A Forgery Attack on Leaked-State Authenticated Encryption

Jieshen Mao[1,2], Daoguang Mu[2], Xuejia Lai[1]
1, Institute of Cryptology and Information Security, Dept of Computer Science and Technology
Shanghai Jiaotong University, Shanghai, China
2, Science and Technology on Communication Security Laboratory, Chengdu, China
Corresponding author: lai-xj@cs.sjtu.edu.cn

***Abstract*—The CAESAR competition is launched in 2013 which aims to find some authenticated encryption with good security and performance. Among these submissions, LAC is designed in a unique way with leaked-state structure. In this paper, based on birthday paradox,we find a forgery attack on LAC in nonce-misused case with time complexity 2^{28}. Moreover, we generalize the attack on normal version of leaked-state authenticated encryption and conclude some suggestions on how to use such structure.**

Keywords—CASESAR; LAC; birthday paradox; leaked-State authenticated encryption; forgery attack

I. INTRODUCTION

Authenticated encryption (AE) [1] schemes are key-based cryptography algorithms which provide both confidentiality and authenticity, and both goals are fundamental goals in cryptography. In later researches, authenticated encryption with associated data (AEAD) [2] and nonce-based authenticated encryption [3] are proposed to satisfy the real-world cryptography. Some modes of operations on block cipher are also designed to combine confidentiality and authenticity together, such as CCM [4], OCB [5] and so on. Among those designs, AES-GCM [6] is most widely accepted for the reason of its good security. However, due to its poor performance, many projects and works use some designs with worse security and faster speed instead of AES-GCM.

In 2013, the CAESAR competition (Competition for Authenticated Encryption: Security, Applicability, and Robustness) [7] is announced for finding better AE than AES-GCM. Many optional features can be considered in the competition design. Besides the security, designer may consider whether AE can be progressed on-line, whether AE only need one path of encryption and decryption and the inverse is free, whether AE allow nonce-misused or decryption-misused, whether AE can increase associated data and whether AE provide an immediate tag. It's a hard problem to get a balance between security, speed and those features. 57 submissions are collected around the world with different ideas and structures. Most of them are based on block cipher, some also give ideas on how to use stream cipher, hash function or other primitives to construct AE. Combined with different ideas of operation modes, researchers will discuss and vote for some of candidates for second round competition.

LAC [8] is one of the submissions which uses a unique structure similar to ALE [9]. We call such structure leaked-state structure because it leaks some inner states and combines them with parts of round states as a leaked-state which xor with message as the ciphertext. LBlock [10] is the primitive of LAC. LAC has many properties, such as inverse-free, on-line, good security if the nonce is not reused. The most attractive features are that it's extremely lightweight, flexible and compatible.

The remainder of this paper is organized as follow. The specification of LAC is given in Section II. Section III and IV give the forgery attack on LAC and generalized leaked-state AE. Some discussion and suggestions are proposed in Section V. Section VI concludes this paper.

II. SPECIFICATION OF LAC

A. Procedure of LAC

LAC uses LBlock as its primitive and is based on the leaked-state structure which is shown in Fig.1. Its encryption accepts an 80-bit master key K, a 64-bit public message number(PMN) as a nonce, message m and an associated data a. Then it outputs the ciphertext c and a 64-bit authentication tag t. Its decryption accepts an 80-bit master key K, a 64-bit public message number(PMN) as a nonce, ciphertext c, an associated data a and a 64-bit tag t. If the tag is correct, it output the message m, otherwise it outputs a terminated mark $\perp$.

Padding pads length and 0 after message whose whole length is a multiple of 48. By initialling the PMN by master key K twice, the higher 80-bit state is used in initialling state and the lower 80-bit state becomes the master key of key schedule. Then it processes associated data with simplified LBlock, called LBlock-s, and processes message with its leaked version.

