Fault Attack on ACORN v3

Xiaojuan Zhang[1,2], Xiutao Feng[1,3], Dongdai Lin[1,]

[1]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Key Laboratory of Mathematics Mechanization, Academy of Mathematics and System Science, Chinese Academy of Sciences, Beijing, China
zhangxiaojuan@iie.ac.cn

Abstract. Fault attack is one of the most efficient side channel attacks and has attracted much attention in recent public cryptographic literatures. In this work we introduce a fault attack on the authenticated cipher ACORN v3. Our attack is done under the assumption that a fault is injected into an initial state of ACORN v3 randomly, and contains two main steps: fault locating and equation solving. At the first step, we introduce concepts of unique set and non-unique set, where differential strings belonging to unique sets can determine the fault location uniquely. For strings belonging to non-unique sets, we use some strategies to increase the probability of determining the fault location uniquely to almost 1. At the second step, we demonstrate several ways of retrieving equations, and then obtain the initial state by solving equations with the guess-and-determine method. With n fault experiments, we can recover the initial state with time complexity $c \cdot 2^{146.5-3.52 \cdot n}$, where c is the time complexity of solving linear equations and $26 < n < 43$. We also apply the attack to ACORN v2, which shows that, comparing with ACORN v2, the tweaked version ACORN v3 is more vulnerable against the fault attack.

Keywords: CAESAR, Authenticated Cipher, Stream Cipher, ACORN, Fault Attack

1 Introduction

The CAESAR competition [1], launched in 2014, aims to find authenticated ciphers that offer advantages over AES-GCM and are suitable for widespread adoption. Totally, 57 candidates have been submitted to the competition. After two rounds of assessment, only 16 survivors were announced to be included in the third round. ACORN submitted by Hongjun Wu is one of the 16 proposals. It contains three versions [2,3,4], and is based on a simple binary feedback shift register (FSR, for short) of length 293. The third round submission ACORN v3 is different from ACORN v2 in the feedback function and the filter function.

Up to now, there are some cryptographic analyses on ACORN that provide some insights into the diffusion ability of the cipher. Using the guess-and-determine and the differential-algebraic techniques, Liu et al. proposed a state recovering attack on ACORN v1 [5]. But the attack is worse than a brute force

attack. Chaigneau et al. showed a key recovery attack on ACORN v1 when nonce is reused to encrypt a small amount of chosen plaintexts [6]. It is shown that if one IV is reused seven times, the security of ACORN is lost. Salam et al. developed cube attacks to the reduced round version of ACORN v1 and v2 [7], which are far from threatening the real-life usage of the cipher. Salam et al. investigated an attack to find a collision of the state under the assumption that the key is known [8]. Lafitte et al. described that they develop practical attacks to recover the state and the key [9]. However, the attacks are much more expensive than the brute force attack. Josh et al. claimed that the associate data do not affect any keystream bits if the size of the associated data is small [10]. Dibyendu et al. gave some results on ACORN [12], one of them is that they find a probabilistic linear relation between plaintext bits and ciphertext bits, which hold with probability $\frac{1}{2} + \frac{1}{2^{350}}$. But the bias is too small to be tested. Another result is that they could recover the initial state of the cipher with time complexity approximately equalling to 2^{40}, which is done under an impractical assumption. The designer of ACORN gave the comments on these analysis in [13], which show that some of the attacks are not really attacks.

Since fault differential attack is one of side channel attacks working on physical implementations, it is interesting to apply side channel cryptanalysis to a cryptographic algorithm that is being used or will be used in reality. Due to the work of Biham et al.[14], fault attack becomes a powerful tool to retrieve the secret key of many cryptographic primitives. The first fault attack on stream cipher was introduced by Hoch and Shamir [15]. They showed that a typical fault attack allows an attacker to inject faults by means of laser shots/clock glitches[16,17] into a device initialized by a secret key and change one or more bits of its internal state. Then by analyzing the difference between the faulty device and the right device, he or she could deduce some information about the internal state or secret key. Under the assumption that a hard fault is injected at a certain position, Dey et al. proposed a hard fault attack on ACORN v1 and v2 in a nonce-respecting scenario in [18]. There are not any results of differential fault attacks on ACORN v3 under a general fault model of random fault injection.

In this work we give some results of fault differential attacks on ACORN v3 under a general fault model of random fault injection into a initial state. Our attack contains two main steps: fault locating and equation solving. At the fault locating step, we show that when a fault is injected into a initial state randomly, we can get a differential string between the error and correct keystream bits. With the differential string, we aims to identify the fault location. First, for each fault location, we give a method to obtain the differential set correspondingly, which contains all possible differential strings. And then sort the sets into two parts: unique set and non-unique set. We say that a differential set is a unique set if all the strings belonging to it can determine the fault locations uniquely. Otherwise, we call it non-unique set. If the differential string belongs to non-unique sets, we use some strategies including the keystream bits extension strategy and the high probability priority strategy, to increase the

probability of determining the fault location uniquely. We show that when 163-bit keystream is available, the probability can reach to 99.998%. At the second step, we first give two algorithms to retrieve equations. Our main idea is based on the observation that the first 99-bit differential keystream of ACORN v3 can be expressed as linear or quadratic functions with respect to the initial state, which helps us to recover the initial state. We also give several methods to retrieve more linear equations. Then we use the guess-and-determine method to obtain the initial state. With n fault experiments, we can recover the initial state with time complexity $c \cdot 2^{146.5-3.52 \cdot n}$, where c is the time complexity of solving linear equations and $26 < n < 43$. We also apply the differential fault attack to ACORN v2. The initial state of ACORN v2 can be recovered with time complexity $c \cdot 2^{146.5-1.95 \cdot n}$, where $40 < n < 77$. The results show that comparing with ACORN v2, the tweaked version ACORN v3 is more vulnerable against the fault attack.

The rest of this paper is organized as follows. In Section 2, a brief description of ACORN is provided. The fault attacks on ACORN v3 and v2 are introduced in Section 3 and 4. Finally, we conclude this paper in Section 4.

2 Brief Description of ACORN

ACORN v3 will be restated briefly in this section, for more details one can refer to [2]. Here we do not intend to introduce the procedures of the initialization, the process of the associated data and the finalization, because our attack does not involve them, and just restate the encryption procedure briefly.

