A forgery and state recovery attack on the authenticated cipher PANDA-s

Xiutao FENG, Fan ZHANG and Hui WANG

Key Laboratory of Mathematics Mechanization, Academy of Mathematics and Systems Science, CAS, China (e-mail: fengxt@amss.ac.cn)

Abstract. PANDA is a family of authenticated ciphers submitted to CARSAR, which consists of two ciphers: PANDA-s and PANDA-b. In this work we present a state recovery attack against PANDA-s with time complexity about $2^{64}$ under the known-plaintext-attack model, which needs about 132 pairs of known plaintext/ciphertext. Based on the above attack, we further deduce a forgery attack against PANDA-s. Our results show that PANDA-s is far from the goal of its security design (128-bit level).

Keywords: CAESAR, PANDA, state recovery attack, forgery attack.

1 Introduction

Authenticated cipher is a cipher combining encryption with authentication, which can provide confidentiality, integrity and authenticity assurances on the data simultaneously and has been widely used in many network session protocols such as SSL/TLS [1, 2], IPSec [3], etc. Currently a new competition is calling for submissions of authenticated ciphers, namely CAESAR [4]. This competition follows a long tradition of focused competitions in secret-key cryptography, and is expected to have a tremendous increase in confidence in the security of authentication ciphers.

PANDA is a family of authenticated ciphers designed by D. Ye et al and has been submitted to the CAESAR competition [5]. PANDA consists of two ciphers: PANDA-s and PANDA-b, and both are based on a simple round function. PANDA-s is similar to authenticated encryption (in short AE) with sponge structures [6] and is a mixture of a stream cipher and a MAC. PANDA-b is an online cipher like APE [7] with a permutation. In [8] Y. Sasaki et al present a forgery attack against PANDA-s under the condition of nonce reuse. It should be pointed that the nonce is usually a counter and is used once, thus it is easy to avoid launching Y. Sasaki et al’ attack in practice. As for PANDA-s, in this work we present a state recovery attack with time complexity about $2^{64}$ under the known-plaintext-attack model, which needs about 132 pairs of known plaintext/ciphertext. What is more, based on the above attack, we further deduce a

* This work was supported by the Natural Science Foundation of China (Grant No. 61121062, 11071285), the 973 Program (Grant No. 2011CB302401)
forgery attack against PANDA-s. Our results show that PANDA-s is far from the goal of its security design (128-bit level).

The rest of this paper is organized as follows: in section 2 we recall PANDA-s briefly, and in section 3 we provide a state recovery attack and a forgery attack against PANDA-s. Finally section 4 concludes the paper.

2 Description of PANDA-s

In this section we recall PANDA-s briefly. Since our attack does not involve in the initialization and the process of associated data of PANDA-s, thus here we omit them, and more details of PANDA-s can be found in [5].

PANDA-s takes in a 128-bit key \( K \), a 128-bit nonce \( N \), a 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. The main part of PANDA-s is a round function RoundFunc, which is a bijection from an eight 64-bit-block input to an eight 64-bit-block output. The state of PANDA-s is seven 64-bit blocks, which is a part of the input and output of RoundFunc. RoundFunc consists of four non-linear transformations SubNibbles and a linear transformation LinearTrans, as shown in Fig. 1.

![Fig. 1 The round function RoundFunc in PANDA-s](image)

Let \((w, x, y, z, S_0, S_1, S_2, m)\) and \((w', x', y', z', S'_0, S'_1, S'_2, r)\) be the input and the output of RoundFunc respectively. Then the specific process of RoundFunc is defined as follows:

**RoundFunc**

\[
\begin{align*}
&w' \leftarrow \text{SubNibbles}(w \oplus x \oplus m) \\
&x' \leftarrow \text{SubNibbles}(x \oplus y) \\
&y' \leftarrow \text{SubNibbles}(y \oplus z) \\
&z' \leftarrow \text{SubNibbles}(S_0) \\
&(S'_0, S'_1, S'_2) \leftarrow \text{LinearTrans}(S_0 \oplus w, S_1, S_2) \\
&r \leftarrow x \oplus x' \\
\text{return } (w', x', y', z', S'_0, S'_1, S'_2, r)
\end{align*}
\]

