A Forgery Attack against PANDA-s

Yu Sasaki and Lei Wang

NTT Secure Platform Laboratories, Japan
sasaki.yu@lab.ntt.co.jp
Nanyang Technological University, Singapore
Wang.Lei@ntu.edu.sg

Abstract. PANDA is an authenticated encryption scheme designed by Ye *et al.*, and submitted to the CAESAR competition. The designers claim that PANDA-s, which is one of the designs of the PANDA-family, provides 128-bit security in the nonce misuse model. In this note, we describe our forgery attack against PANDA-s. Our attack works in the nonce misuse model. It exploits the fact that the message processing function and the finalization function are identical, and thus a variant of the length-extension attack can be applied. We can find a tag for a pre-specified formatted message with 2 encryption oracle calls, 2^{64} computational cost, and negligible memory.

Key words: PANDA, Forgery Attack, Nonce Misuse

1 Specification of PANDA-s

PANDA-s is one of the designs of the PANDA-family designed by Ye *et al.* [1]. PANDA-s encryption function takes a 128-bit key K, a 128-bit nonce N, variable length associated data A, and variable length plaintext P as input, and outputs the corresponding ciphertext C and a 128-bit tag T.

The encryption function consists of 4 parts: initialization, processing associated data, processing plaintext, and finalization, which are computed in this order. The computation structure is illustrated in Fig. 1 and Fig. 2, where the bit size of each arrow line in those figures is 64 bits. 64-bit values are called "blocks" in PANDA.

Initialization. In the initialization part, a 128-bit key K and a 128-bit nonce N are mixed and expanded to 448-bit internal state. We omit the details due to the irrelevance to our attack.

Processing Associated Data. The associated data A is first padded to a multiple of 64 bits $(A_0, A_1, \ldots, A_{s-1})$, and then processed block by block with the round function RF. The round function RF of PANDA-s generally takes an 8-block (or 512-bit) value as input, of which 448 bits are for the previous internal state value and 64 bits are for processing other data. The output of RF is either

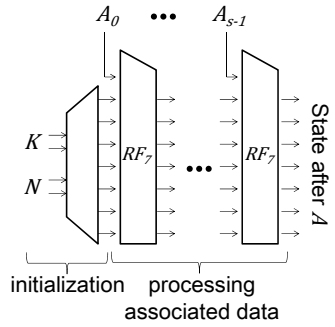

Fig. 1. Initialization and Associated Data Processing

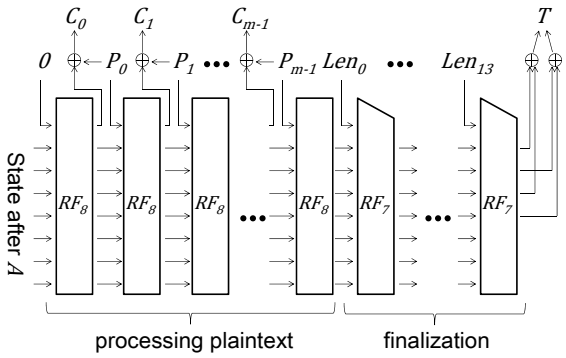

Fig. 2. Plaintext Processing and Finalization

a 7-block value (updated internal state) or a 8-block value (updated internal state and 1-block key stream). We denote the round function by RF_7 when the output size is 7 blocks, and by RF_8 when the output size is 8 blocks.

In RF_7, a 7-block internal state value is split into seven 1-block variable $w, x, y, z, s^{(0)}, s^{(1)}, s^{(2)}$. Let m be another 1-block input value. Then, the updated state value $w', x', y', z', s'^{(0)}, s'^{(1)}, s'^{(2)}$ are computed as follows, which is also illustrated in Fig. 3.

$$\begin{aligned} w' &\leftarrow \texttt{SubNibbles}(w \oplus x \oplus m) \\ x' &\leftarrow \texttt{SubNibbles}(x \oplus y) \\ y' &\leftarrow \texttt{SubNibbles}(y \oplus z) \\ z' &\leftarrow \texttt{SubNibbles}(s^{(0)}) \\ (s'^{(0)}, s'^{(1)}, s'^{(2)}) &\leftarrow \texttt{LinearTrans}(s^{(0)} \oplus w, s^{(1)}, s^{(2)}), \end{aligned}$$

where `SubNibbles` applies a 4-bit S-box and `LinearTrans` applies a linear transformation. We omit the details due to the irrelevance to our attack.