Denote by $S = (s_0, s_1, ..., s_{292})$ the initial state of ACORN v3 before the first keystream bit is outputted and p the plaintext. The functions used in the encryption procedure of ACORN v3 are the feedback function $f(S, p)$, the state update function $F(S, p)$ and the filter function $g(S)$. The feedback function $f(S, p)$ mainly involves in the feedback computation of the FSR, defined as

$$f(S, p) = 1 \oplus s_0 \oplus s_{61} \oplus s_{66} \oplus s_{107} \oplus s_{196} \oplus s_{23}s_{160} \oplus s_{23}s_{244} \oplus s_{160}s_{244} \oplus p.$$

Introducing intermediate variables y_i $(1 \leq i \leq 293)$,

$$\begin{aligned}
y_{293} &= f(S, p) \\
y_{289} &= s_{289} \oplus s_{235} \oplus s_{230} \\
y_{230} &= s_{230} \oplus s_{196} \oplus s_{193} \\
y_{193} &= s_{193} \oplus s_{160} \oplus s_{154} \\
y_{154} &= s_{154} \oplus s_{111} \oplus s_{107} \\
y_{107} &= s_{107} \oplus s_{66} \oplus s_{61} \\
y_{61} &= s_{61} \oplus s_{23} \oplus s_0
\end{aligned}$$

$$y_i = s_i, \ \textit{where } 1 \leq i \leq 292 \textit{ and } i \notin \{61, 107, 154, 193, 230, 289\},$$

the state update function $F(S, p)$ can be described as

$$s_i = y_{i+1}, \ \textit{where } 0 \leq i \leq 292.$$

It is easy to check that the state update function $F(S, p)$ is invertible on S when p is given. The process of introducing intermediate variables can be regarded as a linear transformation L with respect to the internal state. The keystream z is generated by the filter function $g(S)$ defined as

$$g(S) = s_{12} \oplus s_{66} \oplus s_{107} \oplus s_{111} \oplus s_{154} \oplus (s_{61} \oplus s_{23} \oplus s_0)(s_{193} \oplus s_{160} \oplus s_{154}) \oplus (s_{61} \oplus s_{23} \oplus s_0 \oplus s_{193} \oplus s_{160} \oplus s_{154}) s_{235} \oplus (s_{66} \oplus s_{111})(s_{230} \oplus s_{193} \oplus s_{196}).$$

At each step i of the encryption procedure, one plaintext bit p is injected into the state and c is obtained by $p \oplus z$. The pseudo-code of the generation of the encryption procedure of ACORN v3 is given as follows.

$l \leftarrow$ the bit length of the plaintext;
for i from 0 to $l - 1$ **do**
    $z_i = g(S)$;
    $c_i = z_i \oplus p_i$;
    $s = F(S, p_i)$;
end for

For ACORN v2, the only difference from ACORN v3 is that a part of the filter function

$$s_{66} \oplus (s_{66} \oplus s_{111})(s_{230} \oplus s_{193} \oplus s_{196})$$

is moved to the feedback function. Others are the same as those of ACORN v3. For more details, one can refer to [3].

3 Fault Attack on ACORN v3

We first give an outline of the fault attack model before introducing our fault attack on ACORN v3. We assume that an attacker can access the physical device of a stream cipher, and know the IV and the keystream. The attacker just attempts to exploit a fault and tracks the differential trail of the keystream. By analyzing the differential trail, one could deduce some information of the internal state, and then proceed to recover the key or forge a valid tag for any plaintext. In our fault attack, the following privileges of an attacker are required.

1. The attacker has the ability to reset the physical device with the original Key-IV and restart the cipher operations multiple times.
2. The attacker can inject a fault into the initial state randomly before the encryption procedure, but can not choose the location.

Our attack contains two main parts: fault locating and equation solving. At the first step, we will demonstrate how to determine the fault location, and at the second step, we will retrieve a system of equations with respect to the initial state, and exploit how to recover the initial state with this system of equations. Once the initial state is recovered, the forgery attack can be executed easily.

3.1 Fault Locating

In this section, we will present how to identify the fault location after a fault is injected into the initial state randomly. We first introduce a method to obtain differential sets, and then provide a fault locating method.

Denote by $S = (s_0, s_1, ..., s_{292})$ the initial state of the FSR and $P = (p_0, p_1, ..., p_{l-1})$ the $l-$bit plaintext. $[a, b]$ denotes the closed interval from a to b for integers a and b, where $a \leq b$. Let $z = (z_0, z_1, ..., z_{l-1})$ be the correct keystream and $z^i = (z_0^i, z_1^i, ..., z_l^i)$ be the error keystream generated by a faulty initial state at location i, where $i \in [0, 292]$. We define a l-bit differential string Δz^i. The j^{th} element Δz_j^i of Δz^i satisfies $\Delta z_j^i = z_j \oplus z_j^i$. As Δz_j^i is 0, 1 or a non-zero function with respect to S, we also denote by Δz^i a differential set that contains all possible differential strings resulting from the faulty initial state at location i, where $i \in [0, 292]$.

Obtaining differential sets Now, we need to get all differential sets Δz^i, where $i \in [0, 292]$. We represent Δz^i as a sequence of positions where their corresponding components are either 1 or non-constant functions with respect to S by omitting the 0 components. Here, we suppose $l = 99$, since the first 99 bits differential keystream can be represented as linear or quadratic functions with respect to S. These equations can be used to retrieve enough linear equations to recover the initial state. For example, when s_0 is changed, we can get

$$\Delta z^0 = (\Delta z_0^0, 0^{37}, \Delta z_{38}^0, 0^{10}, 1, 0^8, \Delta z_{58}^0, 0^2, \Delta z_{61}^0, 0^{14}, \Delta z_{76}^0, 0^{10}, 1, 0^8, \Delta z_{96}^0, \Delta z_{97}^0, 0)$$

where $0^i (i \in \{37, 10, 8, 2, 14\})$ presents i consecutive 0, and $\Delta z_j^0 (j \in \{0, 38, 58, 61, 76, 96, 97\})$ are non-constant functions with respect to S. Omitting the 0 components, Δz^0 can be represented as

$$\Delta z^0 = (0, 38, \mathbf{49}, 58, 61, 63, 76, \mathbf{87}, 96, 97),$$

where the numbers in bold represent that 1 is always occurring in these positions. Let A be the set of all locations that can be involved in $f(S, p)$ or $g(S)$ directly, that is

$$A = \{0, 12, 23, 61, 66, 107, 111, 154, 160, 193, 196, 230, 235, 244\}.$$

By injecting one fault at loction i, we find that the length of Δz^i is at most 25, where $i \in A$, see Table 1.

