PANDA v1

Designers and submtters:

Dingfeng Ye, Peng Wang, Lei Hu, Liping Wang,

Yonghong Xie, Siwei Sun, Ping Wang

Institute of Information engineering
Chinese Academy of Sciences

wp@is.ac.cn

March 15, 2014

Contents

1 Introduction **1**

2 Specification of PANDA **3**

2.1 The round function 3

2.2 Specification of PANDA-s 5

2.2.1 The encryption: PANDA-s.Enc$_K(N, A, P)$ 5

2.2.2 The decryption: PANDA-s.Dec$_K(N, A, C, T)$ 6

2.3 Specification of PANDA-b 7

2.3.1 The encryption: PANDA-b.Enc$_K(N, A, P)$ 7

2.3.2 The decryption: PANDA-b.Dec$_K(N, A, C, T)$ 7

3 Security goals **9**

4 Security analysis **11**

4.1 (Non-) Linear transformation in PANDA 11

4.2 Analysis of PANDA-s 12

4.2.1 Linear analysis 12

4.2.2 Differential analysis 15

4.3 Analysis of PANDA-b 17

4.3.1 Differential analysis of iterations of $\mathbf{R}$ 17

4.3.2 Online cipher and PANDA-b 17

5 Features **19**

6 Design rationale **20**

6.1 Objective 20

6.2 Choice of algorithm structure 20

6.3 Choice of linear transformation $\mathbb{A}$ 21

6.4 Assumptions 21

7 Intellectual property **22**

8 Consent **23**

Chapter 1

Introduction

We construct a family of *authenticated ciphers* from scratch. We name it PANDA which consists of two authenticated ciphers: PANDA-s and PANDA-b. Authenticated cipher is a symmetric cryptographic primitive that provides data protections of confidentiality and integrity (authentication) simultaneously. It is also called *authenticated encryption* (AE for short) scheme in the literature.

PANDA is based on a simple round function, like other directly constructed ciphers, such as block ciphers, stream ciphers or hash functions. During the design of PANDA, there are three problems we try to deal with: 1) how to construct a round function; 2) how to use it in our authenticated ciphers; and 3) how to prove the security of our ciphers.

The construction of the round function. The most popular way of constructing round functions is to make use of substitution-permutation networks, i.e., put several S-boxes and a linear transformation *in series* as a round function. This method is broadly used in the design of block cipher and hash function [3] , especially after the success of AES [7]. Following the wide trail strategy in the design of AES [7], the lower bound of active S-boxes in iterations of several round function can be estimated through the *branch number* of the linear transformation. The bound provides an explicit evidence of security against classic attacks, such as differential and linear attacks.

In PANDA, we choose a unusual way to put several S-boxes and a linear transformation *in parallel* as the round function. This method is greatly different from that of SP-networks. For instance, the number of active S-boxes is actually connected to the *minimal polynomial* of the linear transformation. Furthermore the round function and its reverse have similar components and are efficient in both software and hardware.

The way to use the round function. Two different ways result in two authenticated ciphers: PANDA-s and PANDA-b. Similar to AE with sponge structures [2], e.g., FIDES [4], or Helix [11] and Phelix [15], PANDA-s is a mixture of a stream cipher and a MAC. The basic component of PANDA-b is a permeation by 14 iterations of the round function. PANDA-b is also an online cipher like APE [1] but with a different structure.

The security proofs of the ciphers. The security proofs of PANDA are different from that of block cipher modes of operation (e.g., OCB [14], GCM [13]) which give reduction proof with an assumption that the

underlying block cipher is secure. It is also different from that of schemes based on AES round function (e.g., Pelican [8], ACS-1 [12], ALE [5], AEGIS [16]) in which the properties of AES can be used.

To PANDA-s, we measure the security of the stream cipher by linear distinguishing attack, and the security of the MAC by differential attack. As to PANDA-b, the differential property of the underlying permutation is analyzed, and then the permutation is treated as a public random oracle to show the security of PANDA-b as an online cipher.

Chapter 2

Specification of PANDA

PANDA is a family of authenticated ciphers, consisting of two ciphers: PANDA-s and PANDA-b. Both PANDA-s and PANDA-b have two deterministic algorithms: the encryption algorithm Enc and the decryption algorithm Dec. The encryption algorithm Enc takes in a 128-bit key K, a 128-bit nonce N, variable-length associated data A and a variable-length plaintext P and outputs a variable-length ciphertext (C, T) where T is a 128-bit authentication tag. We write it as $\text{Enc}_K(N, A, P) = (C, T)$. The decryption algorithm Dec takes in a tuple (K, N, A, C, T) and outputs P or a special symbol $\perp$ indicating that the ciphertext is invalid. We require that $\text{Dec}_K(N, A, C, T) = P$ if $\text{Enc}_K(N, A, P) = C$.

2.1 The round function

First define a general round function RoundFunc from an 8-block input to an 8-block output, consisting of a linear transformation LinearTrans, and four non-linear transformations SubNibbles. We illustrate it in Fig. 2.1.

More specifically, RoundFunc is defined as follows:

$\texttt{Algorithm RoundFunc}(w, x, y, z, S^{(0)}, S^{(1)}, S^{(2)}, m)$
$w' \leftarrow \texttt{SubNibbles}(w \oplus x \oplus m)$
$x' \leftarrow \texttt{SubNibbles}(x \oplus y)$
$y' \leftarrow \texttt{SubNibbles}(y \oplus z)$
$z' \leftarrow \texttt{SubNibbles}(S^{(0)})$
$(S'^{(0)}, S'^{(1)}, S'^{(2)}) \leftarrow \texttt{LinearTrans}(S^{(0)} \oplus w, S^{(1)}, S^{(2)})$

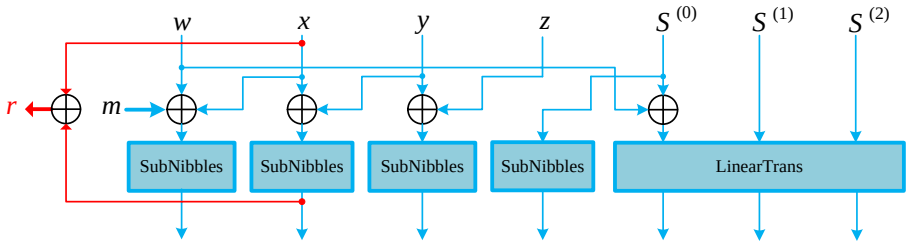


Figure 2.1: The round function in PANDA.

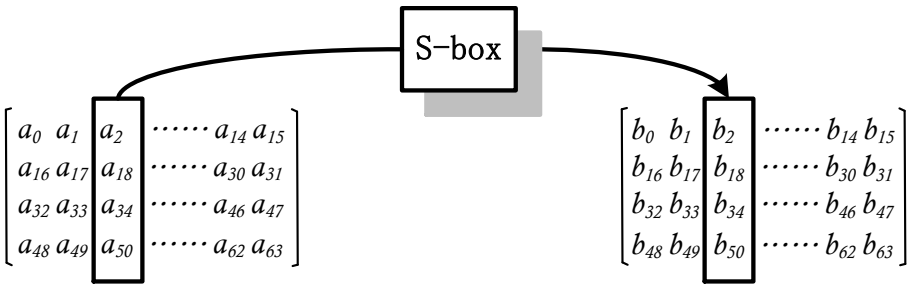


Figure 2.2: SubNibbles acts on the individual columns of its input block.