Finally, by taking the 7-block state value after the initialization, *state*, as input, the associated data is processed by computing $RF_7(state, A_i)$ for $i = 0, 1, \ldots, s-1$.

Processing Plaintext. The plaintext P is first padded to a multiple of 64 bits $(P_0, P_1, \ldots, P_{m-1})$, and then processed block by block with the round function RF_8. RF_8 is almost the same as RF_7. The only difference is that it produces another 1-block output value r by $r \leftarrow x \oplus x'$. The computation of RF_8 is illustrated in Fig. 4. The additional 1-block output value r is used as a key stream. Namely, the ciphertext block C_i for the plaintext block P_i is computed by $C_i \leftarrow P_i \oplus r$. Finally, by taking the 7-block state value after the associated data

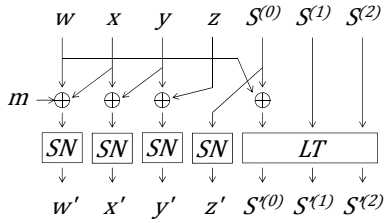

Fig. 3. Round function with 7-block output
SN and LT stand for `SubNibbles` and `Linear Transform`, respectively.

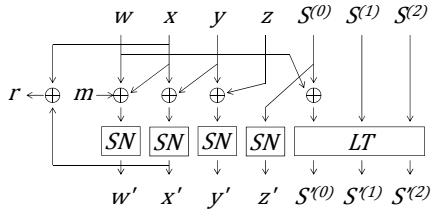

Fig. 4. Round function with 8-block output

processing, $state$, as input, the plaintext is processed by computing as follows:

$$
\begin{aligned}
&(state, r_0) \leftarrow RF_8(state, 0),\\
&\texttt{for } i = 0 \texttt{ to } m-1\\
&\qquad C_i \leftarrow P_i \oplus r_i,\\
&\qquad (state, r_{i+1}) \leftarrow RF_8(state, P_i).
\end{aligned}
$$

Note that after processing the last message block P_{m-1}, the extra key stream r_m is discarded.

Finalization. In the finalization, the state is updated by using the bit length of the associated data $|A|$ and the bit length of the plaintext $|P|$. Let $tempt_i$ be $|A|$ when i is even and $|P|$ when i is odd. In short, it consists of 14-round state update by using $tempt_i$ and the tag generation. In details, the finalization computes the following operation.

$$
\begin{aligned}
&\texttt{for } i = 0 \texttt{ to } 13\\
&\qquad state \leftarrow RF_7(state, tempt_i),\\
&T \leftarrow w \oplus y \| x \oplus z.
\end{aligned}
$$

Claimed Security of `PANDA-s`. The claimed security of `PANDA-s` is given in Table 1. In particular, 128-bit security is claimed for the integrity in the nonce-misuse model.

Table 1. Bits of security goals in `PANDA-s`[1]

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

2 Forgery Attack against `PANDA-s`

In this section, we show the forgery attacks against `PANDA-s`.

2.1 Length Extension Property of `PANDA-s`

The core of our observation is as follows.

1. The internal state is update by RF_7 and RF_8 in the same way, *i.e.* $RF_7(state, m)$ and $RF_8(state, m)$ produce the same state value.
2. The 1-block input values in the finalization, $|A|$ or $|P|$, are public, and thus known to the adversary.
3. Let α be a concatenation of any associated data and any plaintext. Also let $\ell(\alpha)$ be the 14-block value that will be processed in the finalization part for the input α, *i.e.* $\ell(\alpha) = |A|, |P|, \ldots, |A|, |P|$. By querying α to the encryption oracle to obtain the corresponding tag T_1, the adversary can obtain significant information of the internal state value for the extended message $\alpha \| \ell(\alpha)$.