For the non-constant function components in each Δz^i, where $i \in A$, we have checked that at most 4 equal to 1 with probability $\frac{1}{4}$ and others equal to 1 with probability $\frac{1}{2}$. So, for one fault location $i \in A$, it is enough to choose 32 initial states randomly to get all the positions where 1 may occur. That is because when 32 initial states are chosen, for one position, 1 should occur, but does not occur with probability less than 2^{-13}. So for 25 positions, the probability that there is at least one position where 1 may occur, but does not occur, can be neglected. For other fault locations i, where $i \in [0, 292]$ and $i \notin A$, the first new differences that are not the differences caused by shifting, is introduced when Δs_i shifts to

Table 1. Δz^i, $i \in A$

i	Δz^i
0	0 38 **49** 58 61 63 76 **87** 96 97
12	**0** 12 50 **61** 70 73 75 88
23	0 **11** 23 38 **49** 58 63 **72** 76 81 84 86 **87** 96 97
61	0 38 41 46 **49** 58 61 63 76 82 84 87 92 **95** 96 97
66	0 5 41 43 46 51 **54** 58 63 66 68 81 82 84 87 89 92 **95** 97
107	**0** 41 43 46 **47** 58 63 82 84 86 87 88 90 92 93 **94** **95** 97
111	0 **4** 43 45 47 50 **51** 62 67 88 **90** 91 92 93 94 96 97 **98**
154	0 33 39 43 **47** 66 **72** 78 82 86 88 90 91 93 **94** 96
160	0 6 33 **39** 45 49 **53** 58 63 66 72 78 82 84 86 **88** 91 92 94 96 97
193	0 33 34 37 39 66 68 70 71 **72** 74 76 78 82 **86** 91 92 96 97
196	0 3 34 36 37 40 42 58 63 68 69 70 71 73 74 **75** 76 77 79 81 85 **89** 92 94 95
230	0 34 37 54 59 68 70 71 74 76 92 93 96 97
235	0 5 39 42 54 59 64 73 75 76 79 81 93 96 97 98
244	9 14 48 51 58 63 68 73 82 84 85 88 90 97

The numbers in bold represent the positions of components 1 and others represent the positions of the non-constant functions with respect to S.

the locations in A. So, the length of Δz^i is at most 25 and 32 random initial states is enough. Algorithm 1 is used to obtain all differential sets. The main idea is to fix one fault location i, and choose 32 initial states randomly to get 32 differential strings. And then extract the positions where 1 may occur and 1 is always occurring. Due to the limitation of pages, we list a part of differential sets Δz^i in Appendix A.

Fault locating method When a fault experiment is carried out, i.e. a fault is injected into a initial state randomly, we can get a differential string Δz, with which we can determine the fault location according to the differential sets obtained above. The main idea is to compare the 1's positions in Δz with those in strings belonging to Δz^i, where $i \in [0, 292]$. In a very small number of cases, a single differential string can correspond to more than one fault location. Because of this, we separate the differential sets into two parts: unique sets and non-unique sets. We analysis the strings separately is to give a more accurate assessment about our fault locating method. Because strings belonging to unique sets, can increase the possibility of determining the fault location. The definition of unique set is given as follows.

Definition 1. *For one differential set Δz^i, if each string belonging to it can determine only one fault location, we say that Δz^i is a unique set, where $i \in [0, 292]$. Otherwise we say that Δz^i is a non-unique set.*

For all 293 fault locations, Algorithm 2 is used to extract unique sets. The major task is to compare the locations where 1 is always occurring. Running through

Algorithm 1 Obtain differential set Δz^i, where $i \in [0, 292]$

Require: fault location i, where $i \in [0, 292]$
Ensure: the position set MQ^i where 1 occurs with probability less than 1 and the position set AQ^i where 1 is always occurring

Choose 32 initial states randomly
for each initial state **do**
  proceed the encryption phase of ACORN v3 to get a l-bit keystream z
  $s_i \leftarrow s_i \oplus 1$
  proceed the encryption phase of ACORN v3 to get a l-bit keystream z^i, calculate Δz^i
  for j from 0 to $l-1$ **do**
    if $\Delta z_j^i \neq 0$ **then**
      add j to MQ^i (repeated j are always reserved)
    end if
  end for
end for
for j from 0 to $l-1$ **do**
  if there are 32 j in MQ^i **then**
    add j to AQ^i and delete j in MQ^i
  end if
end for
delate the repeated numbers in MQ^i
return MQ^i and AQ^i

Algorithm 2 Find unique set Δz^i, where $i \in [0, 292]$

Require: differential sets Δz^i, MQ^i and AQ^i, where $i \in [0, 292]$
Ensure: unique sets and non-unique sets

for i from 0 to 291 **do**
  for k from $i+1$ to 292 **do**
    if $MQ^i \cup AQ^i \subseteq MQ^k \cup AQ^k$ and $AQ^k \subseteq MQ^i \cup AQ^i$ **then**
      return Δz^i and Δz^k are non-unique set
    end if
  end for
  if there is not any return **then**
    return Δz^i is a unique set
  end if
end for

all differential sets Δz^i, where $i \in [0, 292]$, we find that there are 103 unique sets which are marked out in Appendix A. The number of unique sets can provide some insights into the diffusion ability of the cipher to some extent. Given one differential string, if it belongs to one of the unique set, the unique fault location is clear. The strings belonging to non-unique sets can also be divided into two parts. One part can determine the fault locations uniquely and the other part can not. For the strings that can not determine the fault location uniquely, we adopt the keystream extension strategy and the high probability priority strategy to increase the possibility of determining the fault location. The detail process is shown as follows.

1. Separate strings belonging to non-unique sets into 99 categories denoted by B_t ($t \in [0, 98]$) according to the subscript t satisfying $\Delta z_t^i = 1$ ($t \in [0, 98]$) and $\Delta z_j^i = 0$ ($0 \leq j < t$). For example, B_0 contains Δz^i whose first component Δz_0^i can be 1. It is noticed that for
$$\Delta z^0 = (0, 38, \mathbf{49}, 58, 61, 63, 76, \mathbf{87}, 96, 97),$$
it may occurs in B_0, B_{38} and B_{49} since its first 1 may occur at position $0, 38$ and 49 ($\Delta z_{49}^1 = 1$ always holds).
2. For a given differential string Δz, we need to determine which category it belongs to according to the position of its first 1. And then by comparing other locations of 1 appearing in Δz, we can determine all possible fault locations, see Algorithm 3.

Algorithm 3 Fault Locating

Require: a differential string Δz, MQ^i and AQ^i, where $i \in [0, 292]$
Ensure: the set I of possible fault locations
1: Denote by J the set of j satisfying $\Delta z_j = 1$, where $j \in [0, l-1]$
2: Determine the category B^* that Δz belongs to, according to the first 1's position in Δz
3: **for** all $\Delta z^i \in B^*$ **do**
4: **if** $J \subseteq MQ^i \cup AQ^i$ and $AQ^i \subseteq J$ **then**
5: add location i to the set I
6: **end if**
7: **end for**
8: **return** the set I

3. If the number of the optional fault locations is one, it means that the unique fault location has been determined. Otherwise, we use the keystream bits extension strategy and the high probability priority strategy to guess the right fault location.
   - **Keystream extension strategy.** Extending keystream is a very valid method of increasing the proportion of strings that can determine the fault location uniquely. The longer the keystream available to us, the higher probability of determining the unique fault location.

