Research on State Collisions of Authenticated Cipher ACORN

Pei Zhang[a*], Jie Guan[b], Junzhi Li[c] and Tairong Shi[d]

Information Science and Technology Institute, Zheng Zhou, China

[a*]zhangpeilegend@163.com, [b]guanjie007@163.com, [c]lijunzhi1998@163.com, [d]strwanzi@163.com

Keywords: CAESAR, ACORN, State Collision, Message Authentication.

Abstract. This paper analyzes the authentication security of lightweight authenticated cipher ACORN, a second-round candidate of the CAESAR competition. We first put forward the necessary and sufficient conditions for state collisions of ACORN. Then we point out the errors of the designer's analysis and utilize a reverse derivation method to find proper input differences to the registers which can ensure that the state differences get eliminated in certain rounds. Last but not least, we redefine the differential properties of the overall feedback function to compute the probability of state collisions more precisely and improve the security margin of ACORN given by the designer.

Introduction

ACORN[1] is a binary feedback register based authenticated stream cipher designed by H. Wu and has been selected as a second-round candidate of CAESAR[2] (Competition for Authenticated Encryption: Security, Applicability, and Robustness). The designer specifically prohibits the nonce-reuse and decryption-misuse settings in the first version ACORN v1. To provide protection against nonce-reuse in encryption/decryption a tweaked version named ACORN v2 was submitted in September 2015.

Both versions of ACORN use a 128-bit key and a 128-bit IV (initialization vector) to protect up to 2^{64} bits of associated data and up to 2^{64} bits of plaintext. The designer strongly recommends the use of a 128-bit authentication tag. ACORN contains a 293-bit state consisting of six concatenated LFSRs and the use of bit-oriented registers provides a new approach to design lightweight authenticated ciphers.

Up to now there are some published analyses of ACORN v1, but none of them threatens its security. Liu et al.[3] describe slid properties of keys and IVs for ACORN v1, and propose state recovering attacks using guess-and-determine and differential-algebraic techniques, although the attacks are worse than exhaustive key search. Chaigneau et al.[4] appliy a key recovery attack to ACORN v1 in nonce-reuse and decryption-misuse settings, which is against the security claim of the designer. Salam et al.[5] develop a known-key collision attack on ACORN v1, which is not a feasible attack clearly. Zhang et al.[6] estimate the security margin of ACORN v1 by finding the linear approximations of the output function.

This paper mainly focuses on the message authentication security of ACORN v2. We point out and correct the errors of the designer's analysis, presenting a more precise security bound of authentication security by constructing state collisions. For the rest of the paper, we use ACORN to refer the second-round submission ACORN v2.

Description of ACORN

This section gives a brief description of ACORN by defining the functions used in the cipher and introducing different phases. ACORN contains a 293-bit internal state denoted by $S_i = (S_{i,0}, S_{i,1}, \mathrm{L}\ ,\ S_{i,292})$ for the *i*th step. The state consists of six concatenated LFSRs, as shown in Fig. 1.

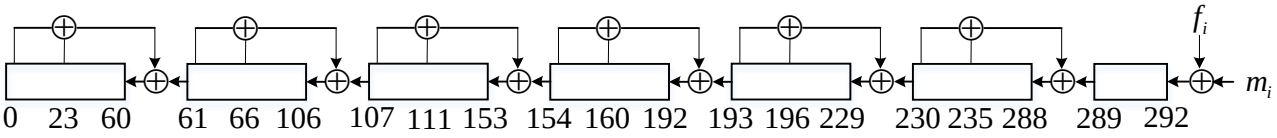

Figure 1. Internal state of ACORN

Functions of ACORN

Keystream generation function. At the *i*th step the keystream bit is computed using the function ks_i=*KSG*128(S_i):

$$ks_i = S_{i,12} \oplus S_{i,154} \oplus maj(S_{i,235}, S_{i,61}, S_{i,193}) \tag{1}$$

where the *maj* function is defined as:

$$maj(x_1, x_2, x_3) = x_1x_2 \oplus x_2x_3 \oplus x_1x_3 \tag{2}$$

Feedback Function. At the *i*th step the keystream bit is computed using the function f_i=*FBK*128(S_i, ca_i, cb_i):