2.1 SubNibbles

SubNibbles is a nonlinear transformation from a 64-bit input to a 64-bit output, and is shown in Fig. 2. Let \( a_0 a_1 \cdots a_{63} \) and \( b_0 b_1 \cdots b_{63} \) be the input and the output of SubNibbles respectively. Then \( b_{i+16} b_{i+32} b_{i+48} = S(a_{i+16} a_{i+32} a_{i+48}) \), where \( S(\cdot) \) represents a \( 4 \times 4 \) S-box and is defined as in [5], \( i = 0, 1, \cdots, 15 \).
2.2 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 $\theta$ is a root of $p(x)$. The block $a_0a_1\cdots a_{63}$ corresponds to $a_0 + a_1\theta + \cdots + a_{62}\theta^{62} + a_{63}\theta^{63} \in \mathbb{F}_{2^{64}}$. The linear transformation $\text{LinearTrans}$ is defined as $\text{LinearTrans}(S_0, S_1, S_2) = (S_0, S_1, S_2)A$, where the matrix

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & \alpha & \alpha + 1 \end{pmatrix}$$

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

2.3 Encryption

Let $p_0p_1\cdots p_{m-1}$ be the plaintext and $\text{state}$ be the internal state of PANDA-s after initialization. Then the encryption is described as below:

$$(\text{state}, r) \leftarrow \text{RoundFunc}(\text{state}, 0)$$
for $t = 0$ to $m - 1$

$$c_t \leftarrow p_t \oplus r$$
$$(\text{state}, r) \leftarrow \text{RoundFunc}(\text{state}, p_t)$$

2.4 The tag $T$

Use $\text{tempt}_i$ to update $\text{state}$ with RoundFunc 14 times, and then output the XOR of some of state bits as the authentication tag $T$, where $\text{tempt}_i = \text{adlen}$ when $i$ is even, $\text{tempt}_i = \text{mslen}$ when $i$ is odd, $\text{adlen}$ and $\text{mslen}$ are the bit-length of the associated data and the plaintext respectively. More specifically,

for $i = 0$ to $13$

$$\text{state} \leftarrow \text{RoundFunc}(\text{state}, \text{tempt}_i)$$

$$T \leftarrow (w \oplus y, x \oplus z)$$

Fig. 2 SubNibbles acts on the individual columns of its input block
3 A forgery and state recovery attack on PANDA-s

In this section we always assume that an attacker has known the plaintext \(p_{t+i}\) got by a portion of the ciphertext \(c_{t+i}\) after time \(t \geq 0\), where \(i = 0, 1, \ldots, m-1\), and \(m\) is large enough for the attacker to launch his attack. Since \(r_{t+i} = p_{t+i} \oplus c_{t+i}\) for \(t \geq 0\), thus the attacker knows the key stream words \(\{r_{t+i}\}_{0 \leq i \leq m-1}\) as well. Our attack is shown as below.

3.1 Guess \(x_t\) and recover the sequence \(\{x_{t+i}\}_{i \geq 0}, \{y_{t+i}\}_{i \geq 0}, \{z_{t+i}\}_{i \geq 0}\) and \(\{S_{0,t+i}\}_{i \geq 0}\)

It is noticed that \(x_{t+i+1} = x_{t+i} \oplus r_{t+i}\) for any \(i \geq 0\), thus we get the whole sequence \(\{x_{t+i}\}_{i \geq 0}\) as soon as some \(x_{t+i}\) is known. Since

\[
y_{t+i} = \text{SubNibbles}^{-1}(x_{t+i+1}) \oplus x_{t+1},
\]

\[
z_{t+i} = \text{SubNibbles}^{-1}(y_{t+i+1}) \oplus y_{t+1},
\]

\[
S_{0,t+i} = \text{SubNibbles}^{-1}(z_{t+i+1}),
\]

thus we further recover \(\{y_{t+i}\}_{i \geq 0}, \{z_{t+i}\}_{i \geq 0}\) and \(\{S_{0,t+i}\}_{i \geq 0}\).