– **High probability priority strategy.** Here we assume that the initial state of the FSR is random and uniformly distributed. For a given string Δz, we find that different fault location candidates appear with different probabilities. For each candidate i, we prefer to choose i with higher probability, and call it high probability priority strategy. For example, when we get

$$\Delta z = (\overbrace{0, \cdots, 0}^{57}, 1, \overbrace{0 \cdots, 0}^{41}),$$

we know each candidate i in B_{57} needs to satisfy $\Delta z^i_j = 0$, where $j \in [0, 98]$ and $j \neq 57$. According to the expression of Δz^i, i takes 292 with the highest probability 2^{-3}. The probabilities of all the candidates i are listed in Table 2.

Table 2. Optional fault locations of $\Delta z^i = (\mathbf{57})$

fault location	Δz^i	probability
233	3 37 40 57 62 71 73 74 77 79 95 96	2^{-12}
238	3 8 42 45 57 62 67 76 78 79 82 84 96	2^{-13}
250	15 20 54 57 64 69 74 79 88 90 91 94 96	2^{-13}
253	18 23 57 60 67 72 77 82 91 93 94 97	2^{-12}
287	52 57 91 94	2^{-4}
292	57 62 96	2^{-3}

Table 3. Determine the Fault Location

2^{20} strings in non-unique sets				
keystream length (bits)	uniquely determine proportion(%)	non-uniquely determine		total proportion(%)
		proportion(%)	guess probability(%)	
99	86.48	13.52	83.01	97.70
109	91.31	8.69	86.21	98.80
119	92.38	7.62	92.82	99.45
129	93.31	6.69	98.11	99.87
139	93.84	6.16	99.07	99.94
149	94.48	5.52	99.59	99.98
159	94.66	5.34	99.80	99.99
169	94.72	5.28	99.96	100.00
179	94.73	5.27	99.98	100.00

uniquely determine proportion(%): the proportion of strings that can determine the fault location uniquely
non-uniquely determine: the strings that can not determine the the fault location uniquely
proportion(%): the proportion of strings that can not determine the fault location uniquely
guess probability(%): the probability of guessing the right fault location
total proportion(%): the total proportion of strings that can get the right fault location

Implementation and verification In ACORN v3, there are at most 2^{293} initial states and a differential string is dependent on both the initial state and the fault location. The whole space is beyond our computation capability. We just choose 2^{20} differential strings belonging to non-unique sets randomly to verify the validity of the above two strategies. We choose 2^{15} initial states randomly, and for each initial state, we choose 2^5 non-unique fault locations randomly to obtain 2^{20} differential strings. Then we calculate the probability of guessing the right fault location every time 10-bit keystream is lengthened. The result shows that when the length of the keystream reaches to 169 bits, we can determine the right fault location with probability 1, see Table 3. When the keystream length is 99 bits, the proportion of the strings that can determine the fault location uniquely is about 86.48%. For the other 13.52% strings, we can guess the right fault location with probability 83.01%. The total proportion of strings that can get the right fault location is

$$86.48\% + 13.52\% * 83.01\% = 97.70\%.$$

When the keystream length reaches to 169 bits, the total proportion is almost 100%.

3.2 Recovering the Initial State

Once several faults are located, we can retrieve enough equations with respect to the initial state to recover the initial state. In this section, we first give a fundamental method to retrieve equations and then give some improvement strategies to get more linear equations. Last, we use the guess-and-determine method to obtain the initial state and the time complexity is bounded by the number of fault experiments.

Fundamental equation retrieving method Here, we just use the first 99-bit keystream, as the first 99 bits differential keystream can be expressed as linear or quadratic functions with respect to the initial state. Denote by $S = (s_0, s_1, \cdots, s_{292})$ the initial state of the FSR and $s_{293}, \cdots, s_{391}$ the 99 feedback variables. The first 58 feedback variables can be expressed as quadratic functions with respect to S. We give two algorithms to retrieve equations. In Algorithm 4, we show how to get differential equations when fault is injected in s_i, where $i \in A$,

$$A = \{0, 12, 23, 61, 66, 107, 111, 154, 160, 193, 196, 230, 235, 244\}.$$

When $i \in [0, 292]$ and $i \notin A$, the main idea to retrieve differential equations is to shift and act the inversion of the linear transformation L on $\Delta z^{i'}$, where $i' \in A$, see Algorithm 5. Note that the inversion of the linear transformation L will not lead to the transformation of a linear function to a non-linear function, but increase the number of terms in the function.

Algorithm 4 Retrieve Equations 1

Require: the set A of fault locations
Ensure: differential equations
proceed the encryption phase of ACORN v3 to represent the 99 feedback variables $s'_{293}, \cdots, s'_{391}$ and keystream $z_0, z_1, \cdots, z_{98}$ as functions of S
for $i \in A$ **do**
  $s_i \leftarrow s_i \oplus 1$
  proceed the encryption phase of ACORN v3 to represent the 99 feedback variables $s''_{293}, \cdots, s''_{391}$ as functions of S
  for j from 293 to 391 **do**
    $s_j \leftarrow s_j \oplus s'_j \oplus s''_j$
  end for
  Regard $S = (s_0, s_1, \cdots, s_{292})$ and $s_{293}, \cdots, s_{391}$ as the initial state and feedback variables, proceed the encryption phase of ACORN v3 to represent the keystream $z'_0, z'_1, \cdots, z'_{98}$ as functions of S
  for j from 0 to 98 **do**
    $\Delta z^i_j \leftarrow z'_j \oplus z_j$
    if $\Delta z^i_j \neq 0, 1$ **then**
      return Δz^i_j is a differential equation
    end if
  end for
end for

Algorithm 5 Retrieve Equations 2

Require: fault location $i \in [a, b]$, where $a - 1, b + 1 \in A$ and there is not any $c \in A$ satisfying $a < c < b$; the components of Δz^{a-1}
Ensure: differential equations
for each component $\Delta z^{a-1}_j \neq 0, 1$, where $j \in [0, 98]$ **do**
  $\Delta z^i_{j+a-1-i} \leftarrow \Delta z^{a-1}_j$
  for each variable s_k in $\Delta z^i_{j+a-1-i}$, where $k \in [0, 292]$ **do**
    $s_k \leftarrow s_k^{a-1-i}$
    $s^0_{k+a-1-i} \leftarrow \underbrace{L^{-1}(L^{-1}(\cdots(L^{-1}(L^{-1}}_{a-1-i}(s_k^{a-1-i}))) \cdots)$
    $s_k^{a-1-i} \leftarrow s^0_{k+a-1-i}$
    return $\Delta z^i_{j+a-1-i}$ is a differential equation
  end for
end for

Statistics show that, on average, we can get 6.38 linear equations and 4.23 non-linear equations with one fault experiment. We choose the quadratic equations of form $x_i x_j$ as 0.5 linear equations, where x_i and x_j are linear functions with respect to the initial state. Because the value of the quadratic equations of the form $x_i x_j$ equal to 1 with probability $\frac{1}{4}$. If $x_i x_j = 1$, we know that $x_i = 1$ and $x_j = 1$. Thus it is expected to obtain 0.5 linear equation.