Using LBlock-s instead of LBlock is for the sake of speed. LBlock is a variant of Feistel Network of 32 rounds and its round function is:

$$X_i = P(S(X_{i-1} \oplus K_{i-1})) \oplus (X_{i-2} <<< 8), i = 2,3,...,33 \quad (1)$$

LBlock-s is a 16 rounds version of LBlock and replaces different S-boxes of LBlock with the same 4-bit S-box. LBlock-s outputs $X_{17}||X_{16}$. Leaked LBlock-s outputs $X_{17}||X_{16}$ and a leaked state $X_9^*||X_{17}^*$, denoted as:

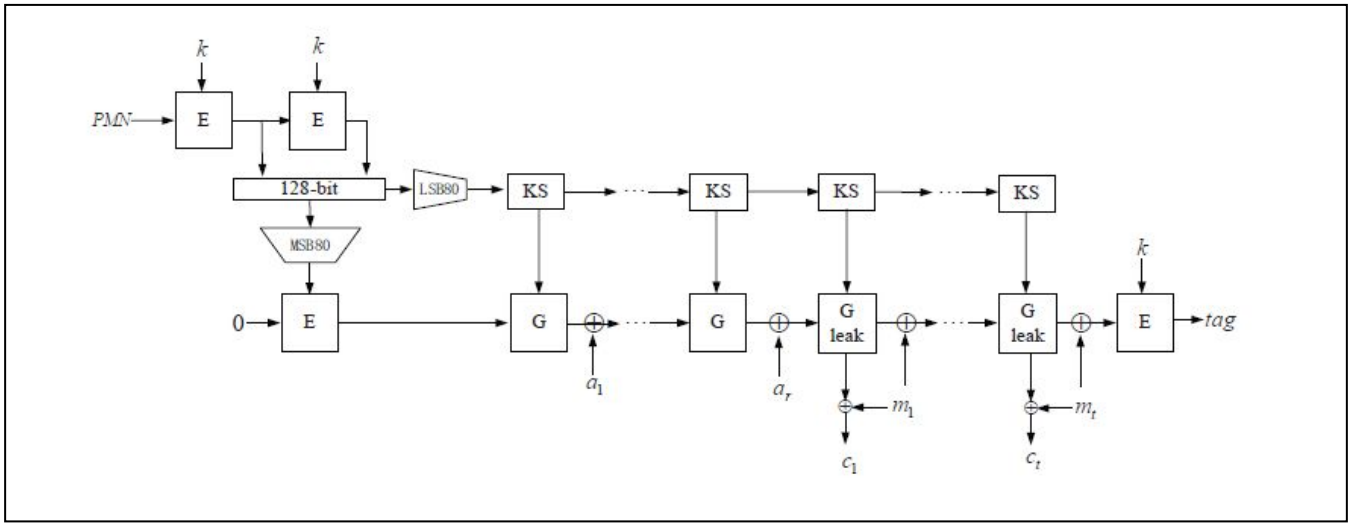

Fig.1. The encryption/authentication operation of LAC [8]

$$X_9^* = X_9[31,30,...,8].$$
$$X_{17}^* = X_{17}[31,30,...,8]. \quad (2)$$

The ciphertext is produced by the leaked state xor message. The lower 48-bit output of LBlock-s and leaked LBlock-s is xor with associated data and message as the next block's round state input.

At the final of encryption, the state is encrypted by master key K with LBlock and generates the authentication tag t. Decryption is similar with encryption, except that the message is produced by the leaked state xor the ciphertext.

Leaked-state AE has the inverse-free feature, it doesn't need decryption operation of primitives. Thus hash round function can be replaced in such structure, if proper state can be leaked by leaked round function.

B. Security Claim of LAC

The security claim of confidentiality of the plaintext is no less than 2^{80}.

The security claim of integrity for the plaintext, associated data and PMN is that any forgery attack with an unused tuple (PMN, a, c, t) has a success probability at most 2^{-64}.

All of the above security claim is under one time nonce case, that is to say, PMN cannot be used. In next Section, we will show how the security drops a lot in nonce-misused case, and we will conclude the suggestions against such attack.

Here we define the integrity to generate a forgery attack the same as INT-PTXT [1]. We can query the encryption oracle in specified q queries and t' times and if we can give an unused tuple (PMN, a, c, t), then we say we successfully launch a forgery attack.