3.2 Eliminate the variables \(\{S_{1,t+i}\}_{i \geq 0}\) and \(\{S_{2,t+i}\}_{i \geq 0}\) and get equations on \(\{w_{t+i}\}_{i \geq 0}\)

When \(\{S_{0,t+i}\}_{i \geq 0}\) are known, we view \(S_{1,t}\) and \(S_{2,t}\) as initial variables and \(\{w_{t+i}\}_{i \geq 0}\) as parameters in the LinearTrans. We need only three equations got at three distinct times in order to eliminate the initial variables \(S_{1,t}\) and \(S_{2,t}\), and get one equation only on \(\{w_{t+i}\}_{i \geq 0}\). More precisely, the process is shown below:

First we get three equations at time \(t + 1, t + 2\) and \(t + 2\):

\[
S_{0,t+1} = (S_{0,t} \oplus w_t, S_{1,t}, S_{2,t})A e_1,
\]

\[
S_{0,t+2} = ((S_{0,t} \oplus w_t, S_{1,t}, S_{2,t})A^2 + (w_{t+1}, 0, 0)A) e_1,
\]

\[
S_{0,t+3} = ((S_{0,t} \oplus w_t, S_{1,t}, S_{2,t})A^3 + (w_{t+2}, 0, 0)A + (w_{t+1}, 0, 0)A^2) e_1,
\]

where \(e_1 = (1, 0, 0)'\) is a basic column vector.

Second, we eliminate the variables \(S_{1,t}\) and \(S_{2,t}\) from the above equations and get

\[
w_{t+2} \oplus C_5 w_{t+1} \oplus C_6 w_t = C_0,
\]

where \(C_0 = C_1 S_{0,t+3} \oplus C_2 S_{0,t+2} \oplus C_3 S_{0,t+1} \oplus C_4 S_{0,t},\) and \(C_1, C_2, \ldots, C_6\) are constants as defined in Appendix A.
3.3 Find a multiple of $x^2 \oplus C_5x \oplus C_6$ with coefficients 0 or 1

It is noticed that the computation of the S-boxes in the SubNibbles can be done in parallel, we need to find a nonzero multiple of $x^2 \oplus C_5x \oplus C_6$ with coefficients 0 or 1 in $F_{2^64}$ in order to solve equation (7) faster. Indeed we do it easily. One can check the following multiple $f(x)$

$$f(x) = \bigoplus_{i \in I} x^i$$

such that $x^2 \oplus C_5x \oplus C_6|f(x)$, where

$$I = \{0, 4, 6, 7, 8, 10, 11, 14, 15, 17, 18, 19, 21, 23, 26, 30, 31, 32, 33, 34, 35, 37, 39, 43, 45, 46, 47, 49, 50, 51, 52, 55, 59, 61, 63, 64, 67, 68, 70, 72, 73, 74, 77, 78, 79, 83, 85, 89, 91, 94, 96, 97, 99, 100, 101, 103, 105, 106, 107, 108, 109, 110, 112, 113, 115, 117, 118, 119, 122, 124, 125, 127\}.$$

So we have

$$\bigoplus_{i \in I} w_{t+i} = C,$$  \hspace{1cm} (8)

where $C$ is a constant.

3.4 Solve $w_t$ by looking up a table directly

Denote $x[j] = x_{j}x_{j+16}x_{j+32}x_{j+48}$ for a 64-bit word in $F_{2^64}$, where $0 \leq j \leq 15$. By the definition of the SubNibbles, we have

$$w_{t+i+1}[j] = S(w_{t+i}[j] \oplus x_{t+i}[j] \oplus p_{t+i}[j])$$  \hspace{1cm} (9)

for $i \geq 0$ and $0 \leq j \leq 15$.