Several improvement strategies In order to get more linear equations, two observations are given.

Observation 1 *When $i \notin A$, if we stretch the length of the differential string, we can get more linear equations. The number of these functions is 0.7 on average.*

For example, when s_0 is changed, we can get

$$\Delta z^0 = (\Delta z^0_0, 0^{37}, \Delta z^0_{38}, 0^{10}, 1, 0^8, \Delta z^0_{58}, 0^2, \Delta z^0_{61}, 0^{14}, \Delta z^0_{76}, 0^{10}, 1, 0^8, \Delta z^0_{96}, \Delta z^0_{97}, 0)$$

where $0^i,\ i \in \{37, 10, 8, 2, 14\}$ presents i consecutive 0, and
$\Delta z^0_0 = s_{154} \oplus s_{160} \oplus s_{193} \oplus s_{235}$,
$\Delta z^0_{38} = s_{159} \oplus s_{165} \oplus s_{192} \oplus s_{194} \oplus s_{197} \oplus s_{198} \oplus s_{231} \oplus s_{273}$,
$\Delta z^0_{58} = s_{20} \oplus s_{43} \oplus s_{58} \oplus s_{73} \oplus s_{78} \oplus s_{81} \oplus s_{119} \oplus s_{173} \oplus s_{185} \oplus s_{212} \oplus s_{214} \oplus s_{217} \oplus s_{218} \oplus s_{251}$,
$\Delta z^0_{61} = s_{176} \oplus s_{188} \oplus s_{215} \oplus s_{217} \oplus s_{220} \oplus s_{221} \oplus s_{237} \oplus s_{242} \oplus s_{254} \oplus \mathbf{s_{296}}$,
$\Delta z^0_{63} = s_{83} \oplus s_{88} \oplus s_{127} \oplus s_{129} \oplus s_{131} \oplus s_{174}$,
$\Delta z^0_{76} = s_{164} \oplus s_{170} \oplus s_{191} \oplus s_{193} \oplus s_{195} \oplus s_{196} \oplus s_{199} \oplus s_{201} \oplus s_{202} \oplus s_{203} \oplus \oplus s_{230} \oplus s_{232} \oplus s_{235} \oplus s_{236} \oplus s_{252} \oplus s_{257} \oplus s_{269} \oplus \mathbf{s_{311}}$,
$\Delta z^0_{96} = (s_{159} \oplus s_{165} \oplus s_{198} \oplus s_{282})(s_{20} \oplus s_{35} \oplus s_{43} \oplus s_{65} \oplus s_{73} \oplus s_{75} \oplus s_{78} \oplus s_{81} \oplus s_{96} \oplus s_{110} \oplus s_{111} \oplus s_{114} \oplus s_{116} \oplus s_{119} \oplus s_{157} \oplus s_{172} \oplus s_{178} \oplus s_{184} \oplus s_{190} \oplus s_{211} \oplus s_{213} \oplus s_{215} \oplus s_{216} \oplus s_{219} \oplus s_{221} \oplus s_{222} \oplus s_{223} \oplus s_{230} \oplus s_{235} \oplus s_{250} \oplus s_{252} \oplus s_{255} \oplus s_{256} \oplus s_{289})$,
$\Delta z^0_{97} = 1 \oplus s_{71} \oplus s_{81} \oplus s_{114} \oplus s_{116} \oplus s_{117} \oplus s_{120} \oplus s_{161} \oplus s_{163} \oplus s_{165} \oplus s_{169} \oplus s_{175} \oplus s_{208}$.

Δz^0_{96} is of form $x_i x_j$ and can be regard as 0.5 linear equations, where $x_i = s_{159} \oplus s_{165} \oplus s_{198} \oplus s_{282}$ and
$x_j = s_{20} \oplus s_{35} \oplus s_{43} \oplus s_{65} \oplus s_{73} \oplus s_{75} \oplus s_{78} \oplus s_{81} \oplus s_{96} \oplus s_{110} \oplus s_{111} \oplus s_{114} \oplus s_{116} \oplus s_{119} \oplus s_{157} \oplus s_{172} \oplus s_{178} \oplus s_{184} \oplus s_{190} \oplus s_{211} \oplus s_{213} \oplus s_{215} \oplus s_{216} \oplus s_{219} \oplus s_{221} \oplus s_{222} \oplus s_{223} \oplus s_{230} \oplus s_{235} \oplus s_{250} \oplus s_{252} \oplus s_{255} \oplus s_{256} \oplus s_{289}$.
Note that Δz^0_{96} and Δz^0_{97} can be regarded as linear functions. Δz^3_{99} and $\Delta z^i_{97+i} (1 < i < 12)$ are also linear functions which will not be used when only 99-bit keystream is in consideration.

Observation 2 *For all equations, we find two features of the equations.*

1. *For one fault experiment, if the number of quadratic equations $x_i x_j$ is larger than 1, there would exist equations of forms $x_i x_{j_1}$ and $x_i x_{j_2}$, where x_i, x_{j_1} and x_{j_2} are linear functions with respect to the initial state. Then we can retrieve more linear equations.*
2. *For one quadratic equation of form $x_i x_j$, there may exist linear equation of form x_i or x_j which can be used to deduce more linear equations, where x_i and x_j are linear functions with respect to the initial state.*

For two experiments where faults are injected at location $i = 0$ and $i = 23$, we can get

1. In Δz^{23}, there are four quadratic equations of form $x_i x_j$, three of which are $\Delta z^{23}_{58} = (s_{160} \oplus s_{244})(s_{20} \oplus s_{43} \oplus s_{58} \oplus s_{73} \oplus s_{78} \oplus s_{81} \oplus s_{119} \oplus s_{173} \oplus s_{185} \oplus$

$s_{212} \oplus s_{214} \oplus s_{217} \oplus s_{218} \oplus s_{251}$),
$\Delta z_{63}^{23} = (s_{160} \oplus s_{244})(s_{83} \oplus s_{88} \oplus s_{127} \oplus s_{129} \oplus s_{131} \oplus s_{174})$,
$\Delta z_{97}^{23} = (s_{160} \oplus s_{244})(1 \oplus s_{71} \oplus s_{81} \oplus s_{114} \oplus s_{116} \oplus s_{117} \oplus s_{120} \oplus s_{161} \oplus s_{163} \oplus s_{165} \oplus s_{169} \oplus s_{175} \oplus s_{208})$.
If at least one of them equal to 1, we can get 4 linear equations.

2. In Δz^0, two equations
$\Delta z_{58}^0 = s_{20} \oplus s_{43} \oplus s_{58} \oplus s_{73} \oplus s_{78} \oplus s_{81} \oplus s_{119} \oplus s_{173} \oplus s_{185} \oplus s_{212} \oplus s_{214} \oplus s_{217} \oplus s_{218} \oplus s_{251}$, and $\Delta z_{63}^0 = s_{83} \oplus s_{88} \oplus s_{127} \oplus s_{129} \oplus s_{131} \oplus s_{174}$,
are parts of Δz_{58}^{23} and Δz_{63}^{23} respectively.

