internal state size for PANDA-s is 448 bits. A unique design feature of PANDA-s is that 192 bits of the state are not simply affected from the message input, and thus cannot be recovered. This prevents the simple application of the message extension attack. We then analyze the details of the round function of PANDA-s. By exploiting the key stream information, we show that the existential forgery attack is mounted with 2^5 chosen plaintexts, 2^{64} computations, and a negligible memory. This is the first result that breaks the security claim of PANDA.

Current Status of PANDA

Between the preprint version (25 March 2014) [21] and the formal publication of the preliminary version of this paper [1], Feng et al. uploaded unreviewed work of the cryptanalysis on PANDA-s which requires 2^{64} computational cost but can work in the nonce-respecting model under the known-plaintext scenario [22] (06 May 2014). In a few days later (10 May 2014), Feng et al. improved their attack complexity to 2^{41} and claimed a practical forgery attack against PANDA-s [23].

PANDA was withdrawn from the CAESAR competition on 13 May 2014.

Paper Outline

The organization of this paper is as follows. We describe message extension attack in a generic form in Sect. 2. We then apply it to mount a forgery attack on PANDA-s in Sect. 3. Possible countermeasures are discussed in Sect. 4. Finally, we conclude the paper in Sect. 5.

2. Generic Approach with Message Extension Attack

In this section we introduce our approach for recovering the internal state value and forging the tag value against the computation structure in Fig. 1. More precisely, our goal

[†]In Sect. 4, we identify features of some of these designs that make the attack presented in this paper in applicable.

is finding a valid pair of (N, A, M) and (C, T) which has not been queried before, i.e. the existential forgery attack in the nonce-repeating model in the chosen-plaintext scenario. Note that the attacker only needs to interact with the encryption oracle. The decryption oracle is never accessed.

We denote the lengths of the associated data A and the message M as A_{len} and M_{len} , respectively. We also denote four parts of the class as follows.

F_{ini} : generates an s -bit state from the initial value IV , key K , and nonce N .

F_{AD} : updates the s -bit state depending on the associated data A .

F_M : updates the s -bit state depending on the message M , and generates the key stream of M_{len} bits. In practice, M_{len} bits of the key stream are not generated at once. A small function generating a certain bit size, say b bits, of the key stream is iteratively performed until M_{len} bits are generated.

F_T : computes an n -bit tag value depending on the s -bit state, A_{len} , and M_{len} . It can also take as input some values computed from A or M , e.g. the checksum of A or M . The class in Fig. 1 assumes that the input value to F_T can be determined easily when A and M are fixed. We denote its value by $\mathcal{I}(A, M)$.

With the message extension attack, we aim to recover n bits of internal state value which is input to the tag generation function F_T by exploiting the tag value obtained by queries of different messages. One natural approach is utilizing the last key stream value, which can be recovered by taking XOR of the last b bits of the message and the ciphertext. Because the key stream is generated from the state, it may derive at most b -bit information about the internal state value. The difficulty of this attempt is that the size of b is usually much smaller than the state size s and thus is not enough to recover the entire state. For instance, the sizes of b and s are 10 bits and 160 bits for FIDES respectively, and 64 bits and 448 bits for PANDA-s respectively. By using the key stream for multi-blocks, the amount of known bits increases. However, this also increases the amount of unknown bits. Thus, even with multi-blocks, the ratio of the number of known bits to the number of unknown bits does not increase.

Our observation is that when the computations to update the state are identical between F_M and F_T , we can utilize the n -bit tag value to recover the internal state, in which n is often larger than b , and thus the attacker can obtain more information. For instance, the tag size n of FIDES and PANDA-s are 80 bits and 128 bits, respectively.

Intuitively the attack works as follows. We first choose the associated data A and the message M , and then compute the corresponding input value to F_T , i.e. $\mathcal{I}(A, M)$. Then, the input value for updating the state in F_M and F_T is represented as $M||\mathcal{I}(A, M)$. We then choose another message M' such that $M' \leftarrow M||\mathcal{I}(A, M)$. That is, M is extended to M' by appending its input value to F_T . This is illustrated in Fig. 2. n bits of the state after the F_M function for M'

Fig. 2 Generic approach with message extension attack.

is already known to the attacker as the tag value T for M . This gives significant information to the attacker about the internal state value.

How many internal state bits are directly recovered from the obtained n -bit information is dependent on the specification. If the tag is generated by the truncation of the state, it immediately derives n bits of the state. If the tag is generated by some function of the state, it does not reveal the state bits directly. For example, PANDA generates the tag by taking the XOR of internal state bits. Thus, the attacker can know the result of the XOR, but cannot know the state bits directly. However, as long as the corresponding n -bit information of the state can be computed from the observed n -bit tag, i.e. the function to generate the tag from the state is invertible, n -bit information is derived.