III. A Forgery Attack on LAC

We generate the forgery attack based on birthday attack [11]. Considering the block state xor message, it's the lower 48-bit, meanwhile the leaked state of higher 24-bit round state also has been proceeded with message. Thus there are 8-bit of the round state can be controlled by the attacker.

Here is the attack:

(a) Query 2^{28} message M_1M_2, which is independently uniformly random chosen, then we can get 2^{28} ciphertext C_1C_2.

(b) According to M_1M_2 and C_1C_2, modify M_2C_2 which makes the 8 duplicate bits of second block state with the same value.

(c) By birthday paradox, the left 56-bit may have a collision with high probability, but due to that the round state isn't outputted, we cannot directly know which are the collision.

(d) Here is the trick, we query those M_1M_2 with the same $M_3M_4M_5$, and get 2^{28} ciphertext $C_3C_4C_5$. We just judge the collision by whether there are the same $C_3C_4C_5$. The reason we choose 3 more blocks is as the following: By each pair of MC, we can recover 24-bit of the block state. Remember that we consider the nonce-misused case, the key schedule will be the same between every query. The probability of different block states with the same 24-bit is 2^{-24}, then the probability of different block states with the same 24-bit on 3 rounds is 2^{-72}, which is less than 2^{-64}. Thus we can say that the round state is the same, and choose that pair as a collision.

(e) Now we get an inner collision, and the subkey is the same, we can randomly add the same message after the pair, use one of them to be authenticated, then the authentication tag can be verified for the other one. Here we get a forgery attack.

Now we need to consider the time complexity and the space complexity. The time complexity is very simple to calculate: 2^{28}. The space complexity is a bit complicated. There are 2^{28} queries, each query need to store 2 blocks of message and 5 blocks of ciphertext, message and the ciphertext is 48-bit. Thus the space complexity is:

$$2^{28} \times 48 \times 7 = 2^{36.39} \text{ bits.}$$

It's a practical attack and it's below the birthday bound. The inner state of the leaked AE is very similar to the inner state of hash function, therefore we can use the birthday bound as the bound of nonce-misused case. In section V, we will discuss whether it's possible to exceed the birthday bound.

IV. A FORGERY ATTACK ON LEAKED-STATE AE

Consider the forgery attack in Section III, we actually view LBlock-s as a black box and then we start our attack. Thus the forgery attack can be generalized to normal case of leaked-state AE.

Now we consider the following case:

s: the bits of leaked state duplicated with the bits of block state which are both operated with message, which is to say, the state can be recovered and be modified easily.

m: the length of message processed in each block operation.

s': the bits of leaked state on the block state output.

l : the length of the inner state.

The time complexity is the number of the query. Since there are s bits duplicated, we only need to find the collision on the left *l*-*s* bits. Thus the time complexity is:

$$2^{\frac{l-s}{2}}. \tag{3}$$

Then we have to calculate the space complexity. For each query, we need to store several plaintext and ciphertext. In the collision part, we need to store both of them. The number of the blocks is determined by the bits we need to find collision. Normally it's recommended to be twice longer.

In the verification of collision, we only need to store the ciphertext for the sake of checking the pair. The number of the blocks is determined on the length of longest leaked inner state, for example, in LAC, both of the leaked inner state is 24 bits, therefore the longest leaked inner state is 24 bits.

Combine both finding collision and verifying collision, the space complexity is:

$$2^{\frac{l-s}{2}} \times m \times \left(\left\lceil \frac{2 \times (l-s)}{l} \right\rceil \times 2 + \left\lceil \frac{l}{s'} \right\rceil \right) \text{bits}. \tag{4}$$

V. DISCUSSION ON LEAKED-STATE AE

A. Nonce-misused Case

In real-world cryptography, it's very common that programmers fix one nonce as a constant. We have to consider that AE schemes should not loss too much security in this situation. In the case of LAC, if the nonce is reused, then the attacker can find a forgery in 2^{28}. Although LAC performs good in one-time nonce case, the security in nonce-misused case performs too poor. For general leaked-state AE, we suggest that $2^{\frac{l-s}{2}}$ is a in a safe bound. This means that the *l* of leaked-state AE at least 128 bits.