$r \leftarrow x \oplus x'$
`return` $(w', y', y', z', S'^{(0)}, S'^{(1)}, S'^{(2)}, r)$

The above algorithm is denoted as RoundFunc$(w, x, y, z, S^{(0)}, S^{(1)}, S^{(2)}, m) = (w', y', y', z', S'^{(0)}, S'^{(1)}, S'^{(2)}, r)$, where the first 7 blocks are used as a state in our ciphers, m is a data block, and r is a key stream block. The following expressions are used in our ciphers:

- $(state, r) \leftarrow$ RoundFunc$(state, m)$: use the data block m to update the state $state = (w, x, y, z, S^{(0)}, S^{(1)}, S^{(2)})$, and output the key stream block r.
- $state \leftarrow$ RoundFunc$_s(state, m)$: use a data block m to update the state $state$, without the output of any key stream block.
- RoundPerm$(state)$ = RoundFunc$_s(state, 0)$: a permutation on a state of 7 blocks, which is used in PANDA-b.

Notice that SubNibble and LinearTrans are all permutations, and so the inverse of RoundPerm which is used in PANDA-b can be calculated as the following:

`Algorithm` RoundPerm$^{-1}(w, x, y, z, S^{(0)}, S^{(1)}, S^{(2)})$
$x \leftarrow$SubNibbles$^{-1}(x)$
$y \leftarrow$SubNibbles$^{-1}(y)$
$z \leftarrow$SubNibbles$^{-1}(z)$
$(S^{(0)}, S^{(1)}, S^{(2)}) \leftarrow$LinearTrans$^{-1}(S^{(0)}, S^{(1)}, S^{(2)})$
$w' \leftarrow z \oplus S^{(0)}$
$x' \leftarrow w \oplus w'$
$y' \leftarrow x \oplus x'$
$z' \leftarrow y \oplus y'$
`return` $(w', x', y', z', z, S^{(1)}, S^{(2)})$

SubNibbles. The nonlinear transformation is defined as SubNibbles$(a_0 a_1 \cdots a_{63}) = b_0 b_1 \cdots b_{63}$, where $b_i b_{i+16} b_{i+32} b_{i+48} = S(a_i a_{i+16} a_{i+32} a_{i+48})$, $i = 0, 1, \cdots, 15$. Here a_i, b_i are binary elements, $i = 0, 1, \cdots, 63$. $S(\cdot)$ represents a 4×4 S-box. Fig. 2.2 illustrates the effect of SubNibbles.

The S-box is defined in hexadecimal by the following table.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	0	1	3	2	f	c	9	b	a	6	8	7	5	e	d	4

Obviously SubNibbles is self-reversible, i.e., SubNibbles^{-1} =SubNibbles.

LinearTrans. The linear transformation uses the operations of a finite field. The finite field $\mathbb{F}_{2^{64}}$ is defined by an irreducible polynomial $p(x) = x^{64} + x^{30} + x^{19} + x + 1$, i.e., $\mathbb{F}_{2^{64}} = \mathbb{F}_2(\theta)$ where θ is a root of $p(x)$. The block $a_0 a_1 \cdots a_{63}$ corresponds to $a_0 + a_1\theta + \cdots + a_{62}\theta^{62} + a_{63}\theta^{63} \in \mathbb{F}_{2^{64}}$. The linear transformation LinearTrans is defined as LinearTrans$(S^{(0)}, S^{(1)}, S^{(2)}) = (S^{(0)}, S^{(1)}, S^{(2)})\mathbb{A}$ where the matrix

$$\mathbb{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & \alpha & \alpha+1 \end{pmatrix}^7,$$

where $\alpha = \theta^{32} \in \mathbb{F}_{2^{64}}$.

The reverse of $\mathbb{A}$ is

$$\mathbb{A}^{-1} = \begin{pmatrix} \alpha & \alpha+1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}^7$$

which is used in LinearTrans^{-1} and RoundPerm^{-1}.

2.2 Specification of PANDA-s

All the operations in PANDA-s are defined in terms of 64-bit block. For the variable-length data such as P, C, A in the above notations, we partition them into blocks before the operations. If the last data block is not a full block, we pad it to 64 bits using successive bits of 0. If the original data is a byte string, we use little-endian encoding to make them into a block string and vice versa.

2.2.1 The encryption: PANDA-s.Enc$_K(N, A, P)$

The encryption is divided into the following 4 steps.

1) Initialization: Upload the state $state$ with the nonce $N = (N_0, N_1)$, the key $K = (K_0, K_1)$ and some constants, then run the permutation RoundPerm 14 times, and XOR the key into the state, where N_i and K_i are 64-bit, $i = 0, 1$. More specifically,

$w \leftarrow K_0$
$x \leftarrow N_0$
$y \leftarrow K_1$
$z \leftarrow N_1$
$S^{(0)} \leftarrow C_0 \oplus L(K_0)$
$S^{(1)} \leftarrow C_1 \oplus L(K_1)$
$S^{(2)} \leftarrow C_2 \oplus N_0 \oplus N_1$
`for` $i = 0$ `to` 13
  $state \leftarrow$ `RoundPerm`$(state)$
$w \leftarrow w \oplus K_0$
$x \leftarrow x \oplus K_1$
$y \leftarrow y \oplus K_0$
$z \leftarrow z \oplus K_1$
$S^{(0)} \leftarrow S^{(0)} \oplus K_0$
$S^{(1)} \leftarrow S^{(1)} \oplus K_1$
$S^{(2)} \leftarrow S^{(2)} \oplus K_0$

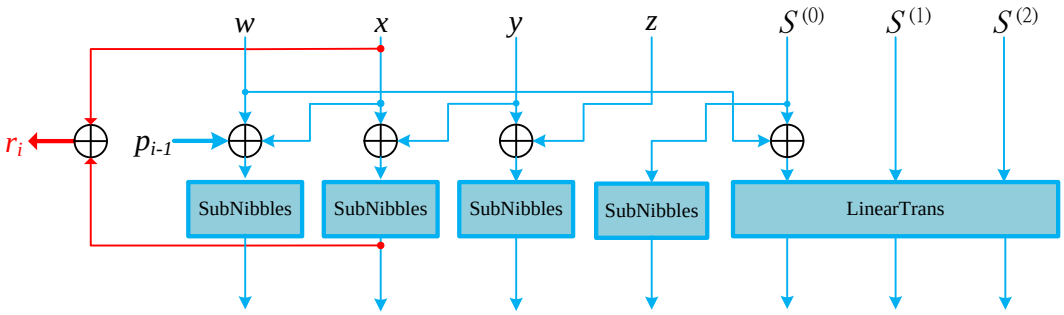


Figure 2.3: Processing plaintext in PANDA-s.

where $L(a,b) = (b, a \oplus b)$, a and b are 64-bit strings. $C_0 = 32||43||f6||a8||88||5a||30||8d$, $C_1 = 31||31||98||a2||e0||37||07||34$ and $C_2 = 4a||40||93||82||22||99||f3||1d$, all in hexadecimal.

2) Processing associated data: After padding and partition, the associated data A becomes $a_0 a_1 \cdots a_{s-1}$, then use the block a_i to update $state$ with RoundFunc. More specifically,

$\texttt{for } i = 0 \texttt{ to } s-1$
  $state \leftarrow \texttt{RoundFunc}_s(state, a_i)$

3) Processing plaintext: After padding and partition, the plaintext P becomes $p_0 p_1 \cdots p_{m-1}$. Use the block p_i to update $state$ with RoundFunc and use the key stream to encrypt the plaintext. More specifically,

$(state, r) \leftarrow \texttt{RoundFunc}(state, 0)$
$\texttt{for } i = 0 \texttt{ to } m-1$
  $c_i \leftarrow p_i \oplus r$
  $(state, r) \leftarrow \texttt{RoundFunc}(state, p_i)$

We illustrate this step in Fig. 2.3.