Solving equation As shown above, on average, we can get 7.03 linear equations and 4.23 non-linear equations with one fault experiment. With 27 fault experiments, we can get 304 equations including 190 linear equations. By guessing 52 bits value, the initial states can be recovered. The time complexity of recovering the initial state is $c \cdot 2^{52}$, where c is the time complexity of solving linear equations of 51 variables. Also, we can get 295 linear equations with 42 fault experiments and the time complexity is to solve linear equations of 293 variables.

Let n be the number of fault experiments. We can get $11.26n$ equations including $7.03n$ linear equations. We use the guess-and-determine method to solve the equations. The time complexity of obtaining the initial state equals to

$$c \cdot 2^{\frac{293-7.03n}{2}} \approx c \cdot 2^{146.5-3.52n}$$

approximately, where c is the time complexity of solving linear equations and $26 < n < 43$. As there are some relations between the equations in practical attack as shown in Observation 2, the time complexity can be smaller.

Implementation and verification We verify the validity of our solving equation method on a shrunk cipher with similar structure and properties. More specifically, we built a small stream cipher according to the design principles used for ACORN v3 but with a small state of 31 bits. We then implemented our attack to recover the initial state. The result shows that if the number of linearly independent equations is larger than 31, we can recover the initial state by guessing some feedback values and a small part of the initial state values involved in these feedback function. Of course, if the linearly independent equations are not enough, we need to proceed more fault experiments.

3.3 The Forgery Attack

Once the initial state of ACORN v3 is recovered we can encrypt any message to generate a valid tag, i.e., we can forge tags for all plaintexts. All the methods used in this work can be easily applied to ACORN v1 and v2, and for ACORN v1, we can recover the key by stepping the cipher backward.

4 The Fault Attack on ACORN v2

We also apply the above attack to ACORN v2. In the fault locating part, we find that there are 127 unique sets in ACORN v2 which is larger than that of ACORN v3. And for strings belonging to non-unique sets, we can also determine the fault location uniquely with the keystream extension strategy and the high probability priority strategy. In the initial state recovery part, we can get 3.9 linear equations and 3.3 non-linear equations, on average, with one fault experiment. And in ACORN v2, observation 1 is not useful to retrieve more linear equations.

Let n be the number of fault experiments. We can get $7.2n$ equations with $3.9n$ linear equations. The time complexity of obtaining the initial state equals to

$$c \cdot 2^{\frac{293-3.9n}{2}} = c \cdot 2^{146.5-1.95n},$$

where c is the time complexity of solving linear equations and $40 < n < 77$.

The result shows that comparing with ACORN v2, the tweaked version ACORN v3 is more vulnerable against fault attack, see Table 4. The main reason

Table 4. Comparation of ACORN v3 and v2

fault experiments' number	time complexity	
	ACORN v3	ACORN v2
n	$c \cdot 2^{146.5-3.52n}$	$c \cdot 2^{146.5-1.95n}$
42	c	$c \cdot 2^{64.6}$
27	$c \cdot 2^{51.46}$	impossible

is caused by the tweak that a part of terms in the feedback function are moved to the output filtering functions. For one experiment, the number of linear equations retrieved from ACORN v3 is larger than that from ACORN v2. The tweak is to provide large security margin against the guess-and-determine attack. However, it makes the algorithm more vulnerable against the fault attack.

5 Conclusion

In this paper, we described a fault attack on ACORN v3 which is one of the third round candidates of CAESAR. We also applied the attack to ACORN v2. This work shows that comparing with ACORN v2, the tweaked version ACORN v3 is more vulnerable against fault attack. For ACORN v3, we can recover the initial state with time complexity $c \cdot 2^{146.5-3.52n}$, where c is the time complexity of solving linear equations and $26 < n < 43$. However, for ACORN v2, the time complexity is $c \cdot 2^{146.5-1.95n}$ with $40 < n < 77$. The difference between ACORN v3 and ACORN v2 makes the algorithm small security margin against the differential fault attack.

References

1. CAESAR: http://competitions.cr.yp.to/index.html.
2. Hongjun Wu. ACORN: A Lightweight Authenticated Cipher (v3). Submission to CAESAR, http://competitions.cr.yp.to/round3/acornv3.pdf, 2016.
3. Hongjun Wu. ACORN: A Lightweight Authenticated Cipher (v2). Submission to CAESAR, http://competitions.cr.yp.to/round2/acornv2.pdf, 2015.
4. Hongjun Wu. ACORN: A Lightweight Authenticated Cipher (v1). Submission to CAESAR, http://competitions.cr.yp.to/round1/acornv1.pdf, 2014.
5. Meicheng Liu and Dongdai Lin. Cryptanalysis of Lightweight Authenticated Cipher ACORN. Posed on the crypto-competition mailing list, 2014.
6. Colin Chaigneau, Thomas Fuhr and Henri Gilbert. Full Key-recovery on ACORN in Nonce-reuse and Decryption-misuse settings. Posed on the crypto-competition mailing list, 2015.
7. Md Iftekhar Salam, Harry Bartlett, Ed Dawson, Josef Pieprzyk, Leonie Simpson and Kenneth Koon-Ho Wong. Investigating Cube Attacks on the Authenticated Encryption Stream Cipher ACORN. International Conference on Applications and Techniques in Information Security. Springer Singapore, 2016: 15-26.
8. Md Iftekhar Salam, Kenneth Koon-Ho Wong, Harry Bartlett, Leonie Ruth Simpson, Ed Dawson and Josef Pieprzyk. Finding state collisions in the authenticated encryption stream cipher ACORN. Proceedings of the Australasian Computer Science Week Multiconference. ACM, 2016: 36.
9. Frédéric Lafitte, Liran Lerman, Olivier Markowitch and Dirk Van Heule. SAT-based cryptanalysis of ACORN. IACR Cryptology ePrint Archive, 2016: 521.
10. Rebhu Johymalyo Josh and Santanu Sarkar. Some observations on ACORN v1 and Trivia-SC. Lightweight Cryptography Workshop, NIST, USA. 2015: 20-21.
11. Pei Zhang, Jie Guan, Junzhi Li and Tairong Shi. Research on State Collisions of Authenticated Cipher ACORN. 2015 4th International Conference on Sensors, Measurement and Intelligent Materials. Atlantis Press, 2016.
12. Roy Dibyendu and Sourav Mukhopadhyay. Some results on ACORN. IACR cryptology ePrint Archive, 2016: 1132.
13. https://groups.google.com/forum/#!forum/crypto-competitions/dzzNcybqFP4
14. Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. Advances in CryptologyCRYPTO'97, 1997: 513-525.
15. Jonathan J. Hoch and Adi Shamir. Fault Analysis of Stream Ciphers. International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2004: 240-253.
16. Sergei P. Skorobogatov. Optically Enhanced Position-Locked Power Analysis. CHES. 2006, 4249: 61-75.
17. Sergei P. Skorobogatov and Ross J. Anderson. Optical Fault Induction Attacks. CHES. 2002, 2523: 2-12.
18. Dey Prakash, Rohit Raghvendra Singh and Adhikari Avishek. Full key recovery of ACORN with a single fault. Journal of Information Security and Applications, 2016, 29: 57-64.