Security of the class shown in Fig. 1 is often claimed up to $s/2$ -bit queries. Hence, the parameter $s = 2n$ is very natural. With the recovered n bits, the remaining unknown bit size is n bits. To recover those n bits efficiently[†], we need more information. Let M'' be a message in which another message block is appended to M' , i.e. $M'' = M' || X$. Then, besides the obtained n -bit information, another b -bit key stream generated by the target state is obtained. How the b -bit key stream relates to the s -bit state depends on the specification. Here we suppose that the size of the exhaustive search is reduced from n bits to $n - b$ bits. For example, when b -bit key stream is unknown state bits of the state or the XOR of the several unknown state bits, the assumption can hold.

The correctness of the exhaustive guess on $n - b$ bits can be verified by computing F_T offline, and by checking the n -bit match of the result and T' (the tag value for M'). Moreover, by appending longer X , the key stream can also be used for the verification. Thus, false positives can be eliminated. Once the internal state value is recovered, the attacker can forge the tag value for any message starting from M' , namely any message of the form $M' || Y$, where Y can be an arbitrary string but for already queried X .

The attack also needs to obtain the ciphertext of the message to be forged. This is easily obtained by querying

[†]A forgery attack may be able to be performed without recovering all the internal state value. For generality, the attack goal is set to recovering the $2n$ -bit state value.

a message in which at least 1 random block is appended to the end. The ciphertext for the extended message contains the ciphertext for the message to be forged. The attacker can copy it.

The detailed attack procedure is described in Algorithm 1.

Algorithm 1 Existential Forgery with Message Extension Attack

Output: a valid pair of (N, A, M) and (C, T)

- 1: Choose values of N , A , and M at random.
 - 2: Query (N, A, M) to the encryption oracle to obtain the corresponding C and T .
 - 3: Compute the corresponding $I(A, M)$ offline and set $M' \leftarrow M \parallel I(A, M)$.
 - 4: Query (N, A, M') to the encryption oracle to obtain the corresponding C' and T' .
 - 5: Compute the corresponding $I(A, M')$ offline.
 - 6: Query $M' \parallel X$ for a random message block X to obtain b -bit key stream.
 - 7: **for** the remaining $s - n - b$ bits of the state after F_M for processing M' **do**
 - 8: With the knowledge of the state and $I(A, M')$, compute F_T offline.
 - 9: **if** the result of F_T and received T' match **then**
 - 10: Choose the value of message $Y (\neq X)$ to be appended at random.
 - 11: Set $M'' \leftarrow M' \parallel Y$, and compute the corresponding $I(A, M'')$ offline.
 - 12: With the knowledge of the state and $I(A, M'')$, compute F_T offline. Let T'' be the resulted output from F_T .
 - 13: Choose 1-block string Z and query $M'' \parallel Z$ to obtain the ciphertext $C'' \parallel C_Z$.
 - 14: **return** (N, A, M'') and (C'', T'') .
 - 15: **end if**
 - 16: **end for**
-

As shown in Algorithm 1, the attack requires to repeat the same nonce three times. Hence, the attack can work only in the nonce-repeating model. The attacker needs to specify the queried message. Therefore, the attack only works in the chosen-plaintext scenario. In the end, the attacker can find a valid pair of (N, A, M'') and (C'', T'') which have not been queried before. Thus, the attack breaks the notion of the existential forgery attack only by accessing the encryption oracle.

The data complexity is 3 chosen-plaintext queries. Note that the first query does not require to specify any particular value. Thus, instead of making queries, eavesdropping the communicated data is sufficient. In this sense, the precise data complexity is 1 known-plaintext query and 2 chosen-plaintext queries. The data complexity is usually measured by the number of queried message blocks, which cannot be determined without giving the detailed specification of F_T . Thus, we only count the number of queries here, and later will discuss the details in the application to a specific design. The time complexity is 2^{s-n-b} computations of F_T . The memory requirement is negligible.

3. Application: Existential Forgery Attack on PANDA-s

In this section, we apply the message extension attack for

PANDA-s, which is claimed to be nonce-misuse resistant with respect to integrity. In Sect. 3.1, we briefly introduce the specification of PANDA-s. Its computation structure belongs to the class in Fig. 1, with the state size $s = 448$ and the tag size $n = 128$. Because n is not big enough, we cannot attack PANDA-s with the straight-forward application of the generic approach. In Sect. 3.2, we improve the attack by looking inside the round function of PANDA-s. By further exploiting the key stream value, we can successfully mount the existential forgery attack.

3.1 Specification of PANDA-s

PANDA-s is a member of the PANDA-family [9] designed by Ye et al. which was submitted to CAESAR. The PANDA-s encryption function takes a 128-bit key K , a 128-bit nonce N , variable length associated data A , and variable length plaintext M as input, and outputs the ciphertext C and a 128-bit tag T .