4) Finalization: Use $tempt_i$ to update $state$ with RoundFunc 14 times, and then output the XOR of some of state bits as the authentication tag T, where $tempt_i = adlen$ when i is even, $tempt_i = mslen$ when i is odd i, $adlen$ and $mslen$ is the bit-length of the associated data and the plaintext respectively. More specifically,

$\texttt{for } i = 0 \texttt{ to } 13$
  $state \leftarrow \texttt{RoundFuncs}(state, tempt_i)$
$T \leftarrow (w \oplus y, x \oplus z)$

2.2.2 The decryption: PANDA-s.Dec$_K(N, A, C, T)$

The initialization and processing associated data are the same as in the encryption of PANDA-s. After padding and partition, the ciphertext becomes $c_0 c_1 \cdots c_{m-1}$, and the decryption as following:

$(state, r) \leftarrow \texttt{RoundFunc}(state, 0)$
$\texttt{for } i = 0 \texttt{ to } m-1$
  $p_i \leftarrow c_i \oplus r$
  $(state, r) \leftarrow \texttt{RoundFunc}(state, p_i)$

The generation of authentication tag in the finalization of the decryption is the same as that in the encryption. If the generated tag is not equal to T then return $\perp$, else delete the padded bits and return P.

2.3 Specification of PANDA-b

The core of PANDA-b is permutation $\mathbf{R} = \text{RoundPerm}$. $\mathbf{R}^{14}$ represents the 14 times compositions of $\mathbf{R}$.

All the operations in PANDA-b are defined in terms of 7-block big-block. For the variable-length data such as A, P in the above notations, we pad and partition them into big-blocks before the operations. The padding method append a bit 1 and minimal bits of 0 to make the length a multiple of big-block. If the original data is a byte string, we use little-endian encoding to make them into a big-block string and vice versa.

2.3.1 The encryption: PANDA-b.Enc$_K(N, A, P)$

The encryption is divided into the 4 steps.

1) Initialization: It is the same is in PANDA-s.

2) Processing associated data:

pad and partition A into big-blocks as $A_0A_1 \cdots A_{s-1}$
$state \leftarrow \mathbf{R}^{14}(state \oplus A_i) \oplus state$

3) Processing plaintext:

pad and partition P into big-blocks as $P_0P_1 \cdots P_{m-1}$
for $i = 0$ to $m-1$
    $U \leftarrow \mathbf{R}^{14}(S_i \oplus P_i)$
    $C_i \leftarrow \mathbf{R}^{14}(U) \oplus S_i$
    $S_{i+1} \leftarrow U \oplus S_i$

where S_0 is the state after the previous step.

4) Finalization:

$(w, x, y, z, a, b, c) \leftarrow \mathbf{R}^{14}(C_{m-1} \oplus S_{m-1}) \oplus S_m$
$T \leftarrow (w \oplus y, x \oplus z)$

2.3.2 The decryption: PANDA-b.Dec$_K(N, A, C, T)$

The decryption is also divided into the following 4 steps. The steps of initialization and processing associated data are the same as those in the encryption of PANDA-b. The other two steps are as follows:

3) Processing ciphertext:

partition C into big-blocks as $C_0C_1 \cdots C_{m-1}$
for $i = 0$ to $m-1$
    $U \leftarrow \mathbf{R}^{-14}(C_i \oplus S_i)$
    $P_i \leftarrow \mathbf{R}^{-14}(U) \oplus S_i$
    $S_{i+1} \leftarrow S_i \oplus U$
where S_0 is the state after the previous step.

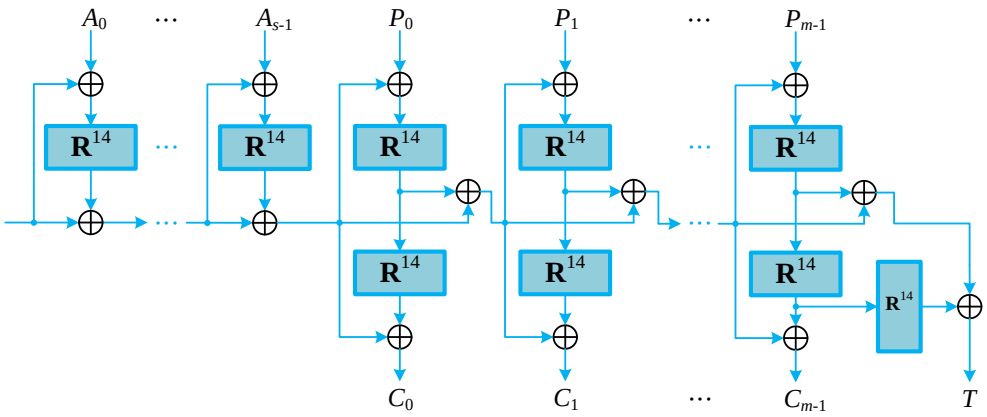


Figure 2.4: Processing associated data and plaintext in PANDA-b.

4) Finalization:

$(w, x, y, z, a, b, c) \leftarrow \mathbf{R}^{14}(C_{m-1} \oplus S_{m-1}) \oplus S_m$
$T' \leftarrow (w \oplus y, x \oplus z)$
`if` $T' = T$
    `return` $P_0 P_1 \cdots P_{m-1}$
`else`
    `return` $\perp$

Chapter 3

Security goals

The security goals of PANDA are listed in Table 3.1.

Table 3.1: Bits of security goals in two models

Goal	Nonce-respecting Model	Nonce-repeating Model	
	PANDA-s/b	PANDA-s	PANDA-b
confidentiality for the plaintext	128	/	128
integrity for the plaintext	128	128	128
integrity for the associated data	128	128	128
integrity for the public message number	128	128	128

Assumptions. We assume that the key of PANDA is generated uniformly at random.

Attacking Models. We consider the following attack models according to whether or not the nonce can be repeated.

1) Nonce-respecting model. In chosen plaintext attacks (CPA), the adversary can not repeat the nonce. In chosen ciphertext attacks (CCA), the adversary can repeat the nonce when query the decryption algorithm, but when the verification of PANDA fails, only a special symbol $\perp$ is revealed.
2) Nonce-repeating model. In CPAs or CCAs, the adversary can repeat the nonce. In CCAs, the adversary can get the corresponding plaintext even when the verification of PANDA fails.

Attacking Goals. The goal of adversary may be one of the following:

1) Recovering the key or the state of PANDA;
2) Recovering the plaintext;
3) Distinguishing the key stream from a random stream;
4) Forgery attack, i.e., generating a valid pair of ciphertext and its tag that never appears before.

Security Claims. We denote the nonce-respecting and nonce-repeating models as Model 1 and Model 2 respectively. We make the following security claims:

1) Resistance against key recovery: PANDA-s/b in Model 1 and 2.

2) Resistance against state recovery: PANDA-s/b in Model 1 and 2.
3) Resistance against plaintext recovery: PANDA-s in Model 1, PANDA-s/b in 1 and 2.
4) Resistance against linear distinguishing attack: PANDA-s/b in Model 1.
5) Resistance against forgery attack: PANDA-s/b in Model 1 and 2.
6) PANDA-b is a secure online cipher.
7) All the above resistances have 128 bits of security.

Details of security analysis are given in the next chapter.

Chapter 4

Security analysis

4.1 (Non-) Linear transformation in PANDA

The S-box in PANDA is the inverse function over $\mathbb{F}_{2^4}$, i.e., S-box$(x) = x^{-1}$. The maximum differential probability of the S-box is 2^{-2}.