A Appendix

Due to the limitation of pages, we just list the differential sets Δz^i, where $i \in [0, 168]$ in Table 5. The first column is fault location, and the second column is differential set. For each differential set Δz^i, the numbers in the second column represent the positions where 1 may occur when fault location is i, that is, fault is injected in s_i. And the numbers in bold represent the positions where 1 is always occurring. The numbers in the first column are the locations where fault are injected, and the numbers in bold represent that the corresponding differential sets are unique sets.

For example, when $i = 12$, the components of Δz^{12} are the positions where 1 may occur when fault is injected in s_{12}, where

$$\Delta z^{12} = (\mathbf{0}, 12, 50, \mathbf{61}, 70, 73, 75, 88).$$

The number **0** and **61** represent the positions where 1 is always occurring. Other numbers represent the positions where 1 may occur with some certain probability. **12** in the first column means that the differential set Δz^{12} is an unique set.

Table 5. Δz^i, $i \in [0, 168]$

i	Δz^i
0	0 38 **49** 58 61 63 76 **87** 96 97
1	1 39 **50** 59 62 64 77 **88** 97 98
2	2 40 **51** 60 63 65 78 **89** 98
3	3 41 **52** 61 64 66 79 **90**
4	4 42 **53** 62 65 67 80 **91**
5	5 43 **54** 63 66 68 81 **92**
6	6 44 **55** 64 67 69 82 **93**
7	7 45 **56** 65 68 70 83 **94**
8	8 46 **57** 66 69 71 84 **95**
9	9 47 **58** 67 70 72 85 **96**
10	10 48 **59** 68 71 73 86 **97**
11	11 49 **60** 69 72 74 87 **98**
12	**0** 12 50 **61** 70 73 75 88
13	**1** 13 51 **62** 71 74 76 89
14	**2** 14 52 **63** 72 75 77 90
15	**3** 15 53 **64** 73 76 78 91
16	**4** 16 54 **65** 74 77 79 92
17	**5** 17 55 **66** 75 78 80 93
18	**6** 18 56 **67** 76 79 81 94
19	**7** 19 57 **68** 77 80 82 95
20	**8** 20 58 **69** 78 81 83 96
21	**9** 21 59 **70** 79 82 84 97
22	**10** 22 60 **71** 80 83 85 98
23	0 **11** 23 38 **49** 58 63 **72** 76 81 84 86 **87** 96 97
24	1 **12** 24 39 **50** 59 64 **73** 77 82 85 87 **88** 97 98
25	2 **13** 25 40 **51** 60 65 **74** 78 83 86 88 **89** 98
26	3 **14** 26 41 **52** 61 66 **75** 79 84 87 89 **90**
27	4 **15** 27 42 **53** 62 67 **76** 80 85 88 90 **91**
28	5 **16** 28 43 **54** 63 68 **77** 81 86 89 91 **92**
29	6 **17** 29 44 **55** 64 69 **78** 82 87 90 92 **93**
30	7 **18** 30 45 **56** 65 70 **79** 83 88 91 93 **94**
31	8 **19** 31 46 **57** 66 71 **80** 84 89 92 94 **95**
32	9 **20** 32 47 **58** 67 72 **81** 85 90 93 95 **96**
33	10 **21** 33 48 **59** 68 73 **82** 86 91 94 96 **97**
34	11 **22** 34 49 **60** 69 74 **83** 87 92 95 97 **98**
35	12 **23** 35 50 **61** 70 75 **84** 88 93 96 98
36	13 **24** 36 51 **62** 71 76 **85** 89 94 97
37	14 **25** 37 52 **63** 72 77 **86** 90 95 98
38	15 **26** 38 53 **64** 73 78 **87** 91 96
39	16 **27** 39 54 **65** 74 79 **88** 92 97
40	17 **28** 40 55 **66** 75 80 **89** 93 98
41	18 **29** 41 56 **67** 76 81 **90** 94
42	19 **30** 42 57 **68** 77 82 **91** 95
43	20 **31** 43 58 **69** 78 83 **92** 96
44	21 **32** 44 59 **70** 79 84 **93** 97
45	22 **33** 45 60 **71** 80 85 **94** 98
46	23 **34** 46 61 **72** 81 86 **95**
47	24 **35** 47 62 **73** 82 87 **96**
48	25 **36** 48 63 **74** 83 88 **97**
49	26 **37** 49 64 **75** 84 89 **98**
50	27 **38** 50 65 **76** 85 90
51	28 **39** 51 66 **77** 86 91
52	29 **40** 52 67 **78** 87 92
53	30 **41** 53 68 **79** 88 93
54	31 **42** 54 69 **80** 89 94
55	32 **43** 55 70 **81** 90 95
56	33 **44** 56 71 **82** 91 96
57	34 **45** 57 72 **83** 92 97
58	35 **46** 58 73 **84** 93 98
59	36 **47** 59 74 **85** 94
60	37 **48** 60 75 **86** 95
61	0 38 41 46 **49** 58 61 63 76 82 84 87 92 **95** 96 97
62	1 39 42 47 **50** 59 62 64 77 83 85 88 93 **96** 97 98
63	2 40 43 48 **51** 60 63 65 78 84 86 89 94 **97** 98
64	3 41 44 49 **52** 61 64 66 79 85 87 90 95 **98**
65	4 42 45 50 **53** 62 65 67 80 86 88 91 96
66	0 5 41 43 46 51 **54** 58 63 66 68 81 82 84 87 89 92 **95** 97
67	1 6 42 44 47 52 **55** 59 64 67 69 82 83 85 88 90 93 **96** 98
68	2 7 43 45 48 53 **56** 60 65 68 70 83 84 86 89 91 94 **97**
69	3 8 44 46 49 54 **57** 61 66 69 71 84 85 87 90 92 95 **98**
70	4 9 45 47 50 55 **58** 62 67 70 72 85 86 88 91 93 96
71	5 10 46 48 51 56 **59** 63 68 71 73 86 87 89 92 94 97
72	6 11 47 49 52 57 **60** 64 69 72 74 87 88 90 93 95 98
73	7 12 48 50 53 58 **61** 65 70 73 75 88 89 91 94 96
74	8 13 49 51 54 59 **62** 66 71 74 76 89 90 92 95 97
75	9 14 50 52 55 60 **63** 67 72 75 77 90 91 93 96 98
76	10 15 51 53 56 61 **64** 68 73 76 78 91 92 94 97
77	11 16 52 54 57 62 **65** 69 74 77 79 92 93 95 98
78	12 17 53 55 58 63 **66** 70 75 78 80 93 94 96
79	13 18 54 56 59 64 **67** 71 76 79 81 94 95 97
80	14 19 55 57 60 65 **68** 72 77 80 82 95 96 98
81	15 20 56 58 61 66 **69** 73 78 81 83 96 97
82	16 21 57 59 62 67 **70** 74 79 82 84 97 98
83	17 22 58 60 63 68 **71** 75 80 83 85 98