2.2 Message Structure

Our attack requires only 2 encryption oracle calls under the same nonce N and the same associated data A. The queried messages $A \| P$ including both associated data A and plaintext P must satisfy the following form.

$$M_1 \leftarrow \alpha,$$
$$M_2 \leftarrow \alpha \| \ell(\alpha) \| \beta \| \ell\big(\alpha \| \ell(\alpha) \| \beta\big) \| \gamma,$$

where α, and β can be any string including `Null`, and γ can be any string as long as its length is longer than or equal to 2 blocks. Then the tag for the message $\alpha \| \ell(\alpha) \| \beta$ is forged.

As you can see, except for 14 blocks of $\ell(\alpha)$ and 14 blocks of $\big(\alpha \| \ell(\alpha) \| \beta\big)$ in total 224 bytes, any message can be the target of our attack.

2.3 Attack Details

Recovering 256-bit Internal State After Processing $\alpha\|\ell(\alpha)$. The adversary first queries the message $M_1 = \alpha$ and obtains the corresponding tag T_1. This reveals some information about the internal state x, y, z, w after processing $\alpha\|\ell(\alpha)$. The adversary later appends the message block $\beta = \beta_0\|\beta_1\|\cdots$. Let the internal state value after processing $\alpha\|\ell(\alpha)$ be $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0}, S^{(0)}_{\beta_0}, S^{(1)}_{\beta_0}, S^{(2)}_{\beta_0})$. The obtained tag T_1 indicates that the internal state value satisfy the following equation.

$$w_{\beta_0} \oplus y_{\beta_0} = T_1^L, \tag{1}$$

$$x_{\beta_0} \oplus z_{\beta_0} = T_1^R, \tag{2}$$

where T_1^L and T_1^R are 64-bit values satisfying $T_1^L\|T_1^R = T_1$.

Then, the adversary queries the message $M_2 = \alpha\|\ell(\alpha)\|\beta\|\ell\big(\alpha\|\ell(\alpha)\|\beta\big)\|\gamma$, and obtains the corresponding ciphertext blocks and tag T_2. The computation to process β_0 is shown in Fig. 5. As a result of $RF_8(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0}, S^{(0)}_{\beta_0}, S^{(1)}_{\beta_0}, S^{(2)}_{\beta_0}, \beta_0)$,

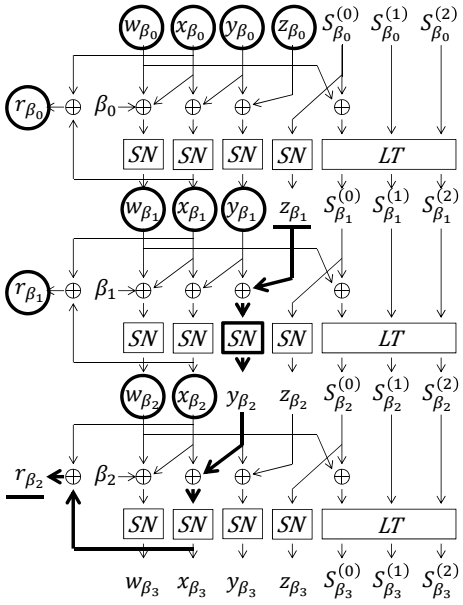

Fig. 5. 256-bit internal state recovery

the adversary obtains the ciphertext block C_{β_0}, which is computed by $r_{\beta_0} \oplus P_{\beta_0}$. Because plaintext value is known to the adversary, the key stream r_{β_0} can be computed as $P_{\beta_0} \oplus C_{\beta_0}$. From the computation structure of the key stream, the

adversary obtains the equation

$$r_{\beta_0} = x_{\beta_0} \oplus \texttt{SubNibbles}(x_{\beta_0} \oplus y_{\beta_0}). \tag{3}$$

