Practical State Recovery Attack on ICEPOLE

Tao Huang, Hongjun Wu, Ivan Tjuawinata

Nanyang Technological University, singapore

Abstract. ICEPOLE is a CAESAR candidate which is claimed to have intermediate level of robustness under nonce reuse circumstances. In this report, we apply the differential-linear cryptanalysis to the ICEPOLE family and show that ICEPOLE is insecure when the nonce is reused. Under the nonce-misuse circumstances, there is differential-linear distinguishing attack on ICEPOLE with time and data complexity less than 2^{36}. Based on the differential-linear properties, we propose the state recovery attacks on all the three variants of the ICEPOLE family. The results show that it is possible to recover the 256 bits unknown state of ICEPOLE–128 and ICEPOLE–128a with practical complexity 2^{46}. And for ICEPOLE–256a, the complexity is 2^{60}. We experimentally verified the state recovery attack on ICEPOLE–128 and ICEPOLE–128a, and the unknown state can be recovered in a few days using a 64 cores server.

1 The ICEPOLE Authenticated Cipher

The ICEPOLE family of authenticated ciphers includes three variants according to the parameters: ICEPOLE–128, ICEPOLE–128a and ICEPOLE–256a. The one with 128-bit secret message number, 128-bit key and 128-bit nonce is named ICEPOLE–128. And the other two variants do not have secret message number and are named ICEPOLE–128a and ICEPOLE–256a according to the key length. The length of nonce for these two variants is 96-bit. We will briefly describe the ICEPOLE authenticated cipher. The full specification can be found in [7]. And an overview of ICEPOLE–128 is provided in Figure 1.

1.1 Notations

The ICEPOLE algorithm has 1280-bit internal state S. It uses the little-endian convention. The organization of internal state is similar to Keccak [1], which uses the 3 dimensions structure. Therefore, the 1280-bit state S can be represented as $S[4][5][64]$, or shortly $S[4][5]$, an array of 64-bit words. For $S[x][y][z]$, it is corresponding to the $64(x+4y)+z$-th bit of the input. ICEPOLE uses 4×5 *slice*. In each slice, it has 4 *rows* and 5 *columns*. And $S_{\lfloor n \rfloor}$ denotes the first n bits of the state.

Following notations are used in the ICEPOLE specification: $\oplus$ is the bitwise XOR, $\cdot$ is the bitwise AND, $\neg$ is the negation.

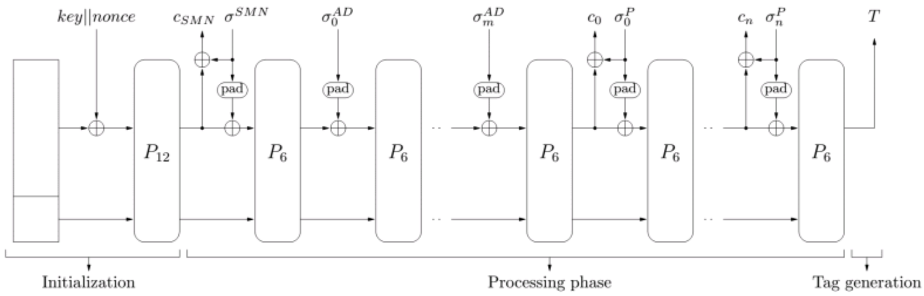

Fig. 1. General scheme of ICEPOLE encryption and authentication (Fig. 1. [7])

1.2 The ICEPOLE permutation P

The permutation P is applied iteratively on the ICEPOLE state S during the encryption and authentication. Each permutation lay is called a *round*, R. The 6–and 12–round of P are represented as P_6 and P_{12} respectively. Each round includes five operations: μ, ρ, π, ψ, and κ.

$$R = \kappa \circ \psi \circ \pi \circ \rho \circ \mu$$

The operations are defined as follows:

μ**:**

A column vector (Z_0, Z_1, Z_2, Z_3) is multiple by a constant matrix producing a vector of four 5-bit words.

$$\begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 1 & 18 & 2 \\ 1 & 2 & 1 & 18 \\ 1 & 18 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \end{bmatrix} = \begin{bmatrix} 2Z_0 + Z_1 + Z_2 + Z_3 \\ Z_0 + Z_1 + 18Z_2 + 2Z_3 \\ Z_0 + 2Z_1 + Z_2 + 18Z_3 \\ Z_0 + 18Z_1 + 2Z_2 + Z3 \end{bmatrix}$$

The operations are done in $GF(2^5)$. The irreducible polynomial $x^5 + x^2 + 1$ is used for the filed multiplication. And μ can be efficiently implemented with simple bitwise equations, see Appendix B in [7].

ρ**:**

The ρ step is the bitwise rotation on each of the 20 64-bit words. It is defined as:

$$\rho(S[x][y]) = S[x][y] \lll \text{offsets}[x][y] \qquad \text{for all}(0 \leq x \leq 3),\ (0 \leq y \leq 4)$$

The rotation offsets are as follows:

offset[0][0] := 0	offset[0][1] := 36	offset[0][2] := 3	offset[0][3] := 41
offset[0][4] := 18	offset[1][0] := 1	offset[1][1] := 44	offset[1][2] := 10
offset[1][3] := 45	offset[1][4] := 2	offset[2][0] := 62	offset[2][1] := 6
offset[2][2] := 43	offset[2][3] := 15	offset[2][4] := 61	offset[3][0] := 28
offset[3][1] := 55	offset[3][2] := 25	offset[3][3] := 21	offset[3][4] := 56

π**:**

π reorders the bits within each slice. It maps $S[x][y]$ to $S[x'][y']$ using following rule:

- $x' := (x + y) \mod 4$
- $y' := (((x + y) \mod 4) + y + 1) \mod 5$

ϕ**:**

ϕ is the S-box layer. ICEPOLE uses following 5-bit S-box:

$$(31, 9, 18, 11, 5, 12, 22, 15, 10, 3, 24, 1, 13, 4, 30, 7,\\ 20, 21, 6, 23, 17, 16, 2, 19, 26, 27, 8, 25, 29, 28, 14, 0)$$