The encryption function consists of 4 parts: initialization, processing associated data, processing plaintext, and finalization, which are computed in this order. The computation structure is illustrated in Fig. 3, where the bit size of each arrow line is 64 bits. 64-bit values are called “blocks” in PANDA-s.

3.1.1 Initialization

A 128-bit key K and a 128-bit nonce N are mixed and expanded to 448-bit internal state. We omit the details due to the irrelevance to our attack.

3.1.2 Processing Associated Data

The associated data A is first padded to a multiple of 64 bits $(A_0, A_1, \dots, A_{a-1})$, and then processed block by block with the round function RF . The round function RF of PANDA-s generally takes an 8-block (or 512-bit) value as input, of which 448 bits are for the previous internal state value and 64 bits are for mixing new input data to the state. The output of RF is either a 7-block value (updated internal state) or a 8-block value (updated internal state and 1-block key stream). We denote the round function by RF_7 when the output size is 7 blocks, and by RF_8 when the output size is

Fig. 3 Computation structure of PANDA-s.

Fig. 4 Round function for 7-block output.

Fig. 5 Round function for 8-block output.

SN and LT stand for SubNibbles and LinearTrans, respectively.

8 blocks.

In RF_7 , a 7-block internal state value is split into seven 1-block variables $w, x, y, z, S^{(0)}, S^{(1)}, S^{(2)}$. Let m be another 1-block input value. Then, the updated state value $w', x', y', z', S'^{(0)}, S'^{(1)}, S'^{(2)}$ are computed as follows:

$$\begin{aligned} w' &\leftarrow \text{SubNibbles}(w \oplus x \oplus m), \\ x' &\leftarrow \text{SubNibbles}(x \oplus y), \\ y' &\leftarrow \text{SubNibbles}(y \oplus z), \\ z' &\leftarrow \text{SubNibbles}(S^{(0)}), \\ (S'^{(0)}, S'^{(1)}, S'^{(2)}) &\leftarrow \text{LinearTrans}(S^{(0)} \oplus w, S^{(1)}, S^{(2)}), \end{aligned}$$

where SubNibbles is a parallel application of a 4-bit S-box and LinearTrans applies a linear transformation. We omit their full specifications due to the irrelevance to our attack. RF_7 is illustrated in Fig. 4. Finally, by taking the 7-block state value after the initialization, $state$, as input, the associated data is processed by computing $state \leftarrow RF_7(state, A_i)$ for $i = 0, 1, \dots, a - 1$.

3.1.3 Processing Plaintext

The plaintext M is first padded to a multiple of 64 bits ($M_0, M_1, \dots, M_{m-1}$), and then processed block by block with the round function RF_8 . RF_8 is almost the same as RF_7 . The only difference is that it produces another 1-block output value r by $r \leftarrow x \oplus x'$ as illustrated in Fig. 5.

The additional 1-block output value r_i in round i is used as a key stream. Namely, the ciphertext block C_i for the plaintext block M_i is computed by $C_i \leftarrow M_i \oplus r_i$. Finally, by taking the 7-block state value after the associated data processing, $state$, as input, the plaintext is processed as follows:

Table 1 Bits of security for PANDA-s in two models [9, Table 3.1].

Goal	Model	
	Nonce-respect	Nonce-repeat
confidentiality for the plaintext	128	/
integrity for the plaintext	128	128
integrity for the associated data	128	128
integrity for the public message number	128	128

```

 $(state, r_0) \leftarrow RF_8(state, 0),$ 
for  $i = 0$  to  $m - 1$  do
   $C_i \leftarrow M_i \oplus r_i,$ 
   $(state, r_{i+1}) \leftarrow RF_8(state, M_i).$ 
end for

```

The extra key stream r_m after the last message block M_{m-1} is discarded.

3.1.4 Finalization

In the finalization, the state is updated by using the bit length of the associated data A_{len} and the bit length of the plaintext M_{len} . Let Len_i be A_{len} when i is even and M_{len} when i is odd. The finalization consists of 14-round state update by using Len_i and the tag generation to produce a 128-bit tag T . In details, the finalization computes the following operations.

```

for  $i = 0$  to  $13$  do
   $state \leftarrow RF_7(state, Len_i).$ 
end for
 $T \leftarrow (w \oplus y) \parallel (x \oplus z).$ 

```

3.1.5 Claimed Security

The claimed security of PANDA-s is given in Table 1. In particular, 128-bit security is claimed for integrity in the nonce-repeating model.

3.2 Existential Forgery Attack against PANDA-s

We first observe that the following properties of PANDA-s enable the application of the message extension attack.