The linear transformation in PANDA is defined by the matrix $\mathbb{A}$ over $\mathbb{F}_{2^{64}}$ in section 2.1. $\alpha \in \mathbb{F}_{2^{64}}$ is carefully chosen so that $f(x) = x^3 + (\alpha+1)x^2 + \alpha x + 1$ is an irreducible polynomial over $\mathbb{F}_{2^{64}}$. If we define $\mathbb{F}_{2^{64\times 3}} = \mathbb{F}_{2^{64}}(\beta)$ where β is a root of $f(x)$ and let $S = (a, b, c) \in \mathbb{F}_{2^{64}}^3$ corresponds to $a + b\beta + c\beta^2 \in \mathbb{F}_{2^{64\times 3}}$, then the matrix $\mathbb{A}$ corresponds to β^7. We will not distinguish the notations between matrix and finite field.

In the following analysis, the differences (or the linear masks) D_i in differential (or linear) analysis satisfy the equation

$$\sum_{i=0}^{s} D_i \mathbb{A}^i = 0.$$

This is a crucial point in our analysis.

If the minimal polynomial of $\mathbb{A}$ over $\mathbb{F}_{2^{64}}$ is $g(x) = 1 + \alpha_1 x + \alpha_2 x^2 + x^3$, then $\sum D_i x^i$ is a multiple of $g(x)$. Therefore, if $s = 3$, then $(D_0, D_1, D_2, D_3) = c(1, \alpha_1, \alpha_2, 1)$ for some $c \in \mathbb{F}_{2^{64}}$.

Definition 1 (Weight) *For a 64-bit block $D = d_0 d_1 \cdots d_{63}$, we define the weight of D as the number of non-zero inputs to SubNibbles, and denote it as $wt(D)$. More specifically,*

$$wt(D) = \#\{i | D_{(i)} \neq 0, 0 \leq i \leq 15\},$$

where $D_{(i)} = d_i d_{i+16} d_{i+32} d_{i+48}$. If a string consists of several blocks, we define its weight by the sum of all the block weights.

The linear transformation in our ciphers has the following property:

Property. The minimal polynomial of $\mathbb{A}$ over $\mathbb{F}_{2^{64}}$ has coefficients not all vanishing at every level. In other words, if the minimal polynomial is $g(x) = 1 + \alpha_1 x + \alpha_2 x^2 + x^3$, then for any $0 \leq i \leq 15$, $1_{(i)}$, $\alpha_{1(i)}$, $\alpha_{2(i)}$ are not all zero.

4.2 Analysis of PANDA-s

4.2.1 Linear analysis

A linear attack [6] exploits the fact some linear combinations of the input and output of the non-linear component are biased. We also call these linear combinations *linear approximations*. The only non-linear components in PANDA-s is SubNibbles consisting of S-boxes. The linear combination is usually defined by the operation of inner product. For two m-bit strings $\alpha = a_1a_2\cdots a_m$ and $B = b_1b_2\cdots b_m$, the inner product of α and B is defined as $\alpha \cdot B = a_1b_1 \oplus a_2b_2 \oplus \cdots \oplus a_mb_m$. When B is the input or the output of a function, α is called a *linear mask*.

The aim of the linear attack is to find a biased linear combination of the key stream generated by the stream cipher, i.e., $\sum_{i=1}^{s} \tau_i \cdot R_i$, where R_i are key stream blocks and τ_i are linear masks. Suppose that the linear masks before and after the four SubNibbles are (A_i, A'_i), (B_i, B'_i),(C_i, C'_i),(D_i, D'_i), and the corresponding state is $(w_i, x_i, y_i, z_i, S_i^{(0)}, S_i^{(1)}, S_i^{(2)})$.

In order to get the linear masks τ_i, we have

$$\sum_{i=0}^{n}[A_i \cdot (w_{i-1} + x_{i-1}) + A'_i \cdot w_i + B_i \cdot (x_{i-1} + y_{i-1}) + B'_i \cdot x_i + C_i \cdot (y_{i-1} + z_{i-1}) + C'_i \cdot y_i + D_i \cdot S_{i-1}^{(0)} + D'_i \cdot z_i + \tau_i \cdot (x_{i-1} + x_i)] = 0,$$

for all w_i, x_i, y_i, z_i and S_i. Then we have the following restricted functions

$$\begin{aligned}
w_{-1} &: A_0 + \sum_{j=0}^{n-1} D_{j+1} \cdot (A^{j+1})_{0,0} = 0;\\
x_{-1} &: A_0 + B_0 + \tau_0 = 0;\\
y_{-1} &: B_0 + C_0 = 0;\\
z_{-1} &: C_0 = 0;\\
S_{-1}^{(0)} &: \sum_{i=0}^{n} D_i \cdot A^i = 0;\\
w_i &: A_{i+1} + A'_i + \sum_{j=i+1}^{n-1} D_{j+1} \cdot (A^{j-i})_{0,0} = 0, \text{ for } 0 \le i \le n-2;\\
x_i &: A_{i+1} + B_{i+1} + B'_i + \tau_{i+1} + \tau_i = 0 \text{ for } 0 \le i \le n-1;\\
y_i &: B_{i+1} + C_{i+1} + C'_i = 0 \text{ for } 0 \le i \le n-1;\\
z_i &: C_{i+1} + D'_i = 0 \text{ for } 0 \le i \le n-1;\\
w_{n-1} &: A_n + A'_{n-1} = 0;\\
w_n &: A'_n = 0;\\
x_n &: B'_n + \tau_n = 0;\\
y_n &: C'_n = 0;\\
z_n &: D'_n = 0,
\end{aligned} \tag{4.1}$$

where the entry in the i-th row and j-th column of a matrix A is denoted by $A_{i,j}$.

Since the S-box used in this stream cipher is a map from $\mathbb{F}_2^4$ to $\mathbb{F}_2^4$, the mask code B_i can be regarded as a 16-dimension vector over $\mathbb{F}_2^4$, i.e.,

$$B_i = (B_{i(0)}, B_{i(1)}, \cdots, B_{i(15)}),$$

where $B_i \in \mathbb{F}_2^{64}$ and $B_{i(j)} \in \mathbb{F}_2^4$ for $0 \le j \le 15$. The weight of B_i, denoted by $\mathrm{wt}(B_i)$, is defined as $\mathrm{wt}(B_i) = |\{j \,|\, B_{i(j)} \neq 0,\ 0 \le j \le 15\}|$. Note that the input and output mask codes $(B_{i(j)}, B_{i'(j)})$ satisfy that $B_{i(j)} = 0$ if and only if $B_{i'(j)} = 0$. Similarly as other mask codes B_i', C_i, C_i', D_i and D_i'. Our strategy is to enumerate the number of active S-boxes by equalities (4.1) when $n = 5$. When $n = 5$, we have the following equalities:

$$\begin{aligned}
&B_0 = C_0 = A_2 = A_3 = A_4 = A_5 = C_5 = D_4 = D_5 = 0,\\
&C_{i+1} = D_i', \quad 0 \le i \le 3,\\
&B_1 = C_1,\\
&B_{i+1} = C_{i+1} + C_i', \quad 1 \le i \le 3,\\
&B_5 = C_4',\\
&A_0 = \tau_0 = D_0,\\
&A_1 + A_0' = D_2 \cdot A_{0,0} + D_3 \cdot (A^2)_{0,0},\\
&A_1' = D_3 A_{0,0},\\
&D_0 + D_1 \cdot A + D_2 \cdot A^2 + D_3 \cdot A^3 = 0.
\end{aligned} \tag{4.2}$$