The ϕ operation can be implemented with following bitwise equation. For 5-bit input $(M_0, M_1, \ldots, M_4)$ and 5-bit output $(Z_0, Z_1, \ldots, Z_4)$, let operations on the index k be done modulo 5. The bitwise AND operator $\cdot$ is omitted for clarity.

for all $(0 \leq k \leq 4)$
$$Z_k = M_k \oplus (\neg M_{k+1} M_{k+2}) \oplus (M_0 M_1 M_2 M_3 M_4) \oplus (\neg M_0 \neg M_1 \neg M_2 \neg M_3 \neg M_4)$$

κ**:**

In κ the 64-bit constant is xored with $S[0][0]$. The constants are different for each round:

constant[0] := 0091A2B3C4D5E6F7	constant[1] := 0048D159E26AF37B
constant[2] := 002468ACF13579BD	constant[3] := 00123456F89ABCDE
constant[4] := 00091A2BFC4D5E6F	constant[5] := 00048D15FE26AF37
constant[6] := 0002468AFF13579B	constant[7] := 000123457F89ABCD
constant[8] := 000091A2BFC4D5E6	constant[9] := 000048D1DFE26AF3
constant[10] := 00002468EFF13579	constant[11] := 00001234F7F89ABC

1.3 Initialization

First, the state S is initialized with 1280-bit constant:

S[0][0] := 0XFF97A42D7F8E6FD4	S[0][1] := 0X90FEE5A0A44647C4
S[0][2] := 0X8C5BDA0CD6192E76	S[0][3] := 0XAD30A6F71B19059C
S[0][4] := 0X30935AB7D08FFC64	S[1][0] := 0XEB5AA93F2317D635
S[1][1] := 0XA9A6E6260D712103	S[1][2] := 0X81A57C16DBCF 555F
S[1][3] := 0X43B831CD0347C826	S[1][4] := 0X01F22F1A11A5569F
S[2][0] := 0X05E5635A21D9AE61	S[2][1] := 0X64BEFEF28CC970F2
S[2][2] := 0X613670957BC46611	S[2][3] := 0XB87C5A554FD00ECB
S[2][4] := 0X8C3EE88A1CCF32C8	S[3][0] := 0X940C7922AE3A2614
S[3][1] := 0X1841F924A2C509E4	S[3][2] := 0X16F53526E70465C2
S[3][3] := 0X75F644E97F30A13B	S[3][4] := 0XEAF1FF7B5CECA249

Then, the key K and the *nonce* are introduced to the state. The *nonce* is 128-bit for ICEPOLE–128will be used without padding. And the 96-bit *nonce* for ICEPOLE–128a and ICEPOLE–256a will be padded with 32 zeros to form a 128-bit *nonce*. $nonce_0$ and $nonce_1$ denote two 64-bit words of the padded nonce.

For ICEPOLE–128 and ICEPOLE–128a, K_0 and K_1 denote two 64-bit words of the key,

$$
\begin{aligned}
S[0][0] &:= S[0][0] \oplus K_0 \\
S[1][0] &:= S[1][0] \oplus K_1 \\
S[2][0] &:= S[2][0] \oplus nonce_0 \\
S[3][0] &:= S[3][0] \oplus nonce_1
\end{aligned}
$$

For ICEPOLE–256a, K_0, K_1, K_2 and K_3 denote four 64-bit words of the key,

$$
\begin{aligned}
S[0][0] &:= S[0][0] \oplus K_0 \\
S[1][0] &:= S[1][0] \oplus K_1 \\
S[2][0] &:= S[2][0] \oplus K_2 \\
S[3][0] &:= S[3][0] \oplus K_3 \\
S[0][1] &:= S[0][1] \oplus nonce_0 \\
S[1][1] &:= S[1][1] \oplus nonce_1
\end{aligned}
$$

Then, the P_{12} permutation is run on the state S.

1.4 Processing associated data and plaintext

ICEPOLE–128 uses 128-bit secret message number σ^{SMN}. It will be processes before associated data and the plaintext. Unlike the ICEPOLE–128, ICEPOLE–128a and ICEPOLE–256a do not have secret message number (SMN). Only the associated data blocks σ_i^{AD} and the plaintext σ_i^P will be processed.

For ICEPOLE–128 and ICEPOLE–128a, the length of the block σ^{SMN} is 128-bit, and the length of σ_i^{AD} and σ_i^P is in the range $[0, 1024]$ bits. And they will be padded to 1026 bits using following rule. First, a *frame bit* is appended. It is set to '1' for the last σ^{AD} block and all σ_i^P except the last one. For all other blocks, it is set to '0'. Then, a bit '1' is appended, following by '0's to make the length of padded block be 1026-bit. The number of blocks under a single key is less than 2^{126}.

For ICEPOLE–256a, the associated data and plaintext blocks have length in the range $[0, 960]$. The same padding rule is applied and the padded blocks have length 962-bit. The number of blocks under a single key is less than 2^{62}.

The process of secret message number is as below for ICEPOLE–128:

$c_{SMN} = S_{\lfloor 128 \rfloor} \oplus \sigma^{SMN}$
$\sigma^{SMN} := pad(\sigma^{SMN})$
$S_{\lfloor 1026 \rfloor} := S_{\lfloor 1026 \rfloor} \oplus \sigma^{SMN}$
$S := P_6(S)$

The process of associated data and plaintext blocks is as below:

for all blocks σ_i^{AD}{
  $\sigma_i^{AD} := pad(\sigma_i^{AD})$
  $S_{\lfloor 1026 \rfloor} := S_{\lfloor 1026 \rfloor} \oplus \sigma_i^{AD}$
  $S := P_6(S)$
}

for all blocks σ_i^P{
  $c_i := S_{\lfloor l \rfloor} \oplus \sigma_i^P$ (l is the length of σ_i^P)
  $\sigma_i^P := pad(\sigma_i^P)$
  $S_{\lfloor 1026 \rfloor} := S_{\lfloor 1026 \rfloor} \oplus \sigma_i^P$
  $S := P_6(S)$
}

1.5 Tag generation

After the AD and P are processed, the 128-bit tag T is derived: (T_0 and T_1 are two 64-bit words of T).

$$T_0 := S[0][0]$$
$$T_1 := S[0][1]$$

The decryption and verification is trivial and we omit it here.