1. The computations for updating the state by RF_7 and RF_8 are exactly the same, i.e. $RF_7(state, m)$ and $RF_8(state, m)$ produce the same state value.
2. The 1-block input values in the finalization, A_{len} and M_{len} , can be computed by the attacker only from A and M .

3.2.1 Overall Strategy and Message Choices

We start by applying the message extension attack to PANDA-s. The attack requires only two encryption oracle calls under the same key and nonce. The associated data can be fixed to any value. Hereafter, we always set A to Null for simplicity. We set the first query Q_1 as $Q_1 \leftarrow \alpha$, where α can be any string. Let $\ell(\alpha)$ be the 14-block value that will be processed in the finalization part for α , i.e. $\ell(\alpha) = A_{len} \parallel M_{len} \parallel \dots \parallel A_{len} \parallel M_{len}$.

By following the generic approach in Sect.2, we should set the second query Q_2 as $Q_2 \leftarrow \alpha \parallel \ell(\alpha)$. However, the 128-bit tag for Q_1 only reveals 128 bits of the 448-bit state after the processing plaintext part for Q_2 . As later explained, we aim to recover 256 bits of the state. Hence, we need more information, at least additional 128 bits, of the state. Here, our strategy is appending 2 more message blocks after $\alpha \parallel \ell(\alpha)$, and obtaining 2-block (128-bit) key stream value, which allows us to recover 256 bits of the state. Hence, we set $Q_2 \leftarrow \alpha \parallel \ell(\alpha) \parallel \beta$, where β can be any string as long as its length is at least 2 blocks. Then, only with 2 queries of Q_1 and Q_2 , 256 bits of the state after $\alpha \parallel \ell(\alpha)$ are recovered.

We then choose a target message M to be forged. By following the generic approach, we should append some string X to $\alpha \parallel \ell(\alpha)$. Namely, $M \leftarrow \alpha \parallel \ell(\alpha) \parallel X$. Then $\ell(\alpha \parallel \ell(\alpha) \parallel X)$ is processed during the finalization. The generic approach in Sect.2 computes the state while X and $\ell(\alpha \parallel \ell(\alpha) \parallel X)$ are processed. However, because we only partially recover the state (256 bits out of 448 bits), the generic approach cannot work for PANDA-s. As later explained, to recover the internal state, we need the corresponding key stream value. While $\ell(\alpha \parallel \ell(\alpha) \parallel X)$ is processed in the finalization, the key stream is never generated, and thus the internal state cannot be recovered. Here, our strategy is using a query Q_3 in which $\alpha \parallel \ell(\alpha) \parallel X \parallel \ell(\alpha \parallel \ell(\alpha) \parallel X)$ is included inside. Then, the key stream is obtained while $\ell(\alpha \parallel \ell(\alpha) \parallel X)$ is processed, and thus the corresponding internal state can be recovered. Finally, by setting $M \leftarrow \alpha \parallel \ell(\alpha) \parallel X$, the tag for M can be computed offline. Note that, precisely speaking, we need at least 1 block after $\alpha \parallel \ell(\alpha) \parallel X \parallel \ell(\alpha \parallel \ell(\alpha) \parallel X)$ in Q_3 by a similar reason to append 2 blocks in Q_2 .

The purpose of Q_2 and Q_3 can be achieved by 1 query. Namely, we set $Q_2 \leftarrow \alpha \parallel \ell(\alpha) \parallel \beta \parallel \ell(\alpha \parallel \ell(\alpha) \parallel \beta) \parallel \gamma$, where β is any string and γ is any string as long as its length is at least 1 block. Finally, the tag for $M \leftarrow \alpha \parallel \ell(\alpha) \parallel \beta$ can be computed offline. In summary, we choose the queries of the following forms:

$$\begin{aligned} Q_1 &\leftarrow \alpha, \\ Q_2 &\leftarrow \alpha \parallel \ell(\alpha) \parallel \beta \parallel \ell(\alpha \parallel \ell(\alpha) \parallel \beta) \parallel \gamma, \\ M &\leftarrow \alpha \parallel \ell(\alpha) \parallel \beta. \end{aligned}$$

Because M is included inside Q_2 , the ciphertext of M can be directly copied from a part of the ciphertext of Q_2 .

As you can see, except for 14 blocks (112 bytes) of $\ell(\alpha)$, any message can be the target of our forgery attack. Therefore, though this attack is the existential forgery attack, it can forge the tag for a huge variety of messages.

3.2.2 Recovering 256-bit Internal State After Processing $\alpha \parallel \ell(\alpha)$

The attacker first queries the message $Q_1 = \alpha$ and obtains the corresponding tag T_1 . This reveals some information about the internal state x, y, z, w after processing $\alpha \parallel \ell(\alpha)$. The attacker then appends the message block

Fig. 6 Recovery of 256 bits of the internal state.