Here, the adversary guesses the 64-bit value of x_{β_0}. For each guess, the corresponding z_{β_0} is obtained from Eq. (2), the corresponding y_{β_0} is obtained from Eq. (3), and the corresponding w_{β_0} is obtained from Eq. (1). Hence, for each guess of x_{β_0}, 256-bit internal state value $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$ is determined. Moreover, the knowledge of $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$ leads to the knowledge of $w_{\beta_1}, x_{\beta_1}, y_{\beta_1}, w_{\beta_2}, x_{\beta_2}$, and r_{β_1}. These give another 64-bit relation

$$r_{\beta_1} = x_{\beta_1} \oplus \texttt{SubNibbles}(x_{\beta_1} \oplus y_{\beta_1}), \tag{4}$$

and only 1 guess of x_{β_0} out of 2^{64} possibilities will satisfy this equation. Therefore, the 256-bit internal state value $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$ is uniquely determined. In Fig. 5, the focused variables so far are stressed by bold circles.

Recovering w, x, y, z for All Rounds. With the knowledge of $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$, the adversary aims to keep revealing the 256-bit state w, x, y, z for all rounds. This can be done with negligible cost.

In Fig. 5, to recover the 64-bit value of z_{β_1}, the adversary uses the key stream value after 1 round. Namely, the adversary focuses on the following 64-bit relation.

$$\begin{aligned} r_{\beta_2} &= x_{\beta_2} \oplus \texttt{SubNibbles}(x_{\beta_2} \oplus y_{\beta_2}), \\ &= x_{\beta_2} \oplus \texttt{SubNibbles}\big(x_{\beta_2} \oplus \texttt{SubNibbles}(y_{\beta_1} \oplus z_{\beta_1})\big). \end{aligned}$$

The above equation is converted to

$$z_{\beta_1} = y_{\beta_1} \oplus \texttt{SubNibbles}^{-1}\big(\texttt{SubNibbles}^{-1}(r_{\beta_2} \oplus x_{\beta_2}) \oplus x_{\beta_2}\big). \tag{5}$$

Then, z_{β_1} is recovered only with 1 computational cost. In Fig. 5, the focused variables to recover z_{β_1} are stress by bold lines. Moreover, by iterating the same procedure for the subsequent blocks, the adversary can recover $(w_{\beta_i}, x_{\beta_i}, y_{\beta_i}, z_{\beta_i})$ for any i as long as the key stream for the next block, $r_{\beta_{i+1}}$, is obtained.

Forging Tag. Finally, the tag for the message $\alpha\|\ell(\alpha)\|\beta$ is forged. Due to the message structure of M_2, the adversary can recover the internal state after $\alpha\|\ell(\alpha)\|\beta\|\ell\big(\alpha\|\ell(\alpha)\|\beta\big)$ is processed. Note that the length of γ must be at least 2 block so that the internal state after the last block of $A\|\alpha\|\ell(\alpha)\|\beta\|\ell\big(\alpha\|\ell(\alpha)\|\beta\big)$ can be recovered. Then, the tag for $\alpha\|\ell(\alpha)\|\beta$ is easily computed by computing $w\oplus y\|x\oplus z$ of this internal state, where we note that the nonce and the associated data are the same with previous two queries M_1 and M_2.

Complexity Evaluation. The attack requires 2 encryption oracle calls under the same key and nonce. To recover the 256-bit internal state $(w_{\beta_0}, x_{\beta_0}, y_{\beta_0}, z_{\beta_0})$, 2^{64} computational cost is required. Then, all the remaining cost is 1. The memory requirement is to store the all ciphertext blocks and the tag, which is very small.

3 Concluding Remarks

In this note, we proposed a forger attack against `PANDA-s`. Our attack can forge a tag of a message satisfying the pre-specified format with 2 encryption oracle calls, 2^{64} computational cost, and negligible memory. The attack works in the nonce-misuse model. The attack clearly breaks the security claim of `PANDA-s`, *i.e.* 128-bit security for integrity in the nonce-misuse model.

References

1. Dingfeng Ye, Peng Wang, Lei Hu, Liping Wang, Yonghong Xie, Siwei Sun, and Ping Wang. PANDA v1. Submitted to the CAESAR competition, March 2014.