1.6 Security goals of ICEPOLE

The main security goals of ICPEPOLE are: 128-bit encryption security for ICEPOLE–128 and ICEPOLE–128a; 256-bit encryption security for ICEPOLE–256a; and 128-bit authentication security for all variants.

An important property of ICEPOLE is that the intermediate level of robustness under nonce-misuse circumstance. It claimed that

1. "*in the case of nonce reuse the key-recovery attack is not possible*".
2. "*Authenticity (integrity) in the duplex construction does not need a nonce requirement, thus is not threatened by nonce reuse.*

2 Differential-Linear Distinguishing Attack on ICEPOLE

In [7], the designers made several initial cryptanalysis on ICEPOLE, including the differential cryptanalysis, linear cryptanalysis and rotational cryptanalysis. In this section, we will revisit the cryptanalysis in [7] and present our results on ICEPOLE based on the differential-linear cryptanalysis.

Before stating the attack, we want to highlight that our attacks only work under the assumption that the *nonce* and *secret message number* can be reused. Otherwise, the differential analysis is impossible.

The differential-linear cryptanalysis is the combination of differential cryptanalysis [3] and linear cryptanalysis [6]. It was introduced by Langford and Hellman in [4] in 1994 to attack the DES, and later Biham, Dunkelman and Keller gave an enhanced version of this method [2]. The idea is to use messages with certain input difference and analyze the statistics of the differences of chosen bits (according to the linear mask) in the output. When the XORed differences of the chosen bits have a significant bias from 0.5, the adversary can distinguish the cipher from a random permutation. In [5], Lu studied the implicit assumptions made in [4] and [2] and gave a theorem to compute the probability for the differential-linear distinguisher under the original two assumptions:

1. The involved round functions behave independently.
2. The two inputs $E_0(P)$ and $E_0(P \oplus \alpha)$ of the linear characteristic for E_1 behave as independent inputs with respect to the linear characteristic, where E_0 is the encryption for the differential rounds and E_1 is the encryption for the linear rounds, P is the plaintext and α is the input difference.

The theorem says when we divide the encryption into two steps (differential and linear), let $\hat{p}$ be probability that the input of the linear mask has no XORed difference after the differential step, let ϵ be the linear characteristic bias for the linear step, then the probability that the XORed difference of the output linear mask is 0 is $1/2 + 2(2\hat{p} - 1)\epsilon^2$.

In the analysis on ICEPOLE, we first divide the encryption P_6 into two steps with equal number of rounds. So the first 3 rounds will be the differential step and the last 3 rounds will be the linear step.

Then, our task is to find good 3-round differential characteristics and 3-round linear characteristics. Unless otherwise specified, we are discussing the ICEPOLE–128 and ICEPOLE–128a in this section under the nonce-misuse assumption. The ICEPOLE–256a will be discussed later.

2.1 Finding the differential characteristics

ICEPOLE uses S-box as the only non-linear operation, and the maximum differential probability of the S-box is 2^{-2}. The differential probability is largely determined by the number of active S-boxes. Although the designers expected that only 3% of the difference transitions has the maximum probability 2^{-2}, this probability is not rare in the early rounds. Because most of the active S-boxes have 1 bit input difference after the diffusion. And the differential probability of a single ICEPOLE S-box is 2^{-2} when the input and output differences are identical with weight 1. Those 1-to-1 identical difference transitions are preferred to the 1-to-n ($n \geq 2$) cases even with the same probability as they will propagate to less number of active S-boxes in the next round.

In [7], the designers analyzed the minimum number of active S-boxes using SAT solver and found that the minimum number is 9 for 3 rounds ICEPOLE. There is an intuitive way to construct the differential characteristics reach this lower bound. We can place 1 active S-box in the middle round and let it propagate backward and forward for one round. Then the number of active S-boxes will be in the form of 4–1–4, which reaches the minimum number 9.

However, the above 3 round differential path is not feasible. In fact, only at most 1024 bits(for ICEPOLE–256a, 960 bits) out of the 1280 bits in a state can be affected by the plaintext and are feasible to introduce difference. Thus, we have the following observation on the operation μ.

Observation 1 *For any 20-bit slice, when the output difference is one bit after μ, there is at least one bit input difference at the last column ($S[\cdot][4]$).*

This observation can be easily verified by analyzing the inverse operation of μ.

Therefore, when an active S-box is propagated backward, it is likely that a number of active bits will be propagated to the last column of the input state, which is infeasible to introduce. And we programed to verified that it is impossible to find such feasible 4–1–4 differential path.

It implies that for any feasible differential characteristics of ICEPOLE, the minimum number of active S-boxes is 2 in the first round. As a result, we will consider the differential characteristics with two active S-boxes at the first round. The ideal case is that the output difference of each active S-box is only 1-bit. Then after μ, this 1-bit difference in a slice is propagated to 4 bits. So we expect the good differential characteristics will have 8 active S-boxes in round 2 and no more than 32 active S-boxes in round 3.

We searched for good 3-round differential characteristics, and found 5 possible initial differences in Table 1 (the rotated differences on the 64-bit word are not considered here), which can lead to feasible 3-round differential characteristics. We will name them as D_1, D_2, D_3, D_4 and D_5.