$$f_i = S_{i,0} \oplus \overline{S_{i,107}} \oplus maj(S_{i,244}, S_{i,23}, S_{i,160}) \oplus ch(S_{i,230}, S_{i,111}, S_{i,66}) \oplus ca_i S_{i,196} \oplus cb_i ks_i \tag{3}$$

where $\overline{S_{i,107}}$ denotes the complement value of $S_{i,107}$. In Eq. 3, the *ch* function is defined as:

$$ch(x_1, x_2, x_3) = x_1x_2 \oplus \overline{x_1}x_3 \tag{4}$$

where $\overline{x_1}$ denotes the complement value of x_1.

State Update Function. At the *i*th step the pseudo code for the state update function S_{i+1}=StateUpdate128(S_i, m_i, ca_i, cb_i) is as follows:

$S_{i,289} = S_{i,289} \oplus S_{i,235} \oplus S_{i,230}$;

$S_{i,230} = S_{i,230} \oplus S_{i,196} \oplus S_{i,193}$;

$S_{i,193} = S_{i,193} \oplus S_{i,160} \oplus S_{i,154}$;

$S_{i,154} = S_{i,154} \oplus S_{i,111} \oplus S_{i,107}$;

$S_{i,107} = S_{i,107} \oplus S_{i,66} \oplus S_{i,61}$;

$S_{i,61} = S_{i,61} \oplus S_{i,23} \oplus S_{i,0}$;

$f_i = FBK128(S_i, ca_i, cb_i)$;

for *j*=0 to 291 do $S_{i+1,j} = S_{i,j+1}$;

$S_{i+1,292} = f_i \oplus m_i$.

The Initialization

The initialization phase consists of injecting the key $K = (k_0, \ldots, k_{127})$ and $IV = (iv_0, \ldots, iv_{127})$ into the state and updating the state for 1792 steps. The pseudo code is given below.

1. Let $S_{-1792} = (0, \ldots, 0)$
2. Let $m_{-1792+i} = k_i$ for *i*=0 to 127.
   Let $m_{-1792+128+i} = iv_i$ for *i*=0 to 127.
   Let $m_{-1792+256} = k_0 \oplus 1$.
   Let $m_{-1792+256+i} = k_{i \bmod 128}$ for *i*=1 to 1535.
3. Let $ca_{-1792+i} = 1$ for *i*=0 to 1791.
   Let $cb_{-1792+i} = 1$ for *i*=0 to 1791.
4. For *i*=-1792 to -1, $S_{i+1} = \text{StateUpdate128}(S_i, m_i, ca_i, cb_i)$.

Processing the Associated Data

After the initialization, the associated data is used to update the state. Denote by l_{ad} the bit length of associated data and $AD = (ad_0, \mathrm{L}\ , ad_{l_{ad}-1})$ the associated data.

1. Let $m_i = ad_i$ for i=0 to l_{ad} -1.
   Let $m_{l_{ad}} = 1$
   Let $m_{l_{ad}+i} = 0$ for i=1 to 255.
2. Let $ca_i = 1$ for i=0 to l_{ad} +127.
   Let $ca_i = 0$ for i=l_{ad} +128 to l_{ad} +255.
   Let $cb_i = 1$ for i=0 to l_{ad} +255.
3. For i=0 to l_{ad} +255, $S_{i+1} = \mathrm{StateUpdate128}(S_i,\ m_i,\ ca_i,\ cb_i)$.

The Encryption

After processing the associated data, one plaintext bit p_i is used to update the state at the ith step and p_i is encrypted to get c_i. Denote by l_p the bit length of plaintext and $P = (p_0, \mathrm{L}\ , p_{l_p-1})$ the plaintext.

1. Let $m_{l_{ad}+256+i} = p_i$ for i=0 to l_p -1.
   Let $m_{l_{ad}+256+l_p} = 1$
   Let $m_{l_{ad}+256+l_p+i} = 0$ for i=1 to 255.
2. Let $ca_i = 1$ for i=l_{ad} +256 to $l_{ad} + l_p$ +383.
   Let $ca_i = 0$ for i=$l_{ad} + l_p$ +384 to $l_{ad} + l_p$ +511.
   Let $cb_i = 0$ for i=l_{ad} +256 to $l_{ad} + l_p$ +511.
3. For i=l_{ad} +256 to $l_{ad} + l_p$ +511,
   $S_{i+1} = \mathrm{StateUpdate128}(S_i,\ m_i,\ ca_i,\ cb_i)$;
   $c_i = p_i \oplus KSG128(S_i)$.
   End for

The Finalization

After processing the plaintext, the authentication tag T is generated.