On the other hand, by equation (8) we get

$$\bigoplus_{i \in I} w_{t+i}[j] = C[j], \hspace{1cm} 0 \leq j \leq 15.$$  \hspace{1cm} (10)

Combine equations (9) and (10), and we solve $w_t[j]$ in turn from $j = 0$ to 15. A faster method is to set up a table to look up $w_t[j]$ directly. More precisely, since the plaintext $\{p_{t+i}\}_{i \geq 0}$ are known, by equations (9) and (10), the attacker sets up a table $T_j$ whose item at row $x_{t+i}[j]$ and column $C[j]$ is just $w_t[j]$ for each $0 \leq j \leq 15$ before launching the attack, and looks up the table $T_j$ directly and gets $w_t[j] = T_j[x_{t+i}[j]|C[j]]$ during the execution of the attack. Note that there are 16 tables, and each table contains 256 item, each item contains 4 bits, if each item is stored by one byte, then the size of the whole memory required is $16 \cdot 256B = 4KB$. 

3.5 The time and data complexity

In section 3.1 we need to guess all possible values of \( x_t \), and for each possible value of \( x_t \), we need to compute \( C \) in equation (8) further, which is associated with \( S_{0,t+i} \ (i = 0, 1, \cdots, 127) \). After the \( C \) is known, we need to look up the table \( T_j \) 16 times in order to recover \( w_t \). From the viewpoint of hardware implementations, the last operation on looking up the tables to get \( w_t \) consumes a negligible time, thus we ignore it during the time complexity evaluation of our attack. Since \( x_i \) has \( 2^{64} \) possible values, thus our time complexity is \( 2^{64} \times T_c \) in the worst case, where \( T_c \) is the time consuming during the computation of \( C \), which is a very small constant.

In order to compute \( C \), we need to compute \( S_{0,t+i} \ (i = 0, 1, 2, \cdots, 127) \). The latter needs about 132 known plaintext/ciphertext pairs. Thus the data complexity of our attack is very low.

3.6 A forgery attack

Let \((C, T)\) be the ciphertext and the authentication tag transported in some communication session. If an attacker has known a small phase of plaintext \( P \) which corresponds to some phase of the ciphertext \( C \), then he can recover all corresponding plaintext of the ciphertext \( C \) and forge arbitrary legal ciphertext \( C' \) and the authentication tag \( T' \), where we assume that the plaintext \( P \) contains at least 132 of 64-bit blocks. That is because: based on the above attack, first the attacker recovers the state of PANDA-s at the beginning of processing the plaintext \( P \) with the plaintext/ciphertext pairs \((P, C)\); second, since the update of the state of PANDA-s is invertible, he further recovers the initial state of PANDA-s in the process of encryption and decrypts the ciphertext \( C \) to get the whole plaintext \( P \); finally, the attacker chooses an arbitrary plaintext \( P' \) and encrypts them with the recovered initial state to get \( C' \) and further generates the tag \( T' \). The attacker sends the message \((C', T')\) to a legal receiver (note: he has the legal secret key). The receiver decrypts \( C' \) and verifies \( T' \) to get \( P' \).

4 Conclusion

In this work we present a state recovery attack against PANDA-s with time complexity about \( 2^{64} \) and data complexity at most 132 pairs of plaintext/ciphertext. Based on the state recovery attack, we further deduce a forgery attack against PANDA-s. The results show that PANDA-s is far from the goal of its security design.

References


A The constants $C_1, C_2, \ldots, C_6$

The bit representation is with regard to the primitive element $\theta$, and the most significant bit is at the left.

$$C_1 = \begin{array}{c} 10000011011100001001000011010000010011000011010000001001101001001 \end{array}$$
$$C_2 = \begin{array}{c} 00111000011011110000101111100111101101100010011100001110011011011100 \end{array}$$
$$C_3 = \begin{array}{c} 1000001101111000011000000011110000111000011111111100111001000100110110000 \end{array}$$
$$C_4 = \begin{array}{c} 1110110000101111110000111000011001100000110100000011010011001001 \end{array}$$
$$C_5 = \begin{array}{c} 1100011001000010111111111111111000011000100100111001111101101001 \end{array}$$
$$C_6 = \begin{array}{c} 100000110111100001100001110000111000011111111110000011111111111111111111111 \end{array}$$