$\beta = \beta_0 \parallel \beta_1 \parallel \dots$. Let the internal state value after processing $\alpha \parallel \ell(\alpha)$ be $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0}, S_{\beta_0}^{(0)}, S_{\beta_0}^{(1)}, S_{\beta_0}^{(2)})$. Remember that the tag is computed by $T \leftarrow (w \oplus y) \parallel (x \oplus z)$. The obtained tag T_1 indicates that the internal state value satisfies the following equations.

$$w_{\beta_0} \oplus y_{\beta_0} = T_1^L, \quad (1)$$

$$x_{\beta_0} \oplus z_{\beta_0} = T_1^R, \quad (2)$$

where T_1^L and T_1^R are 64-bit values satisfying $T_1^L \parallel T_1^R = T_1$.

Then, the attacker queries $Q_2 = \alpha \parallel \ell(\alpha) \parallel \beta \parallel \ell(\alpha \parallel \ell(\alpha) \parallel \beta) \parallel \gamma$, and obtains the corresponding ciphertext block and tag T_2 . The computation to process β is shown in Fig. 6. As a result of $RF_8(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0}, S_{\beta_0}^{(0)}, S_{\beta_0}^{(1)}, S_{\beta_0}^{(2)}, \beta_0)$, the attacker obtains the ciphertext block C_{β_0} , which is computed by $r_{\beta_0} \oplus M_{\beta_0}$. Hence, the key stream value r_{β_0} can be computed as $M_{\beta_0} \oplus C_{\beta_0}$. From the computation structure of the key stream, the attacker obtains the equation

$$r_{\beta_0} = x_{\beta_0} \oplus \text{SubNibbles}(x_{\beta_0} \oplus y_{\beta_0}). \quad (3)$$

Here, the attacker guesses the 64-bit value of x_{β_0} . For each guess, the corresponding z_{β_0} is obtained from Eq. (2), the corresponding y_{β_0} is obtained from Eq. (3), and the corresponding w_{β_0} is obtained from Eq. (1). Hence, for each guess of x_{β_0} , 256-bit internal state value $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$ is determined. Moreover, the knowledge of $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$ leads to the knowledge of $w_{\beta_1}, x_{\beta_1}, y_{\beta_1}, w_{\beta_2}, x_{\beta_2}$, and r_{β_1} . These give another 64-bit relation

$$r_{\beta_1} = x_{\beta_1} \oplus \text{SubNibbles}(x_{\beta_1} \oplus y_{\beta_1}), \quad (4)$$

and 1 guess of x_{β_0} out of 2^{64} possibilities is expected to satisfy this equation. Therefore, the 256-bit internal state value $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$ is almost uniquely determined. In Fig. 6, the focused variables up to here are stressed by bold circles. Note that with some probability, several candidates of $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$ will remain. Those false positives only give negligible impact to the attack complexity. Moreover, if the key leak value is obtained for a few more message blocks, those false positives can be eliminated easily.

The procedure is also described in Algorithm 2. We note that our attack is irrelevant to the specification of SubNibbles, that is even if SubNibbles is perfectly secure, our attack is still applicable.

Algorithm 2 256-bit internal state recovery after processing $\alpha \parallel \ell(\alpha)$

Output: internal state bits w, x, y and z

- 1: Query message α to obtain corresponding tag $T_1 = T_1^L \parallel T_1^R$.
 - 2: Query message $\alpha \parallel \ell(\alpha) \parallel \beta$ to receive the ciphertext C .
 - 3: Compute $r_{\beta_0} = C_{\beta_0} \oplus M_{\beta_0}$.
 - 4: Compute $r_{\beta_1} = C_{\beta_1} \oplus M_{\beta_1}$.
 - 5: **for** all possible values of x_{β_0} **do**
 - 6: Compute $z_{\beta_0} = x_{\beta_0} \oplus T_1^R$.
 - 7: Compute $y_{\beta_0} = x_{\beta_0} \oplus \text{SubNibbles}^{-1}(r_{\beta_0} \oplus x_{\beta_0})$.
 - 8: Compute $w_{\beta_0} = y_{\beta_0} \oplus T_1^L$.
 - 9: Compute $w_{\beta_1} = \text{SubNibbles}(w_{\beta_0} \oplus M_{\beta_0} \oplus x_{\beta_0})$.
 - 10: Compute $x_{\beta_1} = \text{SubNibbles}(x_{\beta_0} \oplus y_{\beta_0})$.
 - 11: Compute $y_{\beta_1} = \text{SubNibbles}(y_{\beta_0} \oplus z_{\beta_0})$.
 - 12: Compute $r'_{\beta_1} = x_{\beta_1} \oplus \text{SubNibbles}(x_{\beta_1} \oplus y_{\beta_1})$.
 - 13: **if** $r_{\beta_1} = r'_{\beta_1}$ **holds then**
 - 14: **return** $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$.
 - 15: **end if**
 - 16: **end for**
-

3.2.3 Recovering w, x, y, z in All Rounds

In the round function of PANDA-s, the input message affects the right most 3 blocks (=192 bits) of the state in a complicated way through LinearTrans. Therefore, without the exhaustive guess for those 192 bits, recovering all 448-bit internal state seems impossible. However, 2^{192} is already more expensive than the cost of the brute force attack.