The total number of 3-round active S-boxes is 42, which implies the differential probability is at most 2^{-84}. But in fact, we do not need any specific differential characteristics. What we need is the input linear mask does not have XORed difference at the beginning of round 4. As long as the number of active S-boxes is relatively low (note that 32 is only 1/8 of the total 256 S-boxes), there is a high probability for the differential step.

(a) Initial difference 1

0x0	0x0	0x1	0x0	0x0
0x1	0x0	0x1	0x0	0x0
0x1	0x1	0x1	0x1	0x0
0x0	0x1	0x0	0x0	0x0

(b) Initial difference 2

0x0	0x0	0x1	0x0	0x0
0x1	0x1	0x1	0x1	0x0
0x0	0x1	0x0	0x1	0x0
0x1	0x0	0x1	0x0	0x0

(c) Initial difference 3

0x1	0x1	0x0	0x0	0x0
0x1	0x0	0x0	0x1	0x0
0x0	0x0	0x0	0x1	0x0
0x1	0x1	0x1	0x0	0x0

(d) Initial difference 4

0x0	0x0	0x0	0x0	0x0
0x1	0x0	0x1	0x0	0x0
0x1	0x1	0x1	0x1	0x0
0x0	0x1	0x0	0x1	0x0

(e) Initial difference 5

0x0	0x0	0x0	0x0	0x0
0x0	0x1	0x0	0x1	0x0
0x1	0x0	0x1	0x0	0x0
0x1	0x1	0x1	0x1	0x0

Table 1. The initial differences. Each entry represents a 64-bit word.

2.2 Finding the linear characteristic

The bias of an ICEPOLE linear characteristic is determined by S-boxes involved in the linear characteristic. We use following method to find the 3-round linear characteristics.

Take a single bit at both the input and output masks of an S-box in the middle round. Then find related bits in the input of the first round (*input linear mask*) and the output of the third round before S-box (*output linear mask*), assuming that the S-boxes are identical mappings. Note that the ICEPOLE S-box has maximum bias 2^{-2} for linear characteristic. And all the 1-bit identical mappings have bias 3/16 which is nearly optimal.

The requirement for the input and output masks is low-weight. When the input linear mask has lower weight, the probability that the input linear mask does not have XORed difference after the 3-round differential step will be higher. When the output linear mask has lower weight, less number of round 6 S-boxes will be involved in the linear relation.

The reason is that we want the weight of the input mask to be as small as possible, so as to increase the probability that the input linear mask does not have XORed difference after the 3-round differential step.

Two good linear characteristics we found are given in Table 2. They are denoted as L_1 and L_2.

2.3 Observations to improve the attack

First, the following observation will be helpful in the selection of the differential characteristics.

Observation 2 *When the 1024 bits of an ICEPOLE state S are known, or equivalently, $S[0..3][0..3]$ are known, we are able to determine $S[0][2]$, $S[1][3]$, $S[2][4]$, $S[3][0]$, $S[3][2]$ after μ, ρ and π operations are performed on S.*

(a) Linear characteristic 1, input linear mask

0x0	0x800000000000000	0x2000000000	0x20000	0x800000000000040
0x40	0x0	0x800000000000000	0x2000020000	0x0
0x0	0x2000000000	0x800000000000000	0x40	0x2000020000
0x0	0x0	0x800002000020000	0x0	0x40

(b) Linear characteristic 1, output linear mask

0x40000	0x1	0x0	0x80000000000	0x0
0x0	0x4	0x0	0x0	0x0
0x0	0x200000	0x2000000000000000	0x0	0x0
0x0	0x0	0x20000000000	0x100000000000000	0x0

(c) Linear characteristic 2, input linear mask

0x120000004000000	0x8000000000000000	0x100000000000	0x0	0x0
0x8020000000000000	0x4000000	0x8000100000000000	0x0	0x100000000000000
0x8100000000000000	0x20000000000000	0x0	0x100000000000	0x4000000
0x4000000	0x8100100000000000	0x0	0x0	0x20100000000000

(d) Linear characteristic 2, output linear mask

0x40000	0x1	0x0	0x0	0x0
0x8000	0x4	0x0	0x0	0x0
0x0	0x0	0x2000000000000000	0x0	0x0
0x0	0x400	0x20000000000	0x100000000000000	0x0

Table 2. The linear characteristics, assuming S-boxes are identical mappings

According to the above observation, we are able to determine the values of certain input bits to the S-boxes in the first round. Thus, we can update the differential tables for the S-boxes with fixed input values.

The following example shows how this will help to improve the differential probability. Suppose the input of an S-box is in the form "$1 * 0 * *$", where the '$*$'s are the unknown bits and the '1' and '0' are the bits with fixed values, and the input difference is 2, then the output difference is 2 with probability 1, which is larger than the 0.25 in the general case. To get the fixed values, we can manipulate the input bits according to the mask in Table 3.

0x0	0x800000010	0x200000001	0x0	0x0
0x800000010	0x0	0x0	0x200000001	0x0
0x0	0x800000010	0x0	0x200000001	0x0
0x800000010	0x0	0x800000010	0x200000001	0x0

Table 3. Input mask for the fixed bits in round 1.

By setting the bits in $S[0][1]$ selected by the mask $0x800000010$ as '1' and all the other bits selected by the other masks as '0', the two active S-boxes in round 1 will satisfy the above condition on the input values.

Our next observation is about the S-box in the round 6 of P_6. Since the ciphertext is generated by XORing the keystream and the plaintext, we can obtain the values of at most 1024 bits in the state after 6 rounds. Hence, for ICEPOLE–128 and ICEPOLE–128a, 4 of the 5 bits in the output of all the S-boxes are known. And for ICEPOLE–256a, 4 of the 5 bits for 192 S-boxes and 3 of the 5 bits for 64 S-boxes in the output are known.

It is possible to recover some of the input bits from the output bits of the S-boxes. Table 4 gives the probability that we can recover a bit at each position of the S-box input.