Meanwhile, we suggest the *s* and *s'* of leaked-state should be small. The former one directly decreases the time complexity of the attack. The later one means the round state into next round can be detected for some bits, it may cause security trouble. Take LAC as an example, if the leaked-state consists of 2 parts, then the leaked-state on block state should be third of whole length. Meanwhile it's perferred that LBlock-s uses 24 rounds and leaks the 8-th,16-th round, which may improve the security in nonce-misused case.

B. Birthday Bound

It seems to be a question whether the security in nonce-misused case can exceed the birthday bound. In the CAESAR competition, some similar candidates exceed the birthday bound, such as SHELL. However they use more complex structures to achieve this goal.

Consider there's a leaked state AE, besides the key schedule and leaked ciphertext, it is very similar to a normal hash function. If the nonce is reused, then the key is the same and we can ignore the effect on key schedule. The leak of the inner state may leak more information to the attacker. So we believe it's hard to exceed the birthday bound with leaked-state AE.

C. Advantage of Leaked-state AE

The first advantage is the use of nonce. If we change the nonce every time, the subkey will all be changed. This feature seems to be one-time-pad. We believe the security in one-time nonce case is one-time-pad if the primitives is CCA secure and the key schedule doesn't generate weak keys.

Using nonce in key schedule is a good idea, however many other candidates also have similar use of nonce. The most different feature of leaked-state AE is the light-weighted. Because the encryption is just in one line, and the round of primitive can be slightly decreased, if the primitive is light-weight enough, then the AE scheme can be fast and flexible. We suggest some 128-bit light-weighted cipher and hash function may be good choice for the primitive.

VI. CONCLUSION

In this paper, we construct a forgery attack on LAC with 2^{28} time complexity and $2^{36.39}$ bits space complexity in nonce-misused case. We also generalize the forgery attack to leaked-state AE with $2^{\frac{l-s}{2}}$ time complexity. According to the attack and analysis, we also propose some ideas to use such structure quickly and safely.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (61272440, 61472251), China Postdoctoral Science Foundation (2013M531174, 2014T70417), and Science and Technology on Communication Security Laboratory.

REFERENCES

[1] M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In ASIACRYPT, volume 1976 of Lecture Notes in Computer Science, pp. 531–545. Springer, 2000.

[2] P. Rogaway. Authenticated-Encryption with Associated-Data. In ACM Conference on Computer and Communications Security, pp. 98–107, 2002.

[3] P. Rogaway. Nonce-Based Symmetric Encryption. In FSE, pp. 348–359, 2004.

[4] D. Whiting, R. Housley, N. Ferguson,:AES Encryption and Authentication Using CTR Mode and CBC-MAC. IEEE 802.11-02/001r2 (2002)

[5] P. Rogaway, M. Bellare, J. Black, T. Krovetz: OCB: a block-cipher mode of operation for efficient authenticated encryption. In ACM Conference on Computer and Communications Security, pp. 196–205. ACM (2001)

[6] D. McGrew and J. Viega. The Galois/Counter Mode of Operation (GCM). Submission to NIST. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf, 2004.

[7] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, 2014, http://competitions.cr.yp.to/caesar.html

[8] L. Zhang, W. Wu, Y. Wang, S. Wu, and J. Zhang. Lac: A lightweight authenticated encryption cipher. http://competitions.cr.yp.to/round1/lacv1.pdf, 2014.

[9] A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, E. Tischhauser: ALE: AES-Based Lightweight Authenticated Encryption. Accepted by FSE 2013. http://www2.compute.dtu.dk/~anbog/fse13-ale.pdf (2013)

[10] W. Wu, L. Zhang: LBlock: A Lightweight Block Cipher. ACNS 2011. LNCS, vol. 6715,pp. 327-344. Springer, (2011)

[11] M. Bellare and T. Kohno. Hash function balance and its impact on birthday attacks. Advances in Cryptology – EUROCRYPT ’04,