1. Let $m_{l_{ad}+l_p+512+i} = 0$ for i=0 to 767.
2. Let $ca_i = 1$ for i=$l_{ad} + l_p$ +512 to $l_{ad} + l_p$ +1279.
   Let $cb_i = 1$ for i=$l_{ad} + l_p$ +512 to $l_{ad} + l_p$ +1279.
3. For i=$l_{ad} + l_p$ +512 to $l_{ad} + l_p$ +1279,
   $S_{i+1} = \mathrm{StateUpdate128}(S_i,\ m_i,\ ca_i,\ cb_i)$;
   $ks_i = KSG128(S_i)$.
   End for

The authentication tag T is the last t keystream bits, i.e.,
$T = ks_{l_{ad}+l_p+1279-t+1} \,||\, ks_{l_{ad}+l_p+1279-t+2} \,||\, \mathrm{L}\ \,||\, ks_{l_{ad}+l_p+1279}$.

Note that the designer strongly recommends the use of a 128-bit authentication tag.

Construction of State Collisions in ACORN

In this section we try to construct state collisions in ACORN. We first study the properties of state update function and put forward the necessary and sufficient conditions for state collisions of ACORN. Based on this we utilize a reverse derivation method to find proper input differences to the registers which can ensure that the differences in the state get eliminated in certain rounds.

Theorem. Let d_i be the input feedback bit to the concatenated LFSRs, i.e. $d_i = f_i \oplus m_i$. For two different states of ACORN S_i and S_i', the necessary and sufficient conditions of $S_{i+1} = S_{i+1}'$ are as follows:

1. The two states S_i and S_i' *only* differ at seven stages: 0, 61, 107, 154, 193, 230 and 289, while the other stages stay equivalent.

2. $d_i = d_i'$.

Proof. *Sufficiency*. We take the first linear recurrence of the state update function as an example.

$$S_{i,289} = S_{i,289} \oplus S_{i,235} \oplus S_{i,230} \tag{5}$$

Eq. 5 shows that if S_i and S_i' differ at stage 289 and stage 230, while stay equivalent at stage 235, the values of the updated stage 289 for two states are equivalent. Similarly, we can obtain that the values of the other six particular register stages are also equivalent after the update. With equal input feedback bits, the two states are certain to collide after one round iteration since the shifting operation doesn't change the values of stages. In this way the sufficiency of the conditions is proved.

Necessity. If $S_i \neq S_i'$ and $S_{i+1} = S_{i+1}'$, we can obtain the following equations due to the shifting properties of the state update function:

$$\begin{cases} S_{i,j} = S_{i,j}' (0 \le i \le 292, i \ne 0, 61, 107, 154, 193, 230, 289) \\ S_{i,0} \oplus S_{i,61} = S_{i,0}' \oplus S_{i,61}' \\ S_{i,61} \oplus S_{i,107} = S_{i,61}' \oplus S_{i,107}' \\ S_{i,107} \oplus S_{i,154} = S_{i,107}' \oplus S_{i,154}' \\ S_{i,154} \oplus S_{i,193} = S_{i,154}' \oplus S_{i,193}' \\ S_{i,193} \oplus S_{i,230} = S_{i,193}' \oplus S_{i,230}' \\ S_{i,230} \oplus S_{i,289} = S_{i,230}' \oplus S_{i,289}' \\ d_i = d_i' \end{cases} \tag{6}$$

From Eq. 6 we get that at least one of the above seven particular stages for the two states has to be different in order to satisfy the condition $S_i \neq S_i'$. However, if one of the seven stages is different, the other six stages also have to be different to ensure $S_{i+1} = S_{i+1}'$. Namely, the conditions are satisfied and the necessity is proved.