Position	0	1	2	3	4
Probability	$\frac{6}{8}$	$\frac{5}{8}$	$\frac{4}{8}$	$\frac{4}{8}$	$\frac{1}{8}$

Table 4. Probability that the input bits can be recovered for an S-box at round 6.

Therefore, instead of using the bias of the linear relation in the round 6, we can recover the chosen input bits in round 6 before S-box, by enumerating the values of output bits. And thus compute the value of the bit in the linear relation in the output of round 5.

For example, if we consider the output linear mask of L_1, the probability that we can recover the 8 bits can be computed as

$$Pr_{L_1} = (3/4) \times (5/8)^3 \times (1/2)^4 = 2^{-6.45}.$$

And using these 8 bits, we can compute the value of bit $S[1][1][0]$ at the output of round 5.

Similarly, the probability for L_2 is

$$Pr_{L_2} = (3/4)^2 \times (5/8)^3 \times (1/2)^3 = 2^{-5.86}.$$

2.4 Concatenating the differential and linear characteristics

After we analyzed the 3-round differential characteristics and 3-round linear characteristics, we can concatenate them to form 6-round differential-linear characteristics. And this will lead to a 6-round distinguisher for ICEPOLE.

First, we choose the initial difference D_2 because the two active S-boxes in round 1 are in the last row which has two known bits. The difference of state before the S-box for D_2 is given in Table 5.

To choose the 3-round linear characteristic, we consider all the possible rotations of the two linear characteristics L_1 and L_2 given in Section 2.2. There are 128 possible rotated linear characteristics. The selection of linear characteristic is done experimentally:

0x0	0x0	0x0	0x0	0x0
0x0	0x0	0x0	0x0	0x0
0x0	0x0	0x0	0x0	0x0
0x0	0x400	0x20000000000	0x0	0x0

Table 5. The difference of state before the first S-box in D_2.

1. Randomly pick 1024-bit plaintext blocks pairs with the chosen initial difference D_2 and the fix values (1, 0) for the two known bits at positions 0 and 2 in the round 1 active S-boxes.
2. After 5 rounds, verify whether the XORed difference of the states under the linear characteristic is zero or not. If it is zero, add it to a counter $cntSame$.
3. Repeat the above process, and choose the one with highest bias at $cntSame$ from 0.5.

For D_2, the highest bias is $2^{-9.2}$ and the linear characteristic is L_1 left rotated by 33 bits.

We remark that the experimentally determined bias is significantly higher than the one computed using the Theorem 1 in [5]. The reason may due to the assumption 2 does not hold in this case. Namely, the correlation between two linear steps can not be ignored. Therefore, we use the experimental results here.

From the bias above, we immediately get a distinguishing attack on ICEPOLE.

1. Generate $2^{33.9}$ pairs of two-block plaintext such that the first 1024-bit plaintext block has initial difference D_2 and the fixed values as specified above. All the other bits are random.
2. Use ICEPOLE to encrypt the plaintext blocks and then decide the 1024 bits input and output state of the first P_6. Discard those pairs if the two bits in round 5 output in the linear characteristic L_1 left rotated 33 bits cannot be recovered. Note that for each bit, the probability is $2^{-6.45}$ to recover. So there are 2^{21} pairs left. Then compute the bit at position $S[1][1][33]$ of the output of round 5 using the recovered bits from round 6.
3. Analyze the bias of the XORed difference of those two bits ($S[1][1][33]$). If the bias is larger than $2^{-10.2}$ we conclude that it is the ICEPOLE encryption.

The success probability is computed by using the normal distribution to approximate the binomial distribution of the bias. For ICEPOLE, the bias is a random variable $X \sim N(n(1/2 + 2^{-9.2}), n(1/2 + 2^{-9.2})(1/2 - 2^{-9.2}))$, where n is the number of pairs of the recovered 5 rounds output. When $n = 2^{21}$, the probability that $X \geq 2^{-10.2}$ is 99.3%. Assuming for a random permutation, the bias is a random variable $Y \sim N(n/2, n/4)$. The value is larger than $2^{-10.5}$ with probability 0.7%. Hence, we have a very good chance to distinguish the ICEPOLE encryption from a random permutation by using $2^{35.9}$ plaintext blocks (a pair of two-block messages are counted as 4 plaintext blocks), assuming the nonce can be reused.

3 State Recovery Attack on ICEPOLE

In this section, we will use the differential-linear characteristics to launch state recovery attacks on ICEPOLE.

3.1 State recovery attacks on ICEPOLE–128 and ICEPOLE–128a

For ICEPOLE–128 and ICEPOLE–128a, there are 256 unknown bits in the state before P_6. They are in the last column of each slice. For convenience, we denote those four 64-bit unknown words in the last column as $\{U_0, U_1, U_2, U_3\}$ according to the row index. And We will recover them step by step.

3.1.1 Recovering U_0 and U_3

To recover the unknown bits in the state, we will analyze the input values at the active S-boxes. For D_2, there are two active S-boxes in round 1. One has input difference 2 and the other has input difference 4. We will focus on the one with input difference 4 and denote the five input bits as $b_0, b_1, ..., b_4$ according to their positions (b_0 is the least significant bit).

When the two bits of an ICEPOLE S-box are fixed with values $b_0 = 1$ and $b_2 = 0$, there are 8 possible values for the rest three bits. When the input difference is 4 (at b_2), the output difference have weight 1 only when $b_1 = 1$ and $b_3 = 0$. Intuitively, the lower the weight of the output difference after the first round, the higher the probability that there is no XORed difference at the output linear mask after 5 rounds. Hence, it is possible to relate the input value of the active S-box in round 1 to the bias of XORed output difference in round 5.