Since the characteristic polynomial $f_{\mathbb{A}}(\lambda)$ of the linear transformation $\mathbb{A}$

$$f_{\mathbb{A}}(x) = 1 + \alpha_1 x + \alpha_2 x^2 + x^3$$

over $\mathbb{F}_{2^{64}}$ is irreducible, $(D_0, D_1, D_2, D_3) = c(1, \alpha_1, \alpha_2, 1)$ for some $c \in \mathbb{F}_{2^{64}}$, where $a_1, a_2 \in \mathbb{F}_2^{64}$. For convenience, denote $\mathcal{N}_{D(j)} = |\{i \,|\, D_{i(j)} \neq 0, 0 \le i \le 3\}|$, $0 \le j \le 15$. Similarly, denote $\mathcal{N}_{B(j)} = |\{i \,|\, B_{i(j)} \neq 0, 1 \le i \le 5\}|$, $\mathcal{N}_{C(j)} = |\{i \,|\, C_{i(j)} \neq 0, 1 \le i \le 4\}|$ and $\mathcal{N}_{(j)} = \mathcal{N}_{B(j)} + \mathcal{N}_{C(j)} + \mathcal{N}_{D(j)}$ for $0 \le j \le 15$. Next we will estimate the value of $\mathcal{N}_{(j)}$ for all values of $(D_{0(j)}, D_{1(j)}, D_{2(j)}, D_{3(j)})$ by (4.2). In other words, we will estimate the number of active S-boxes.

Lemma 1 *If $D_{0(j)} = D_{3(j)} \neq 0$, we have $\mathcal{N}_{(j)} \ge 8$.*

Proof: It is easy to see that $\mathcal{N}_{C(j)} = \mathcal{N}_{D(j)}$ by (4.2). To prove this lemma, it is divided into four cases as $(D_{1(j)}, D_{2(j)}) = (0,0)$, $(*,0)$, $(0,*)$ and $(*,*)$, where the notation "$*$" denotes nonzero value.

Case I, $(D_{1(j)}, D_{2(j)}) = (0,0)$. Then we have $\mathcal{N}_{D(j)} = 2$. By (4.2), we have $C_{1(j)} = D'_{0(j)} \neq 0$, $C_{2(j)} = D'_{1(j)} = 0$, $C_{3(j)} = D'_{2(j)} = 0$ and $C_{4(j)} = D'_{3(j)} \neq 0$. Therefore, $\mathcal{N}_{C(j)} = 2$. Similarly, by (4.2), $B_{1(j)} = C_{1(j)} + C'_{0(j)} \neq 0$, $B_{2(j)} = C_{2(j)} + C'_{1(j)} \neq 0$, $B_{3(j)} = C_{3(j)} + C'_{2(j)} = 0$, $B_{4(j)} = C_{4(j)} + C'_{3(j)} \neq 0$ and $B_{5(j)} = C'_{4(j)} \neq 0$, i.e., $\mathcal{N}_{B(j)} = 4$. Therefore, $\mathcal{N}_{(j)} = \mathcal{N}_{B(j)} + \mathcal{N}_{C(j)} + \mathcal{N}_{D(j)} = 8$.

Case II, $(D_{1(j)}, D_{2(j)}) = (*,0)$. By (4.2), we have $C_{1(j)} = D'_{0(j)} \neq 0$, $C_{2(j)} = D'_{1(j)} \neq 0$, $C_{3(j)} = D'_{2(j)} = 0$ and $C_{4(j)} = D'_{3(j)} \neq 0$, i.e., $\mathcal{N}_{D(j)} = \mathcal{N}_{C(j)} = 3$. Similarly, by (4.2), $B_{1(j)} = C_{1(j)} + C'_{0(j)} \neq 0$, $B_{3(j)} = C_{3(j)} + C'_{2(j)} \neq 0$, $B_{4(j)} = C_{4(j)} + C'_{3(j)} \neq 0$ and $B_{5(j)} = C'_{4(j)} \neq 0$. We can not determine whether $B_{2(j)} = C_{2(j)} + C'_{1(j)} = 0$ or not. Therefore, $\mathcal{N}_{B(j)} \ge 4$ and $\mathcal{N}_{(j)} \ge 10$.

Case III, $(D_{1(j)}, D_{2(j)}) = (0, *)$. Similarly as the proof of Case II, we have $\mathcal{N}_{(j)} \geq 9$.

Case IV, $(D_{1(j)}, D_{2(j)}) = (*, *)$. It is easy to see that $\mathcal{N}_{(j)} \geq 10$ since $\mathcal{N}_{C(j)} = \mathcal{N}_{D(j)} = 4$, $B_{1(j)} = C_{1(j)} \neq 0$ and $B_{5(j)} = C'_{4(j)} \neq 0$.

Combining with Cases I, II, III and IV, the proof of this lemma is completed.

Lemma 2 *If $D_{0(j)} = D_{3(j)} = 0$ and $(D_{1(j)}, D_{2(j)}) \neq 0$, we have $\mathcal{N}_{(j)} \geq 4$.*

Proof: The proof of this lemma is similar to that of Lemma 1.

For safety considerations, the number $\mathcal{N}$ of active S-boxes is preferably more than 80. By Lemma 1, we directly find that the number of active S-boxes decreases as the weight $\mathrm{wt}(D_i)$ decreases. Since the number of all possible values of (D_0, D_1, D_2, D_3) is $2^{64} - 1$, it can not be checked by exhaustive search. We only focus on the cases that the value of $\mathcal{N}$ is possible less than 80. To this end, we verify three cases that (I) $N \geq 5$, (II) $\mathrm{wt}(D_0) \leq 3$ and (III) $\mathrm{wt}(D_0) \geq 4$ and $N \leq 4$, where $N = |\{j \,|\, (D_{0(j)}, D_{1(j)}, D_{2(j)}, D_{3(j)}) = (0,0,0,0),\ 0 \leq j \leq 15\}|$.

For Case I, assume that $(D_{0(j_t)}, D_{1(j_t)}, D_{2(j_t)}, D_{3(j_t)}) = (0,0,0,0)$, $1 \leq t \leq 5$ and $0 \leq j_t \leq 15$. Let $D_0 = c = (c_0, c_1, \cdots, c_{63})$ which is regarded as $D_0 = \sum_{i=0}^{63} c_i \theta^i$ in $\mathbb{F}_{2^{64}}$. Then we have

$$c_{j_t} = c_{j_t+16} = c_{j_t+32} = c_{j_t+48} = 0, \quad 1 \leq t \leq 5.$$

Since $(D_0, D_1, D_2, D_3) = c(1, a_1, a_2, 1)$, then $D_1 = ca_1 = \sum_{i=0}^{63} (\theta^i a_1) c_i$. Denote $\theta^i a_1 = (d_{i,0}, d_{i,1}, \cdots, d_{i,63})$, $0 \leq i \leq 63$, and then we also have 20 linear equations

$$\sum_{i=0}^{63} d_{i,j_t+16k} c_i = 0, \quad 1 \leq t \leq 5 \text{ and } 0 \leq k \leq 3.$$

Similarly, we have other 20 linear equations

$$\sum_{i=0}^{63} e_{i,j_t+16k} c_i = 0, \quad 1 \leq t \leq 5 \text{ and } 0 \leq k \leq 3,$$

where $\theta^i a_2 = (e_{i,0}, e_{i,1}, \cdots, e_{i,63})$. Solving the 60 linear equations, the dimension of this linear equations is about 4, and the number of possible cases is about $\binom{16}{5} \times 2^4$. By the help of a computer, the minimal number of active S-boxes is 111.

For Case II, the mask code D_0 can have less than $\binom{16}{3} \times (2^{12} - 1)$ values. For these values, we can check it one by one with the help of a computer. As a result, the minimal number of active S-boxes is 102.