□

To attack ACORN authentication the designer injects a difference into the state and manages to eliminate the differences in the register stages by selecting proper input differences. As the designer has explained, the analysis of the difference in associated data and ciphertext are similar, so we regard the feedback bit and the messages bit as a whole. However, the designer makes a slip in writing one of the linear recurrences, leading to an invalid difference which cannot eliminate the state differences. Different from the designer's analysis, we utilize a reverse derivation method to find proper input differences to the registers which can ensure that the differences in the state get eliminated after certain rounds.

Assuming $\Delta S_0 = (0, 0, \mathrm{L}\ , 0)$ and $\Delta d_0 = 1$, and we obtain a state collision $\Delta S_t = (0, 0, \mathrm{L}\ , 0)$ after t rounds of iteration for the first time with input differences $\Delta d_1, \Delta d_2, \mathrm{L}\ , \Delta d_{t-1}$. Then we use a reverse

derivation method to determine the input differences of each round Δd_i. According to the theorem, the state differences at time *t*-1 are given below:

$$\Delta S_{t-1,j} = \begin{cases} 1 & for\ j = 0,61,107,154,193,230,289 \\ 0 & otherwise \end{cases} \tag{7}$$

Besides, we can also get the input difference at time *t*-1 $\Delta d_{t-1} = \Delta S_{t,292} = 0$. Next we use the inverse function of the state update function to work out the state differences and the input difference of each round. Given ΔS_{i+1}, the internal state differences of ACORN at time *i* are defined as:

$$\Delta S_{i,j} = \begin{cases} 0 & for\ j = 0 \\ \Delta S_{i+1,22} \oplus \Delta S_{i+1,60} & for\ j = 61 \\ \Delta S_{i,61} \oplus \Delta S_{i+1,65} \oplus \Delta S_{i+1,106} & for\ j = 107 \\ \Delta S_{i,107} \oplus \Delta S_{i+1,110} \oplus \Delta S_{i+1,153} & for\ j = 154 \\ \Delta S_{i,154} \oplus \Delta S_{i+1,159} \oplus \Delta S_{i+1,192} & for\ j = 193 \\ \Delta S_{i,193} \oplus \Delta S_{i+1,195} \oplus \Delta S_{i+1,229} & for\ j = 230 \\ \Delta S_{i,230} \oplus \Delta S_{i+1,234} \oplus \Delta S_{i+1,288} & for\ j = 289 \\ \Delta S_{i+1,j-1} & otherwise \end{cases} \tag{8}$$

Hence we can get $\Delta d_i = \Delta S_{i+1,292}$ directly. Note that we define $\Delta S_{i,j} = 0$ for each inverse iteration in order to find the shortest difference. Through program, the shortest input difference (290 bits) is given below:

1000 0000 0000 0000 0000 0000 0000 0000 0110 0111 0101 0011 0000 0010 0001 0100
0001 0011 1110 0010 0011 1011 0101 0110 1000 1100 1011 1010 1100 1011 1010 0010
1101 1111 0001 0111 1101 1111 0100 1011 1001 1100 1000 1110 1110 0100 1110 1010
0010 0100 1110 0110 0111 1111 1111 1100 1000 0101 1101 0000 0011 0010 1100 0000
0101 1011 0000 0110 1101 0000 1011 0110 01

Our experiment shows that the difference given by the designer is wrong, which cannot eliminate the internal differences. We also verify that the above input difference can ensure that the states are definitely to collide after 293 rounds of iteration.

Next we have to introduce proper differences by modifying the ciphertext or associated data in order to get the above input difference to the registers. Same with the designer, we analyze the differential properties of the feedback function. Nevertheless, we find that the differential property of *ch*(*x*, *y*, *z*) described by the designer is incorrect. Besides, the designer analyzes the two nonlinear functions separately, instead of putting them together and resolving differential properties of the overall feedback function, which is also inaccurate. We redefine the differential properties of *maj*(*x*, *y*, *z*) and *ch*(*x*, *y*, *z*), as shown in Tab. 1.