We experimentally find the following biases for different values of the input bit b_1 and b_3.

values	bias (log based 2)
$b_1 = 0$, $b_3 = 0$	-13.0
$b_1 = 1$, $b_3 = 0$	-7.3
$b_1 = 0$, $b_3 = 1$	-13.9
$b_1 = 1$, $b_3 = 1$	-11.9

Note that b_1 is related to an unknown bit in U_3, and b_3 is related to two unknown bits, in U_0 and U_3. So we have following relations:

$$b_1 = U_3^{31} \oplus a_0$$

and

$$b_3 = U_0^{49} \oplus U_3^{49} \oplus a_1,$$

where U^x is the x-th bit in the 64-bit word U; a_0 and a_1 are constants can be computed from the 1024-known bits.

We describe the state recovery process given as below:

1. Generate $2^{33.9}$ pairs of two-block plaintext satisfied following requirements. The first block of the plaintext has difference D_2 and each active S-box has fixed values '1' and '0' in bit 0 and 2 respectively. All the other bits are random.
2. Use ICEPOLE to encrypt the plaintext blocks and then decide the 1024 bits input and output state of the first P_6. Discard the pairs if the two bits in round 5 output in the linear characteristic L_1 left rotated 33 bits cannot be recovered. There are 2^{21} pairs left. Then compute the bit at position $S[1][1][33]$ of the output of round 5 using the recovered bits from round 6.
3. If the two bits at the position $S[1][1][33]$ are the same, we compute the values of a_0 and a_1 according to the input and increase the counter for the value of (a_0, a_1) by 1.
4. Suppose the largest counter of (a_0, a_1) takes value $a_0 = v_0$ and $a_1 = v_1$, we guess that $U_3^{31} = v_0 \oplus 1$ and $U_0^{49} \oplus U_3^{49} = v_1$.
5. By rotating the differential-linear characteristic for the other 63 bits, we can recover the two 64-bit unknown words U_0 and U_3.

To access the success probability, it is equivalent to compute the probability that a random variable $X \sim N(n(1/2 + 2^{-7.3}), n(1/2 + 2^{-7.3})(1/2 - 2^{-7.3}))$ has value greater than the random variable $Y \sim N(n(1/2 + 2^{-11.9}), n(1/2 + 2^{-11.9})(1/2 - 2^{-11.9}))$. When $n = 2^{19}$, the probability is almost 1. Since we have 2^{21} pairs of input, each of the four choices of b_0 and b_1 will be around 2^{19} pairs.

We remark that here the probability is high enough to make sure that even if the experiment is repeated for 64 times, the success probability is still close to 1.

3.1.2 Recovering U_2

Assuming that U_0 and U_3 have been recovered correctly, we can use similar method to recover U_2. In this case, we use D_1 and L_2 left rotated by 58 bits as the differential-linear characteristic.

The difference of state before the first round S-box for D_1 is given in the Table 6.

0x0	0x0	0x0	0x0	0x0
0x0	0x4	0x0	0x0	0x0
0x0	0x200000	0x0	0x0	0x0
0x0	0x0	0x0	0x0	0x0

Table 6. The difference of state before the first S-box in D_2.

Fixed bits: With the knowledge of U_0 and U_3, we fix bit 0, 2, 3 with the values (1, 0, 0) respectively for the active S-box at the second row. This is to ensure the output difference of this active S-box has weight exact 1. And we fix bit 0, 4 with the values (1, 1) for the active S-box at the third row. This is to make the weight of output difference can be distinguished from the input value of bit 2, which is denoted as b_2.

We experimentally find the following biases for b_2 after 5 rounds. The biases are base on the difference of the output bit at position $S[3][1][58]$.

values	bias (log based 2)
$b_2 = 0$	-11.0
$b_2 = 1$	-15.4

From the active S-box at the third round, it is possible to find following relation:

$$b_2 = U_2^{24} \oplus a,$$

where a is a constant can be computed from the known input bits.
The state recovery process process:

1. Generate $2^{36.7}$ pairs of two-block plaintext satisfied following requirements. The first block of the plaintext has difference D_1 and each active S-box has fixed values according to the above paragraph. All the other bits are random.
2. Use ICEPOLE to encrypt the plaintext blocks and then decide the 1024 bits input and output state of the first P_6. Discard the pairs if the two bits in the linear relation (according to L_2 rotated by 58 bits) in the output of round 5 cannot be recovered. There are 2^{25} pairs left. Then compute the bit at position $S[3][1][58]$ of the output of round 5 using the recovered bits from round 6.
3. If the two bits at the position $S[3][1][58]$ are the same, we compute the value of a according to the input and increase the counter for that value by 1.
4. Suppose the largest counter of a take value $a = v$, we guess that $U_1^0 = v$.
5. By rotating the differential-linear characteristic for the other 63 bits, we can recover the 64-bit unknown word U_1.

The estimated success probability is 99.6% for each bit.

3.1.3 Recovering U_1

At this stage, we assume that U_0, U_2 and U_3 have been recovered correctly. In this case, we select D_3 and L_2 left rotated by 35 bits as the differential-linear characteristic.

The differential of state before the first round S-box for D_3 is given in the Table 7.

0x0	0x0	0x8000000000000	0x0	0x0
0x8000	0x0	0x0	0x0	0x0
0x0	0x0	0x0	0x0	0x0
0x0	0x0	0x0	0x0	0x0

Table 7. The differential of state before the first S-box in D_3.

Fixed bits: We fix bit 1, 2, 4 with the values (1, 0, 1) for the active S-box at the first row. It is to ensure that the weight of output difference of this active S-box is determined by the value of input bit 3 which is denoted as $b_{0,3}$. And we fix bit 0, 2, 3, 4 with the values (0, 1, 1, 1) for the active S-box at the second row. It is to ensure that the weight of output difference is determined by the value of input bit 1 which is denoted as $b_{1,1}$.

The $b_{0,3}$ and $b_{1,1}$ are related to the unknown bits:

$$b_{0,3} = a_0 \oplus U_1^{12}$$
$$b_{1,1} = a_1 \oplus U_1^{13}$$

where a_0 and a_1 are constants can be computed from the known input.