For Case III, if $N = 0$, we have $\mathcal{N} = \mathcal{N}_A + \mathcal{N}_B + \mathcal{N}_C + \mathcal{N}_D \geq 4 + 4 \times 8 + 4 \times 12 = 84$ by Lemma 1 and 2. Similarly, if $N = 1$, 2, 3 and 4, we have $\mathcal{N} \geq 80$, 76, 72 and 68, respectively. Since we only focus on the cases that the value of $\mathcal{N}$ is possible less than 80, the three sub-cases as (i) $\mathrm{wt}(D_0) = 4$ and $N = 2$, (ii) $\mathrm{wt}(D_0) = 5$ and $N = 3$, and (iii) $\mathrm{wt}(D_0) = 6$ and $N = 4$ should be considered. For sub-case (i), there are 64 equations with 64 unknowns. By the help of a computer, the minimal number of active S-boxes is 103. For sub-cases (ii) and (iii), there is no solution by the help of a computer.

Note that Cases I, II and III have contained all the possible cases that the value of $\mathcal{N}$ is possible less than 80. For example, if $N = 2$ and $\mathrm{wt}(D_0) \geq 5$, then $\mathcal{N} \geq 4 + 5 \times 8 + 9 \times 4 = 80$.

4.2.2 Differential analysis

The probability that two different messages show the same authentication tag is associated with the number of active S-boxes which will be analyzed in the following. Without loss of generality, the first message has non-zero difference, and all the state differences after initialization are zero. It is easy to know that state differences before round 5 can not be all zero with the first non-zero message. Here we just count the number of active S-boxes that all state differences of round 5 are zero, because there usually have more active S-boxes if the zero state differences appear after more rounds. Suppose that the differences before and after the four SubNibbles are (A_i, A_i'), (B_i, B_i'),(C_i, C_i'),(D_i, D_i'), the difference of message m_i is Δm_i. Combining with theory and experience, the lower bound of the number is 84 in the situation that the active S-boxes in A_1, A_2, D_3 have not be contained.

The following relations between the differences can be easily obtained:

$$\begin{cases} A_i = A_{i-1}^{'} + B_{i-1}^{'} + \Delta m_i \\ B_i = B_{i-1}^{'} + C_{i-1}^{'} \\ C_i = C_{i-1}^{'} + D_{i-1}^{'} \\ D_i = \Delta S_{i-1}^{(0)} = (\sum_{j=0}^{i-2} A^{i-j-1} A_j^{'})^{(0)} \end{cases} \quad i \geq 2 \tag{4.3}$$

Obviously, the non-zero message difference can not propagate to all states. Some state differences are zero at first:

$$B_0 = C_0 = D_0 = B_1 = C_1 = D_1 = B_2 = C_2 = B_3 = 0 \tag{4.4}$$

In our consideration, all state differences of round 5 will be zero. i.e.,

$$\begin{cases} A_5 = B_5 = C_5 = D_5 = 0 \\ \Delta S_4 + A_4^{'} = 0 \end{cases} \tag{4.5}$$

.

Combing Eq. (4.3), (4.4) and (4.5), some relations between the differences can be expressed as follows:

$$\begin{cases} C_3 = D_2^{'} \\ B_4 = C_3^{'} \\ C_4^{'} = B_4^{'} \\ D_4^{'} = C_4^{'} \end{cases} \tag{4.6}$$

$$\begin{cases} D_5 = (\mathbb{A}^4 A_0^{'} + \mathbb{A}^3 A_1^{'} + \mathbb{A}^2 A_2^{'} + \mathbb{A} A_3^{'})^{(0)} = 0 \\ \Delta S_4 + A_4^{'} = \mathbb{A}^4 A_0^{'} + \mathbb{A}^3 A_1^{'} + \mathbb{A}^2 A_2^{'} + \mathbb{A} A_3^{'} + A_4^{'} = 0 \end{cases} \tag{4.7}$$

Since the minimal polynomial of $\mathbb{A}$ is

$$x^3 + (1 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^7)x^2 + (1 + \alpha^2 + \alpha^5 + \alpha^6 + \alpha^7)x + 1,$$

the following consequences can be obtained from Eq.(4.7):

$$\begin{pmatrix} A_0^{'} \\ A_1^{'} \\ A_2^{'} \\ A_3^{'} \end{pmatrix} = \tau \begin{pmatrix} 1 \\ 1+\alpha^2+\alpha^3+\alpha^4+\alpha^7 \\ 1+\alpha^2+\alpha^5+\alpha^6+\alpha^7 \\ 1 \end{pmatrix} \tag{4.8}$$

$$A_4^{'} = 0 \tag{4.9}$$

where τ belongs to $\mathbb{F}_{2^{64}}$.

Another two relations can also be obtained from Eq.(4.3), (4.8) and (4.9):

$$\begin{cases} A_3^{'} = D_4 \\ A_0^{'} = A_3^{'} \end{cases} \tag{4.10}$$

The Eq.(4.6) and Eq.(4.10) show the fact that the position of active (non-active) S-boxes in $D_2, C_3, B_4, C_4, D_4, A_0, A_3$ are all the same. Next, the number of active S-boxes are counted as follows:

1. If 16 S-boxes are all active in D_2, $112 = 16 * 7$ S-boxes in $D_2, C_3, B_4, C_4, D_4, A_0, A_3$ are active.
2. If only 1 sbox is non-active in D_2, $105 = 15 * 7$ S-boxes in $D_2, C_3, B_4, C_4, D_4, A_0, A_3$ are active.
3. If only 2 S-boxes are non-active in the D_2, $98 = 14 * 7$ S-boxes in $D_2, C_3, B_4, C_4, D_4, A_0, A_3$ are active.
4. If only 3 S-boxes are non-active in the D_2, $91 = 13 * 7$ S-boxes in $D_2, C_3, B_4, C_4, D_4, A_0, A_3$ are active.
5. If only 4 S-boxes are non-active in the D_2, $84 = 12 * 7$ S-boxes in $D_2, C_3, B_4, C_4, D_4, A_0, A_3$ are active.
6. If at least 5 S-boxes are non-active in D_2, we will count the active S-boxes by experience which shows that at least 108 S-boxes are active totally. The process of the experience is stated as follows:
   First, if the position in D_2 is non-active, the corresponding position in C_3, C_4 and D_4 are non-active from Eq.(4.6). Besides, $C_4 = C_3^{'} + D_3^{'}$ gives that the position in D_3 is also non-active. So the non-active position in D_2 which is the same as the non-active in D_4 is contained by the non-active position in D_3. After letting any five nibble in D_2, D_3 and D_4 be zero, we obtain the equations for τ from the next three equations

$$\begin{cases} D_2 = (\mathbb{A}A_0^{'})^0 \\ D_3 = (\mathbb{A}^2A_0^{'} + \mathbb{A}A_1^{'})^0 \\ D_4 = (\mathbb{A}^3A_0^{'} + \mathbb{A}^2A_1^{'} + \mathbb{A}A_0^{'})^0 \end{cases} \tag{4.11}$$

   where

$$\begin{pmatrix} A_0^{'} \\ A_1^{'} \\ A_2^{'} \\ A_3^{'} \end{pmatrix} = \tau \begin{pmatrix} 1 \\ 1+\alpha^2+\alpha^3+\alpha^4+\alpha^7 \\ 1+\alpha^2+\alpha^5+\alpha^6+\alpha^7 \\ 1 \end{pmatrix} \tag{4.12}$$

   For each τ, we get the values of $A_1^{'}, A_2^{'}, A_3^{'}, A_4^{'}, D_2, D_3, D_4$. If the active positions in D_2, D_4 are the same and the active positions in D_3 are contained by the active positions in D_2, count the number of the active S-boxes in $A_0^{'}, A_1^{'}, A_2^{'}, A_3^{'}, D_2, D_3$. The positions of active (non-active) S-boxes in D_2, C_3, B_4, C_4, D_4 are all the same, so the number of active positions in $A_0^{'}, A_1^{'}, A_2^{'}, A_3^{'}, D_2, D_3, D_4, C_3, B_4, C_4$ have been obtained. The lower bound of the number is 108 by experience.