To solve this problem, our strategy is to keep revealing the 256-bit state values w, x, y, z for all rounds by using the knowledge of $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$. This can be done with negligible cost for PANDA-s. Let us recall Fig. 6. Once the 256-bit state values w_i, x_i, y_i, z_i are recovered for some i , the attacker can easily obtain 192-bit state values for the next round, i.e. $w_{i+1}, x_{i+1}, y_{i+1}$. Therefore, if the 64-bit state value z_{i+1} can be recovered efficiently, the attacker can reveal the 256-bit state values w, x, y, z for any number of rounds. In Fig. 6, in order to recover the 64-bit value of z_{β_1} , the attacker uses the key stream value after 1 round, i.e. r_{β_2} . In details, the attacker focuses on the following 64-bit relation.

$$r_{\beta_2} = x_{\beta_2} \oplus \text{SubNibbles}(x_{\beta_2} \oplus y_{\beta_2}),$$

$$= x_{\beta_2} \oplus \text{SubNibbles}(x_{\beta_2} \oplus \text{SubNibbles}(y_{\beta_1} \oplus z_{\beta_1})).$$

The above equation is converted to

$$z_{\beta_1} = y_{\beta_1} \oplus \text{SubNibbles}^{-1}(\text{SubNibbles}^{-1}(r_{\beta_2} \oplus x_{\beta_2}) \oplus x_{\beta_2}). \quad (5)$$

Then, z_{β_1} is recovered only with 1 computation. In Fig. 6, the focused variables to recover z_{β_1} are stressed by bold lines. By iterating the same procedure for all the subsequent blocks, the attacker can recover $(w_{\beta_i}, x_{\beta_i}, y_{\beta_i}, z_{\beta_i})$ for any block-length i as long as the key stream for the next block, $r_{\beta_{i+1}}$, is obtained.

3.2.4 Forging Tag

Due to the message structure of Q_2 , the attacker can recover the internal state after $\alpha \parallel \ell(\alpha) \parallel \beta \parallel \ell(\alpha \parallel \ell(\alpha) \parallel \beta)$ is processed. Note that the length of γ must be at least 1 block so that the internal state after the last block of $\alpha \parallel \ell(\alpha) \parallel \beta \parallel \ell(\alpha \parallel \ell(\alpha) \parallel \beta)$ can be recovered. Then, the tag value for a new message $M (= \alpha \parallel \ell(\alpha) \parallel \beta)$ is easily computed by $(w \oplus y) \parallel (x \oplus z)$ of this state. Because M is included inside Q_2 , the ciphertext C can be copied from the ciphertext of Q_2 .

3.2.5 Complexity Evaluation

The attack requires 2 encryption oracle calls in the chosen-plaintext scenario and the nonce repeating model. The number of queried message blocks is minimized when we set $\alpha, \beta \leftarrow \text{Null}$ and $|\gamma| \leftarrow 1$. Then, the number of queried message blocks is 1 for Q_1 and $0 + 14 + 0 + 14 + 1 = 29$ for Q_2 , in total 30 blocks. Therefore, the data complexity is about 2^5 chosen-plaintext message blocks. To recover the 256-bit state $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$, 2^{64} computational cost is required. Then, all the remaining cost is 1. Thus, the time complexity of this attack is less than 2^{64} PANDA-s computations. The memory requirement is to store all the ciphertext blocks and the tag, which is negligible.

3.2.6 Attack Procedure

The attack procedure for the parameter $\alpha, \beta \leftarrow \text{Null}$ and $|\gamma| \leftarrow 1$ in the algorithmic form is given in Algorithm 3. For simplicity, the associated data is supposed to be Null.

4. Countermeasures

The message extension attack seems to be possible to prevent with some extra cost. To apply the message extension attack, two conditions must be satisfied.