We experimentally find the following biases for different values of the input bit $b_{0,3}$ and $b_{1,1}$. The biases are base on the difference of the output bit at position $S[3][1][35]$.

values	bias (log based 2)
$b_{0,3} = 0,\ b_{1,1} = 0$	-11.2
$b_{0,3} = 1,\ b_{1,1} = 0$	-15.2
$b_{0,3} = 0,\ b_{1,1} = 1$	-16.4
$b_{0,3} = 1,\ b_{1,1} = 1$	-14.8

We remark that the biases other than the first row in above table may not be very accurate because that they are quite small numbers.

The state recovery process process:

1. Generate $2^{37.7}$ pairs of two-block plaintext satisfied following requirements. The first block of the plaintext has difference D_3 and each active S-box has fixed values according to the above paragraph. All the other bits are random.
2. Use ICEPOLE to encrypt the plaintext blocks and then decide the 1024 bits input and output state of the first P_6. Discard those pairs if the two bits in the linear relation (according to L_2 rotated by 35 bits) in the output of round 5 cannot be recovered. There are 2^{26} pairs left. Then compute the bit at position $S[3][1][35]$ of the output of round 5 using the recovered bits from round 6.
3. If the two bits at the position $S[3][1][35]$ are the same, we compute the value of a_0 and a_1 according to the input and increase the counter for (a_0, a_1) by 1.

4. Suppose the largest counter of (a_0, a_1) take value $a_0 = v_0$ and $a_1 = v_1$, we guess that $U_1^{12} = v_0$ and $U_1^{13} = v_1$.
5. By rotating the differential-linear characteristic for the other 31 even bits less than 64, we can recover the 64-bit unknown word U_1.

Note that in each rotation of the differential-linear characteristic, we are able to recover two consecutive bits, so we only need to test 32 rotations to recover the 64-bit U_1. The estimated success probability is 98.7% for each bit.

3.1.4 Correcting the recovered state

In the previous state recovery attack, only the first 128 bits can be correctly recovered with probability almost 1. For the other 128 bits, although the success probability is around 99%, it is still possible that the state we recovered has some error bits. Whether the state is correct can be easily verified by encryption some new messages and compare the ciphertext.

We can correct up to 7 error bits with relatively low complexity. To correct i error bits is to choose any i bits from the 128 bits and flip the values. Then test whether the modified unknown state is correct.

Suppose that for the 128 bits U_1 and U_2, the probability that each bit is correct is 0.99, we can compute the probability that the number of error bits is less than 8 as

$$\sum_{i=0}^{7} \binom{128}{i} \times .99^{128-i} \times .01^{i} = 0.99995.$$

The total number of encryptions to correct up to 7 error bits is $2^{37.5}$, which is negligible to the whole attack.

3.1.5 Summary of the attack

The data complexity is:

- U_0 and U_3: $2 \times 2 \times 2^{33.9} \times 2^6$. The first 2 is from the two blocks and the second 2 is from the pair of messages.
- U_2: $2 \times 2 \times 2^{36.7} \times 2^6$
- U_1: $2 \times 2 \times 2^{37.7} \times 2^5$
- Total: $2^{45.8}$

The time complexity is the $2^{45.8}$ encryptions of the one block plaintext and the possible $2^{37.5}$ encryptions for correction.

The memory cost is mainly on the storage of some counters, which is negligible.

The success rate of this attack is close to 1, and can be adjusted through the number of input messages.

3.2 State recovery attacks on ICEPOLE–256a

In the case of ICEPOLE–256a, there are 320 unknown bits in the input and output states in the encryption of a block. In addition to the U_0 to U_3 in the previous subsection, we use U_4 to denote the unknown 64-bit word $S[3][3]$.

An essential difference from the previous analysis of ICEPOLE–128 and ICEPOLE–128a is that in the last row of the output in ICEPOLE–256a, there are 3 known bits instead of 4 known bits. Consequently, it is impossible to recover the input bits given the output bits for that row. To deal with this issue, we have to consider the linear relation on the input mask to the S-box. When the input mask is less than 8, the largest bias is 3/16, but if the input mask is 8, the largest bias is 1/16.

For L_1, there are two mask bits placed in the last row, one is 4 and the other is 8. By Piling up Lemma [6], the bias is $2^{-5.4}$. And we save 2^2 in recovering those two bits.

For L_2, there are three mask bits placed in the last row, 2, 4, and 8. By Piling up Lemma, the bias is $2^{-6.8}$. And the saved computation is $2^{-2.7}$ in recovering those bits.

To increase the bias of the last row, we introduce another linear characteristic L_3, which has only 1 bit in the last row of the input mask of round 6 S-box.

(a) input linear mask

0x0	0x10000000000000	0x20000	0x82000000000	0x400000000000000
0x410000000000000	0x0	0x80000000000	0x20000	0x2000000000
0x400000000000000	0x10000000000000	0x2000000000	0x80000020000	0x0
0x410000000000000	0x0	0x10000000000000	0x2000020000	0x80000000000

(b) output linear mask

0x40000	0x0	0x0	0x80000000000	0x200000000000
0x8000	0x4	0x0	0x0	0x0
0x8	0x200000	0x0	0x0	0x100000000000
0x0	0x0	0x20000000000	0x0	0x0

Table 8. The linear characteristic 3, assuming S-boxes are identical mappings

For L_3, the bias is 3/16 and the probability to recover other bits is $2^{-9.6}$.

To recover the U_4, we use the differential characteristic D_2 with linear characteristic L_1 left rotated 33 bits, same as the recovery of U_0 and U_3. But here we don't known the value of $S[3][2]$, so we will only fix the active bits in $S[3][0]$ as 1.

We will find the value of bit 2 in the input of active S-box related to $S[3][1]$ with difference $0x400$. We denote this bit $b_{3,2}$, and we have $b_{3,2} = a \oplus U_4^{33}$, where a is a constant from the known input.