To sum up, the lower bound of the number of the active S-boxes is 84 in the situation that we did not consider the active S-boxes in A_1, A_2, D_3.

4.3 Analysis of PANDA-b

4.3.1 Differential analysis of iterations of R

For any trail, the number of active S-boxes is greater than

$$\sum_i (wt(A_i) + wt(D_i)).$$

The lower bound can be estimated as $(2n - (n+3)) \times 15 = 15(n-3)$, if we take (A_i, S_0, D_i) as a random linear code of dimension $64(n+3)$.

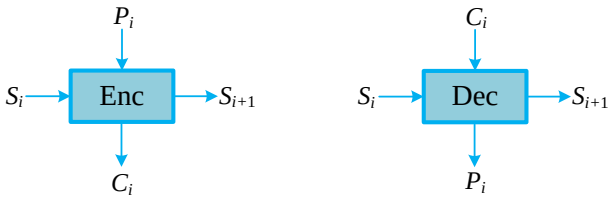
This estimation makes sense only if we exclude some obvious subspaces with vanishing A_i, S_i. For example, we can have 4 consecutive rounds with vanishing A_i, S_i for all trails ≥ 4 rounds, so we would decree the estimation as $15 \times (n-7)$, and we hope this is valid for $n \leq 14$.

In fact, we can find very thin trails up to 7 rounds. For example: we begin with a state with only 1 active S-box at z, rewinding 2 rounds would get only 5 S-box, and forwarding 5 rounds would get 15 S-boxes. So we have a class of 7-round trails with only 20 S-boxes. The lowest trails of less than 10 rounds seem to lie in extends of these trails. Extending these 7-round trails with 3 more rounds will result in at least 60 more active S-boxes. We hope these are the lowest trails of 10 rounds, and it is unlikely that there exist exploitable differential characteristics for $\mathbf{R}^{14}$.

4.3.2 Online cipher and PANDA-b

Definition 2 (Ideal online cipher) *It is an oracle accepting queries of the two types: $q_1 q_2 \cdots q_k$ for encryption or for decryption, where q_i is the blocks of the plaintext or ciphertext. The oracle $\mathcal{I}$ maintains a list recording the previous queries and answers. It treats a fresh query as follows: find maximal match (may be empty) for prefix, say $q_1 q_2 \cdots q_i$, give the answer given by the list. For the rest of blocks choose uniform random blocks as the answer. Append $q_1 \cdots q_k$ and its answer to the list.*

A real online cipher in the following



also defines an oracle R_K, where K is the key. It answers queries by running the encryption and decryption algorithms. We call an online encryption scheme *secure online cipher* if any efficient adversary A cannot distinguish $\mathcal{I}$ and R_K, or in other words,

$$\Pr[A^{\mathcal{I}} = 1] - \Pr[K \leftarrow_R \{0,1\}^\lambda : A^{R_K} = 1]$$

is negligible.

If we model $\mathbf{R}^{14}$ in PANDA as a public random permutation oracle, then it is not hard to see that PANDA-b is a secure online cipher. The result is showed by the following lemmas.

Lemma 3 *For a block cipher E (its reverse is D), we define a oracle $\mathcal{I}[E]$. It begins with a random key k_0. It maintain a list recording all previous queries and associated key sequence. To treat a fresh query, it first check the list to find maximal match (may be empty) for prefix, say $(q_1 q_2 \cdots q_i, k_0 k_1 k_2 \cdot k_i)$, give the answer given by the list. It replies q_{i+1} with E_{k_i} for encryption or D_{k_i} for decryption, then select a uniformly random key as k_{i+1} and go on. It append $q_1 \cdots q_k$, its answer and the key sequence to the list when a query is completed.*

If E is secure block cipher, $\mathcal{I}[E]$ is a secure online cipher. In other words, any adversary can not distinguish $\mathcal{I}$ and $\mathcal{I}[E]$.

Lemma 4 *[10] Let T be public random permutation oracle. Then $Enc(K, M) := K \oplus P(M \oplus K)$ is a secure block cipher (or a pseudorandom permutation).*

Lemma 5 *If we model $\mathbf{R}^{14}$ as a public random permutation oracle, any efficient adversary can not distinguish $\mathcal{I}[E]$ and* PANDA-b.

For practical security of PANDA-b, we can not take $\mathbf{R}^{14}$ as an oracle; it has obvious weakness: it commutes with a simple function $\mathbf{R}$. But it seems that this weakness is not exploitable for a practical attack to invalidate Lemma 4 because of the large block size. For a practical attack to invalidate Lemma 5, it seems that the most effective approach is to exploit the differential characteristic of $\mathbf{R}^{14}$, which (we contend in section 4.3.1) is also infeasible.

Chapter 5

Features

Compared with block cipher modes of operation, such as AES-GCM, PANDA as a family of directly constructed ciphers does not have the performance restriction by the underlying block cipher and does not suffer from birthday attacks which is universal in models. Furthermore there are three features we want to emphasize.

Unique structure. The structure of the underlying round function in PANDA is different from the usual structure of SP-networks. Putting S-boxes and a linear transformation in parallel also makes the analysis different. We hope to bring some new ideas in the design of ciphers.

Security analysis. In order to guarantee the security of ciphers, it is important to provide some evidences, e.g., a reduction proof or analysis to the classic attacks such as differential or linear attacks. For those that have such evidences may suffer from the classic attacks later. PANDA-s is a mixture of a stream cipher and a MAC. We measure the security of the stream cipher by linear distinguishing attacks, and the security of the MAC by differential attacks. We consider the security of PNADA-b when the underlying permutation is a public random oracle.

Note that some of analyses are not delicate enough, we will improve them in the process of CAESAR.

Misuse resistance. The design of PANDA bears the robustness against misuse resistance in mind. We believe that even if the nonce is reused, the differential property of PANDA-s guarantees the security of integrity, and the online-cipher properties of PANDA-b make it strong enough to provide both confidentiality and integrity.

Chapter 6

Design rationale

6.1 Objective

Our goal is to design a stream-friendly AE of 128-bit security which is efficient on common software and hardware platforms. The security goal is mainly to stop nonce-respecting adversaries, while allowing the choice to stand for nonce-misusing via simple construction based on the round function of the design.

6.2 Choice of algorithm structure

To be stream-friendly, the outlook of the algorithm should look like:

$$c_i = m_i \oplus f(m_{i-1}, S_{i-1})$$

$$S_i = g(m_{i-1}, S_{i-1})$$

where m_i, c_i, S_i are message, ciphertext, state respectively. g is the state-transfer function and f is the output function. That is, our AE is a mixture of a stream cipher and a MAC. We assume that the AE is secure if both the stream cipher and the MAC are secure. We measure the security of the stream cipher by linear distinguishing attack, and the security of the MAC requires that the adversary can not figure out two different message streams resulting the same state. These define the problem of our linear and differential analysis.

Since the output function is simply based on the state transfer, the main issue here is the choice of the state transfer function. One option is to use the round-function structure of a good block cipher or hash function [7, 3] . This would make the linear and differential analysis resulting large linear-spaces, which makes it hard to estimate the lower bound of active S-boxes; to get a verifiable bound, one has to choose very small message block size, which results in inefficient algorithm. Inspired by ideas of the stream cipher Snow [9], we use a separated linear transform for diffusion, but our algorithm is different from Snow in following respect:

1) We choose small S-boxes and large non-linear state, this makes our non-linear part can be efficiently implemented both in hardware and software.