1. The functions to update the state in F_M and in F_T are identical.
2. $I(A, M)$ can be derived only from A and M .

In order to break the first condition, giving a small tweak to the state, e.g. XORing a constant at the beginning of F_T , is a possible option. The XOR must be done

Algorithm 3 Existential Forgery Attack on PANDA-s

Input: nonce N , $Q_1 = \text{Null}$, $Q_2 = \ell(\text{Null}) \parallel \ell(\ell(\text{Null})) \parallel \gamma$
Output: ciphertext C and tag $\mathcal{T}$ in which $(N, C, \mathcal{T})$ is valid

```

1: Query  $Q_1$  to the encryption oracle to obtain the tag  $T_1 = T_1^L \parallel T_1^R$ .
2: Query  $Q_2$  to obtain the key stream  $r_{\beta_0}, r_{\beta_1}, \dots, r_{\beta_{14}}$  and set  $C$  to the first
   14 blocks of the ciphertext of  $Q_2$ .
3: for  $2^{64}$  guesses of  $x_{\beta_0}$  do
4:   Compute  $w_{\beta_0}, y_{\beta_0}, z_{\beta_0}$  with equations (1),(2),(3).
5:   if Equation (4) is satisfied then
6:     Fix the values of  $w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0}$ .
7:   end if
8: end for
9: for  $i = 1, 2, \dots, 14$  do
10:   With  $r_{\beta_{i+1}}$  and equation (5), compute  $z_{\beta_i}$  to recover  $(w_{\beta_i}, x_{\beta_i}, y_{\beta_i}, z_{\beta_i})$ .
11: end for
12: Set  $\mathcal{T} \leftarrow (w_{\beta_{14}} \oplus y_{\beta_{14}}) \parallel (x_{\beta_{14}} \oplus z_{\beta_{14}})$ 
13: return  $(C, \mathcal{T})$ .
```

to the state which is not directly updated by the message input. Otherwise, the impact of the tweak can be canceled by the attacker by modifying the message. For example, for PANDA-s, it is hard for the attacker to control the impact to $S^{(0)}, S^{(1)}, S^{(2)}$ from a message block. Therefore, if a constant is XORed to one of $S^{(0)}, S^{(1)}, S^{(2)}$ at the beginning of F_T , the attacker cannot cancel its impact and thus the message extension attack can be prevented. Indeed, AEGIS adopts such a mechanism, i.e. XORing some constant at the begging of F_T . This prevents the message extension attack on AEGIS though it does not claim the security in the nonce-repeating model to begin with.

In order to break the second condition, the simplest way is to use the key K in F_T . Actually, many of known secure block-cipher based MACs run the full-round encryption both in the beginning and the end, and many of hash function based MACs use the key both in the beginning and the end. Compared to those, using K in F_T may be a reasonable extra cost to increase the security. Actually, the message extension attack can be prevented by masking the final output with a key-dependent value.

5. Concluding Remarks

In this paper, we proposed the new approach called message extension attack for a class of authenticated encryptions, which includes many of currently discussed designs. The message extension attack aims to mount the internal state recovery attack or the existential forgery attack only with the encryption oracle in the nonce-repeating model. The attack exploits the similarity of the state updating function for processing the message and generating the tag.

We applied the message extension attack to PANDA-s, which is one of the designs submitted to the CAESAR competition. Due to the state size and the computation structure particular to PANDA-s, the simple application cannot work. With some detailed analysis, we found that the forgery attack can be performed with 2^5 chosen plaintexts, 2^{64} computational complexity, and negligible memory. The result clearly breaks the designers' security claim of PANDA-s.

To apply the message extension attack, several conditions must be satisfied. Thus it can be prevented with a small cost. Nevertheless, we believe that the message extension attack is an useful object to learn. Accumulating the knowledge of generic approaches at this stage is important to discuss the authenticated encryption security in future. We hope that future authenticated encryption designers understand the approach and make their designs resistant to it.

Acknowledgments

The authors would like to thank anonymous reviewers of CANS 2014 for their helpful comments. Lei Wang is supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