i	Δz^i
84	18 23 59 61 64 69 **72** 76 81 84 86
85	19 24 60 62 65 70 **73** 77 82 85 87
86	20 25 61 63 66 71 **74** 78 83 86 88
87	21 26 62 64 67 72 **75** 79 84 87 89
88	22 27 63 65 68 73 **76** 80 85 88 90
89	23 28 64 66 69 74 **77** 81 86 89 91
90	24 29 65 67 70 75 **78** 82 87 90 92
91	25 30 66 68 71 76 **79** 83 88 91 93
92	26 31 67 69 72 77 **80** 84 89 92 94
93	27 32 68 70 73 78 **81** 85 90 93 95
94	28 33 69 71 74 79 **82** 86 91 94 96
95	29 34 70 72 75 80 **83** 87 92 95 97
96	30 35 71 73 76 81 **84** 88 93 96 98
97	31 36 72 74 77 82 **85** 89 94 97
98	32 37 73 75 78 83 **86** 90 95 98
99	33 38 74 76 79 84 **87** 91 96
100	34 39 75 77 80 85 **88** 92 97
101	35 40 76 78 81 86 **89** 93 98
102	36 41 77 79 82 87 **90** 94
103	37 42 78 80 83 88 **91** 95
104	38 43 79 81 84 89 **92** 96
105	39 44 80 82 85 90 **93** 97
106	40 45 81 83 86 91 **94** 98
107	**0** 41 43 46 **47** 58 63 82 84 86 87 88 90 92 93 **94** **95** 97
108	**1** 42 44 47 **48** 59 64 83 85 87 88 89 91 93 94 **95** **96** 98
109	**2** 43 45 48 **49** 60 65 84 86 88 89 90 92 94 95 **96** **97**
110	**3** 44 46 49 **50** 61 66 85 87 89 90 91 93 95 96 **97** **98**
111	0 **4** 43 45 47 50 **51** 62 67 88 **90** 91 92 93 94 96 97 **98**
112	1 **5** 44 46 48 51 **52** 63 68 89 **91** 92 93 94 95 97 98
113	2 **6** 45 47 49 52 **53** 64 69 90 **92** 93 94 95 96 98
114	3 **7** 46 48 50 53 **54** 65 70 91 **93** 94 95 96 97
115	4 **8** 47 49 51 54 **55** 66 71 92 **94** 95 96 97 98
116	5 **9** 48 50 52 55 **56** 67 72 93 **95** 96 97 98
117	6 **10** 49 51 53 56 **57** 68 73 94 **96** 97 98
118	7 **11** 50 52 54 57 **58** 69 74 95 **97** 98
119	8 **12** 51 53 55 58 **59** 70 75 96 **98**
120	9 **13** 52 54 56 59 **60** 71 76 97
121	10 **14** 53 55 57 60 **61** 72 77 98
122	11 **15** 54 56 58 61 **62** 73 78
123	12 **16** 55 57 59 62 **63** 74 79
124	13 **17** 56 58 60 63 **64** 75 80
125	14 **18** 57 59 61 64 **65** 76 81
126	15 **19** 58 60 62 65 **66** 77 82
127	16 **20** 59 61 63 66 **67** 78 83
128	17 **21** 60 62 64 67 **68** 79 84
129	18 **22** 61 63 65 68 **69** 80 85
130	19 **23** 62 64 66 69 **70** 81 86
131	20 **24** 63 65 67 70 **71** 82 87
132	21 **25** 64 66 68 71 **72** 83 88
133	22 **26** 65 67 69 72 **73** 84 89
134	23 **27** 66 68 70 73 **74** 85 90
135	24 **28** 67 69 71 74 **75** 86 91
136	25 **29** 68 70 72 75 **76** 87 92
137	26 **30** 69 71 73 76 **77** 88 93
138	27 **31** 70 72 74 77 **78** 89 94
139	28 **32** 71 73 75 78 **79** 90 95
140	29 **33** 72 74 76 79 **80** 91 96
141	30 **34** 73 75 77 80 **81** 92 97
142	31 **35** 74 76 78 81 **82** 93 98
143	32 **36** 75 77 79 82 **83** 94
144	33 **37** 76 78 80 83 **84** 95
145	34 **38** 77 79 81 84 **85** 96
146	35 **39** 78 80 82 85 **86** 97
147	36 **40** 79 81 83 86 **87** 98
148	37 **41** 80 82 84 87 **88**
149	38 **42** 81 83 85 88 **89**
150	39 **43** 82 84 86 89 **90**
151	40 **44** 83 85 87 90 **91**
152	41 **45** 84 86 88 91 **92**
153	42 **46** 85 87 89 92 **93**
154	0 33 39 43 **47** 66 **72** 78 82 86 88 90 91 93 **94** 96
155	1 34 40 44 **48** 67 **73** 79 83 87 89 91 92 94 **95** 97
156	2 35 41 45 **49** 68 **74** 80 84 88 90 92 93 95 **96** 98
157	3 36 42 46 **50** 69 **75** 81 85 89 91 93 94 96 **97**
158	4 37 43 47 **51** 70 **76** 82 86 90 92 94 95 97 **98**
159	5 38 44 48 **52** 71 **77** 83 87 91 93 95 96 98
160	0 6 33 **39** 45 49 **53** 58 63 66 72 78 82 84 86 **88** 91 92 94 96 97
161	1 7 34 **40** 46 50 **54** 59 64 67 73 79 83 85 87 **89** 92 93 95 97 98
162	2 8 35 **41** 47 51 **55** 60 65 68 74 80 84 86 88 **90** 93 94 96 98
163	3 9 36 **42** 48 52 **56** 61 66 69 75 81 85 87 89 **91** 94 95 97
164	4 10 37 **43** 49 53 **57** 62 67 70 76 82 86 88 90 **92** 95 96 98
165	5 11 38 **44** 50 54 **58** 63 68 71 77 83 87 89 91 **93** 96 97
166	6 12 39 **45** 51 55 **59** 64 69 72 78 84 88 90 92 **94** 97 98
167	7 13 40 **46** 52 56 **60** 65 70 73 79 85 89 91 93 **95** 98
168	8 14 41 **47** 53 57 **61** 66 71 74 80 86 90 92 94 **96**