2) We choose to output 1 bit for each S-box, and the size of linear part as in PANDA seems appropriate. Another choice would be to output 1 bit for every 2 S-boxes, and the linear part may only have 32 bits. Our analysis for this choice has not been completed.
3) We choose a complex linear transform but a smaller linear state; this makes our algorithm more compact.
4) We choose to let the non-linear part and linear part affect each other. This improves the results of linear and differential analysis.

6.3 Choice of linear transformation $\mathbb{A}$

We choose $\mathbb{A}$ according to the following considerations:

1) Efficient both in software and hardware. We decide $\mathbb{A}$ be the form $\mathbb{A} = \beta^n$, where β is simple.
2) $\mathbb{A}^{-1}$ has similar implementation with $\mathbb{A}$. This is because we wish our round function can be used in PANDA-b. So we choose $\beta \in \mathbb{F}_{2^{64\times 3}}$ and the minimal polynomial over $\mathbb{F}_{2^{64}}$ as: $x^3 + (\alpha+1)x^2 + \alpha x + 1$.
3) The minimal polynomial of $\mathbb{A}$ over $\mathbb{F}_{2^{64}}$ should have coefficients not all vanishing at every level (See section 4.1). This seems sufficient so that PANDA is secure.

These considerations lead to our choice of $\mathbb{A}$ in PANDA.

6.4 Assumptions

The main assumptions during our design and security analysis are all related to the problem of finding the minimal "weight" of non-zero code words in some linear code. The simplest form of the problem is as follows:

Let $f_i : \{0,1\}^r \to \{0,1\}^b, i = 1, \cdots, n$ be n given linear maps. For a code-word $x \in \{0,1\}^r$, define its weight $wt(x) = \#\{i : f_i(x) \neq 0\}$. Here $b = 4$ is the S-box size, n is the number of S-boxes involved and r is the dimension of a linear code.

If $\{f_i\}$ seems independent, we estimate the lower bound for the minimal weight as

$$(n - r/b)(1 - 2^{-b}).$$

If $n \gg r/b$, we assume this estimation is quite close to the real bound; If $n < 2r/b$, we lower the estimation by $log_2\binom{n}{r/b}/b$.

More general "weight" functions can be defined, in order to treat the case that there are S-boxes not included in the linear code.

In this case, we remove $r/b + n \cdot 2^{-b}$ S-boxes which are most significant in weight counting, and summing the remained weight as the lower bound of active S-boxes.

This kind of estimations makes us to believe that the more rounds involved in the linear (differential) analysis of PANDA, the larger lower bound of the number of active S-boxes. This indicates that the analysis given in sections 4.2 and 4.3 is enough for assuring security of PANDA against differential and linear cryptanalysis.

The designers have not hidden any weaknesses in PANDA.

Chapter 7

Intellectual property

We have not applied for any patents on PANDA. We explicitly release any intellectual property rights to PANDA into the public domain. If any of this information changes, the submitters will promptly (and within at most one month) announce these changes on the crypto-competitions mailing list.

Chapter 8

Consent

The submitters hereby consent to all decisions of the CAESAR selection committee regarding the selection or non selection of this submission as a second-round candidate, a third-round candidate, a finalist, a member of the final portfolio, or any other designation provided by the committee. The submitters understand that the committee will not comment on the algorithms, except that for each selected algorithm the committee will simply cite the previously published analyses that led to the selection of the algorithm. The submitters understand that the selection of some algorithms is not a negative comment regarding other algorithms, and that an excellent algorithm might fail to be selected simply because not enough analysis was available at the time of the committee decision. The submitters acknowledge that the committee decisions reflect the collective expert judgments of the committee members and are not subject to appeal. The submitter understands that if he disagrees with published analyses then he is expected to promptly and publicly respond to those analyses, not to wait for subsequent committee decisions. The submitters understand that this statement is required as a condition of consideration of this submission by the CAESAR selection committee.

Acknowledgment

We would like thank all that participate in our discussion. They are Jinyong Shan, Danping Shi, Zhelei Sun, Kaiyan Zheng, Kexin Qiao, Xiaoshuang Ma, Ling Song. All of them undertook some testing or writing work during the design of PANDA.

Bibliography

[1] Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: Authenticated permutation-based encryption for lightweight cryptography (2013), http://eprint.iacr.org/2013/791 1

[2] Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Duplexing the sponge: Single-pass authenticated encryption and other applications. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7118, pp. 320–337. Springer (2011) 1

[3] Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Keccak. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT. Lecture Notes in Computer Science, vol. 7881, pp. 313–314. Springer (2013) 1, 20

[4] Bilgin, B., Bogdanov, A., Knezevic, M., Mendel, F., Wang, Q.: FIDES: Lightweight authenticated cipher with side-channel resistance for constrained hardware. In: Bertoni, G., Coron, J.S. (eds.) CHES. Lecture Notes in Computer Science, vol. 8086, pp. 142–158. Springer (2013) 1

[5] Bogdanov, A., Mendel, F., Regazzoni, F., Tischhauser, E., Rijmen, V.: ALE: AES-based lightweight authenticated encryption. Lecture Notes in Computer Science (2013) 2

[6] Coppersmith, D., Halevi, S., Jutla, C.S.: Cryptanalysis of stream ciphers with linear masking. In: Yung, M. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 2442, pp. 515–532. Springer (2002) 12

[7] Daemen, J., Rijmen, V.: The design of Rijndael: AES-the advanced encryption standard. Springer (2002) 1, 20

[8] Daemen, J., Rijmen, V.: The Pelican MAC function. IACR Cryptology ePrint Archive 2005, 88 (2005) 2

[9] ETSI/SAGE: Specification of the 3GPP confidentiality and integrity algorithms UEA2& UIA2. document 2: SNOW 3G specifications. (2006) 20

[10] Even, S., Mansour, Y.: A construction of a cioher from a single pseudorandom permutation. In: Imai, H., Rivest, R.L., Matsumoto, T. (eds.) ASIACRYPT. Lecture Notes in Computer Science, vol. 739, pp. 210–224. Springer (1991) 18

[11] Ferguson, N., Whiting, D., Schneier, B., Kelsey, J., Lucks, S., Kohno, T.: Helix: Fast encryption and authentication in a single cryptographic primitive. In: Johansson, T. (ed.) Fast Software Encryption

2003, Lecture Notes in Computer Science, vol. 2887, pp. 330–346. Springer Berlin Heidelberg (2003) 1

[12] Jakimoski, G., Khajuria, S.: ASC-1: An authenticated encryption stream cipher. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7118, pp. 356–372. Springer (2011) 2

[13] McGrew, D.A., Viega, J.: The Galois/Counter mode of operation (GCM) (2004), http://csrc.nist.gov/groups/ST/toolkit/BCM/ 1

[14] Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: Reiter, M.K., Samarati, P. (eds.) ACM Conference on Computer and Communications Security. pp. 196–205. ACM (2001) 1

[15] Whiting, D., Schneier, B., Lucks, S., Muller, F.: Phelix: fast encryption and authentication in a single cryptographic primitive, estream. ecrypt stream cipher project, report 2005/020 (2005). www, ecrypt. en, org/stream (2005) 1

[16] Wu, H., Preneel, B.: AEGIS: A fast authenticated encryption algorithm (full paper). Cryptology ePrint Archive, Report 2013/695 (2013), http://eprint.iacr.org/2013/695 2