References

- [1] Y. Sasaki and L. Wang, "Message extension attack against authenticated encryptions: Application to PANDA," *Cryptology and Network Security, Lecture Notes in Computer Science*, vol.8813, pp.82–97, Springer, 2014.
- [2] D. Bernstein, CAESAR Competition, <http://competitions.cr.yp.to/caesar.html>, 2013.
- [3] N.J.A. Fardan and K.G. Paterson, "Lucky thirteen: Breaking the TLS and DTLS record protocols," *2013 IEEE Symposium on Security and Privacy*, pp.526–540, 2013.
- [4] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Duplexing the sponge: Single-pass authenticated encryption and other applications," *Selected Areas in Cryptography, Lecture Notes in Computer Science*, vol.7118, pp.320–337, Springer, 2012.
- [5] H. Wu and B. Preneel, "AEGIS: A fast authenticated encryption algorithm," *Selected Areas in Cryptography — SAC 2013, Lecture Notes in Computer Science*, vol.8282, pp.185–201, Springer, 2014.
- [6] H. Wu and B. Preneel, "AEGIS: A fast authenticated encryption algorithm (v1)," Submitted to the CAESAR competition, 2014.
- [7] L. Zhang, W. Wu, Y. Wang, S. Wu, and J. Zhang, "LAC: A lightweight authenticated encryption cipher," Submitted to the CAESAR competition, 2014.
- [8] D. Ye, P. Wang, L. Hu, L. Wang, Y. Xie, S. Sun, and P. Wang, "PAES v1," Submitted to the CAESAR competition, 2014.
- [9] D. Ye, P. Wang, L. Hu, L. Wang, Y. Xie, S. Sun, and P. Wang, "PANDA v1," Submitted to the CAESAR competition, 2014.
- [10] G. Jakimoski and S. Khajuria, "ASC-1: An authenticated encryption stream cipher," *Selected Areas in Cryptography, Lecture Notes in Computer Science*, vol.7118, pp.356–372, Springer, 2012.
- [11] A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, and E. Tischhauser, "ALE: AES-based lightweight authenticated encryption," *Fast Software Encryption, Lecture Notes in Computer Science*, vol.8424, pp.447–466, Springer, 2014.
- [12] B. Bilgin, A. Bogdanov, M. Knežević, F. Mendel, and Q. Wang, "FIDES: Lightweight authenticated cipher with side-channel resistance for constrained hardware," *Cryptographic Hardware and Embedded Systems — CHES 2013, Lecture Notes in Computer Science*, vol.8086, pp.142–158, Springer, 2013.
- [13] U.S. Department of Commerce, National Institute of Standards and Technology: Federal Register, vol.72, no.212, Friday, Nov. 2, 2007, Notices 2007. http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf
- [14] D. Khovratovich and C. Rechberger, "The LOCAL attack: Cryptanalysis of the authenticated encryption scheme ALE," *Selected Areas in Cryptography — SAC 2013, Lecture Notes in Computer Science*, vol.8282, pp.174–184, Springer, 2014.
- [15] S. Wu, H. Wu, T. Huang, M. Wang, and W. Wu, "Leaked-state-

- forgery attack against the authenticated encryption algorithm ALE,” *Advances in Cryptology—ASIACRYPT 2013, Lecture Notes in Computer Science*, vol.8269, pp.377–404, Springer, 2013.
- [16] I. Dinur and J. Jean, “Cryptanalysis of FIDES,” *Fast Software Encryption, Lecture Notes in Computer Science*, vol.8540, pp.224–240, Springer, 2015.
 - [17] Y. Sasaki and L. Wang, “A practical universal forgery attack against PAES-8,” *Cryptology ePrint Archive, Report 2014/218*, 2014. <https://eprint.iacr.org/2014/218>
 - [18] J. Jean and I. Nikolić, “Using AES round symmetries to distinguish PAES,” http://www1.spms.ntu.edu.sg/~syllab/m/images/6/6e/Using_AES_Round_Symmetries_to_Distinguish_PAES.pdf, 2014.
 - [19] M.J.O. Saarinen, “PAES and rotations,” <https://groups.google.com/forum/#!topic/crypto-competitions/vRmJdRQBzOo>, 2014.
 - [20] J. Jean, I. Nikolić, Y. Sasaki, and L. Wang, “Practical cryptanalysis of PAES,” *Selected Areas in Cryptography—SAC 2014, Lecture Notes in Computer Science*, vol.8781, pp.228–242, Springer, 2014.
 - [21] Y. Sasaki and L. Wang, “A forgery attack against PANDA-s,” *Cryptology ePrint Archive, Report 2014/217*, 2014. <https://eprint.iacr.org/2014/217>
 - [22] X. Feng, F. Zhang, and H. Wang, “A forgery and state recovery attack on the authenticated cipher PANDA-s,” *Cryptology ePrint Archive, Report 2014/325*, 2014.
 - [23] X. Feng, F. Zhang, and H. Wang, “A practical forgery and state recovery attack on the authenticated cipher PANDA-s,” http://www.amss.ac.cn/xwdt/kydt/201405/t20140506_4109871.html, 2014.

Yu Sasaki received the B.E., M.E. and Ph.D. from The University of Electro-Communications in 2005, 2007, and 2010. Since 2007, he has been a researcher at NTT Secure Platform Laboratories. His current research interests are in cryptography. He was awarded a paper prize from SCIS 2007 and IEICE Trans. in 2009. He also received a best paper award from IWSEC 2009, SECRIPT 2012, and IWSEC 2012.

Lei Wang received the M.E. and Ph.D. from The University of Electro-Communications in 2009, and 2011, respectively. His current research interests are in cryptography. He was awarded a paper prize from SCIS 2008 and IEICE Trans. in 2009. He also received a best paper award from IWSEC 2009 and IWSEC 2012.