We experimentally find the following biases for the input bit values $b_{3,2}$ after 5 rounds. The biases are base on the difference of the output bit at position $S[1][1][33]$.

values	bias (log based 2)
$b_{3,2} = 0$	-9.2
$b_{3,2} = 1$	-15.3 (negative)

Considering the linear relation of this XORed value to the 4 bits in the two output states, the bias of the XORed difference of the 4 bits becomes 2^{-18} by Piling up lemma. To make the guess has probability of success close to 1, we need around 2^{40} pairs of two-block plaintext. The total data needed is around $2^{58.7}$.

Next, the complexity of recovering the other unknown words need to be amended. We omit the similar attack process and discuss the estimated complexities here.

For U_0 and U_3, the increased bias is $2^{-8.8}$ which need to be compensated by around $2^{17.6}$ addition messages. And a factor of 2^4 will increase the probability that recover the bits from the round 6 S-box layer. The total effect is the data complexity becomes $2^{55.5}$.

For U_2, D_1 with L_3 left rotated 13 bits will be used and for U_1, D_3 with L_1 left rotated 2 bits will be used. The increased bias is roughly $2^{-2.8}$ for both cases (including the increased 5-round bias). And the decreased probability for recovering the values before round 6 S-box has a factor $2^{-7.5}$. Therefore, the total data complexity for recovering each bit is increased by $2^{13.1}$, which is $2^{57.8}$.

Therefore, the total data complexity of the state recovery attack for is estimated to be $2^{59.8}$, which is less than the constraint 2^{62} by ICEPOLE–256a.

3.3 Implications of the state recovery attacks

For ICEPOLE–128, the state recovery attack implies the failure of encryption if both the nonce and the secret message number are reused. When an adversary have the full knowledge of a state, he can inverse the cipher until the secret message number is injected. Thus, the adversary can decrypt arbitrary plaintext blocks. It also implies a forgery attack on the authentication of plaintext and associated data since the valid tag for any modified associated data or plaintext block can be computed. Since both the key and secret message number are unknown, the adversary cannot recover the key.

For ICEPOLE–128a and ICEPOLE–256a, the state recovery attack implies the whole security is broken if the nonce is reused. The initialization of ICEPOLE–128a is invertible, so the adversary can directly recover the secret key from the known state. Then both the encryption and authentication are insecure.

Summary of the securities of ICEPOLE under our analysis when nonce is reused is in Table 9.

Goal	ICEPOLE-128	ICEPOLE-128a	ICEPOLE-256a
confidentiality for the plaintext	46	46	60
confidentiality for the secret message number	128	-	-
integrity for the plaintext	46	46	60
integrity for the associated data	46	46	60
integrity for the secret message number	46	-	-
integrity for the public message number	46	46	60

Table 9. Number of bits of security when nonce and secret message number can be reused.

4 Experimental Results

We experimentally verified the state recovery attack on ICEPOLE–128a. And we managed to recover the 256 unknown bits practically.

The experiments used the state after ICEPOLE initialization with all zeros IV and key. The unknowns states are:

$U_0 = 0x1e7aed5bfaeb535f$
$U_1 = 0xe0dcc6422595e5ba$
$U_2 = 0x892bf76586876c23$
$U_3 = 0x8b2ef3bf50e902f6$

To recover U_0 and U_3, we run the attack in Sect. 3.1.1 on a server with 48 cores (AMD Opteron(tm) Processor 6172). Instead of checking the constants from input, we used an equivalent method: directly extract the input bits of the active S-box in the first round, and decide whether they are the estimated ones. The number of plaintext pairs we used is 2^{34} for each bit. The attacks takes 15.3 hours and all the 128 bits recovered are correct.

To recover U_2, we run the attack in Sect. 3.1.2 on a server with 64 cores (AMD Opteron(tm) Processor 6276). The number of plaintext pairs we used is 2^{37} for each bit. The attacks takes 3.5 days and all the bits are correct.

To recover U_1, we run the attack in Sect. 3.1.3 on a server with 64 cores (AMD Opteron(tm) Processor 6276). The number of plaintext pairs we used is 2^{38} for each bit. The attacks takes 3.5 days and there is one error bit.

Since the error bit can be easily corrected, the experiments show that our state recovery attack indeed works for ICEPOLE–128a.

5 Conclusion

In this report, we studied the vulnerability of the ICEPOLE authenticated cipher using the differential-linear cryptanalysis. Under the nonce-misuse assumption, we proposed the state recovery attacks on the three variants of ICEPOLE family and show that we can recover the state of ICEPOLE–128 and ICEPOLE–128a with practical complexity, which is around 2^{46}. And the ICEPOLE–256a can be attacked with complexity around 2^{60}.

References

1. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Keccak Sponge Function Family Main Document. Submission to NIST (Round 2), 2009.
2. E. Biham, O. Dunkelman, and N. Keller. Enhancing Differential-Linear Cryptanalysis. In Y. Zheng, editor, *Advances in Cryptology–ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 254–266. Springer Berlin Heidelberg, 2002.
3. E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
4. S. Langford and M. Hellman. Differential-Linear Cryptanalysis. In Y. Desmedt, editor, *Advances in Cryptology–CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 17–25. Springer Berlin Heidelberg, 1994.
5. J. Lu. A Methodology for Differential-Linear Cryptanalysis and Its Applications. In A. Canteaut, editor, *Fast Software Encryption*, volume 7549 of *Lecture Notes in Computer Science*, pages 69–89. Springer Berlin Heidelberg, 2012.
6. M. Matsui. Linear Cryptanalysis Method for DES cipher. In *Advances in Cryptology–EUROCRYPT'93*, pages 386–397. Springer, 1994.
7. P. Morawiecki, K. Gaj, E. Homsirikamol, K. Matusiewicz, J. Pieprzyk, M. Rogawski, M. Srebrny, and M. Wójcik. ICEPOLE v1. submission to CAESAR competition, available from http://competitions.cr.yp.to/round1/icepolev1.pdf.