Table 1. The differential properties of *maj*(*x*, *y*, *z*) and *ch*(*x*, *y*, *z*)

Function	Input difference	Output difference	Probability
maj(*x*, *y*, *z*)	(0,0,0)	0	1
	(1,1,1)	1	1
	otherwise	0/1	0.5
ch(*x*, *y*, *z*)	(0,0,0)	0	1
	(0,1,1)	1	1
	otherwise	0/1	0.5

The nonlinear part of the feedback function consists of two *maj* functions and one *ch* function and all the nine inputs are different, so the variables are independent and uniformly distributed. In that case we can directly get the differential properties of the feedback function. Without loss of generality, we describe the nonlinear part of the feedback function as follows:

$$g(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9) = maj(x_1, x_2, x_3) + maj(x_4, x_5, x_6) + ch(x_7, x_8, x_9) \tag{9}$$

From Table 1 and Eq. 9 we can get that there are eight input differences whose corresponding output differences can be determined with probability 1, while for other input differences the output differences are 0/1 with probability 0.5. With this property and the above input differences to the LFSRs, we program to get that the probability of eliminating the state differences is 2^{-181}, which is better than the security margin 2^{-189} given by the designer.

Conclusions

ACORN is a lightweight authenticated stream cipher which has been selected as a second-round candidate of CAESAR. The published analyses of ACORN haven't threatened its security up to now. This paper mainly focuses on the message authentication security of ACORN by constructing state collisions. We point out and correct the errors of the designer's analysis, improving the probability of state collisions from 2^{-189} to 2^{-181}. Note that our results show that ACORN is able to provide 128-bit MAC security and how to attack ACORN effectively is still an open problem for our future work.

Acknowledgements

This work was supported by Natural Science Foundation of China under Grant 61572516.

References

[1] H. Wu, ACORN: A Lightweight Authenticated Cipher (v2). http://competitions.cr.yp.to/caesar-submissions.html.

[2] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. Available from: http://competitions.cr.yp.to/index.html.

[3] M. Liu and D. Lin, Cryptanalysis of Lightweight Authenticated Cipher ACORN. Cryptographic Competitions Mailing List. https://groups.google.com/forum/#!topic/crypto-competitions.

[4] C. haigneau, T. Fuhr, and H. Gilbert, Full Key-Recovery on ACORN in Nonce-Reuse and Decryption-Misuse Settings. Cryptographic Competitions Mailing List. https://groups.google. com/forum/ #!topic/crypto-cmpetitions.

[5] M. I. Salam, K. K. Wong, H. Bartlett, L. Simpson, E. Dawson, and J. Pieprzyk, Finding State Collisions in the Authenticated Encryption Stream Cipher ACORN. Cryptology ePrint Archive, Report 2015/918 (2015). http://eprint.iacr.org/.

[6] L. Jiao, B. Zhang and M. Wang, in: Two Generic Methods of Analyzing Stream Ciphers. edited by J. Lopez and C. J. Mitchell, ISC 2015, LNCS 9290, pp. 379-396, 2015.

[7] B. Zhang, Z. Li. D. Feng, D. Lin: Near collision attack on the grain v1 stream cipher. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 518–538. Springer, Heidelberg (2014).

[8] M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. Journal of Cryptology, 21(4):469–491, 2008.

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

[10] N. Ferguson and B. Schneier, A cryptographic evaluation of IPsec. Counterpane Internet Security, Inc, 3031, 2000.

[11] G. Leurent, Differential Forgery Attack against LAC. HAL Id: hal-01017048, https://hal.-inria.fr/hal-01017048, 2014.

[12] T. Fuhr, G. Leurent and V. Suder, Forgery and Key-Recovery Attacks on CAESAR Candidate Marble. HAL Id: hal-01102031, https://hal.- inria.fr/hal- 01102031, 2015.

[13] J. Lu, On the Security of the COPA and Marble Authenticated Encryption Algorithms against (Almost) Universal Forgery Attack. Cryptology ePrint Archive, Report 2015/079 (2015). http://eprint.iacr.org/.

[14] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schlaffer, Cryptanalysis of Ascon. Cryptology ePrint Archive, Report 2015/030 (2015). http://eprint.iacr.org/.

[15] T. Peyrin, S. M. Sim, L. Wang, and G. Zhang, Cryptanalysis of JAMBU. Cryptology ePrint Archive, Report 2014/931 (2014). http://eprint.iacr.org/.