PAES v1:

Parallelizable Authenticated Encryption Schemes based on AES Round Function

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

Chapter 1

Introduction

Authenticated encryption, AE for short, is a symmetric cryptographic primitive that provides protections for confidentiality and integrity (authentication) simultaneously. Simply speaking, confidentiality guarantees that an adversary can not get any information (except the length) about the plaintext from the ciphertext, whilst integrity guarantees that the ciphertext is truly delivered from the sender and not modified by an adversary during transmission.

An AE scheme can be based on some cryptographic components, such as a block cipher, a stream cipher combined with a MAC (e.g., Grain-128a [1], 128-EIA3 [10]), or a permutation in sponge structure (e.g., FIDES [3], APE [2]). The most popular approach to construct AE is to use a block cipher in a mode. The advantage of this approach is that we can prove the security of the mode in some adversarial model, given that the underlying block cipher is secure, e.g., it is a pseudorandom permutation (PRP). A large number of AE modes have been proposed and standardized, such as OCB [17], GCM [16], CCM [19], SIV [18]. Although efficient AE modes, such as OCB, can get both confidentiality and integrity by processing messages in only one-pass, they still need one block cipher-invocation per message-block. Furthermore, almost all the modes suffer from birthday attacks which reduce the bits of security from n in block cipher to $n/2$. It is not big deal for 128-bit block cipher modes, but for 64-bit modes, 32-bit strength is so weak that it is allowed to a practical attack.

The other approach to construct AE is to use the component of block cipher such as AES round function, which is also used in other schemes, for example, message authentication code Pelican [9] and stream cipher LEX [4]. The existing constructions of AE in this way include ACS-1 [14], ALE [5], Marvin [15], and AEGIS [20, 21]. Although all these schemes lack provable security against general attacks such as chosen plaintext/ciphertext attacks (CPA/CCA) as the above modes, the demonstration of security against some specific attacks such as linear/differential attacks is straightforward by utilizing some AES properties, for instance, the number of active S-boxes of four-round AES is at lest 25.

Moreover, a set of new instructions AES-NI (Advanced Encryption Standard New Instructions) is supported in recent years, first by Intel CPU and then by AMD CPU. AES-NI has six instructions: four instructions for the AES round functions, and the other two for the AES key expansion. These instructions greatly improve the software performance of AES, its modes, and the schemes based on the AES round function.

AEGIS is the fastest AE scheme so far due to its parallel structure. There are three versions of

AEGIS [21], i.e., AEGIS-128, AEGIS-256 and AEGIS-128L, which maintain a state of 5, 6 and 8 parallel blocks (1 block = 16 bytes) respectively. AEGIS-128L amazingly achieves the speed of around 0.48 cpb (cycles per byte) by using AES-NI.

Unfortunately, as its authors pointed out, AEGIS is not secure against when the nonce is reused. In other words, the initial vector (IV) of AEGIS should be used a nonce . When it is repeated, it is easy to recover the internal states of AEGIS, which leads to successful attacks against both confidentiality and integrity. The security of AEGIS relies on the assumption that IV is never repeated.

In this paper, we propose a new family of AE schemes PAES (PAES-4 and PAES-8) based on AES round function. The structures of PAES-4 and PAES-8 are similar, except that the state of PAES-4 is 4 blocks while the state of PAES-8 is 8 blocks. PAES-4 aims at using a smaller state than AEGIS-128, and PAES-8 aims at achieving an extra robustness in nonce-repeating model. Both schemes preserve the advantages of AEGIS as many as possible.

Chapter 2

Specification of PAES

PAES consists of two deterministic algorithms: the encryption algorithm PAES.Enc and the decryption algorithm PAES.Dec. The encryption algorithm PAES.Enc takes in a 128-bit key K, a 128-bit nonce N, variable-length associated data A and a variable-length plaintext P as its input, and outputs a variable-length ciphertext (C, T) where T is a 128-bit authentication tag. We write it as PAES.Enc$_K(N, A, P) = (C, T)$. The decryption algorithm PAES.Dec takes in a tuple (K, N, A, C, T) as its input and outputs an authenticated string P or a specific symbol $\perp$ indicating that the ciphertext is invalid. We require that PAES.Dec$_K(N, A, C, T) = P$ if PAES.Enc$_K(N, A, P) = (C, T)$. The nonce N is a public message number. There is no secret message number.

All the operations in PAES are defined in terms of 128-bit block. For the variable-length data such as P, C and 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 128 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.

2.1 Specification of PAES-4

2.1.1 State update functions of PAES-4

In the encryption and decryption of PAES-4, a state $State$ consisting of 4 128-bit blocks S_i, $i = 1, 2, \cdots, 4$ is always maintained. Two similar state update functions StateUpdate$_0$ and StateUpdate$_1$ are used in PAES-4. The state update functions use the 128-bit block M to update $State$. More specifically,

```
Algorithm StateUpdate$_i$($State, M$)
  if $i = 0$
     $S_4 \leftarrow S_4 \oplus S_3$
  $V_1 \leftarrow AESRound(S_2 \oplus S_4)$
  $V_2 \leftarrow AESRound(S_1)$
  $V_3 \leftarrow AESRound(S_3 \oplus S_2)$
  $V_4 \leftarrow AESRound(S_4 \oplus M)$
  return $(V_1, V_2, V_3, V_4)$
```

where $AESRound$ is the AES encryption round function, consisting of a series of three operations: SubBytes, ShiftRows and MixColumns.

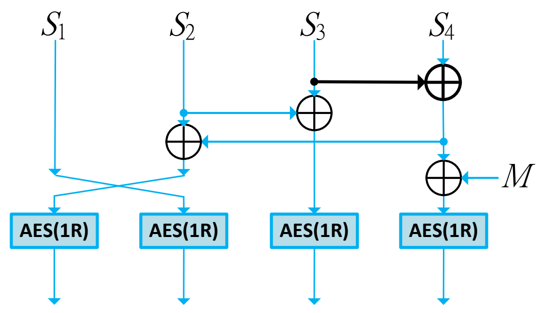

Figure 2.1: The state update function StateUpdate_0. StateUpdate_1 is illustrated without the bold line.

We illustrate the two functions in Fig. 2.1.

2.1.2 The encryption: PAES-4.Enc$_K(N, A, P)$

The encryption is divided into the following 4 steps.

1) Initialization: Upload the state $State$ with the nonce N, the key K, then run the update function StateUpdate_0 5 times, and then XOR the key to each block of the state. More specifically,

```
$S_1 \leftarrow K \oplus N$
$S_2 \leftarrow L(K) \oplus L^3(N)$
$S_3 \leftarrow L^2(K) \oplus L(N)$
$S_4 \leftarrow L^3(K) \oplus L^2(N)$
for $i = 1$ to $5$
  $State \leftarrow \texttt{StateUpdate}_0(State, 0)$
for $i = 1$ to $4$
  $S_i \leftarrow S_i \oplus K$
```

where L is defined as $L(a, b, c, d) = (b, c, d \oplus a, a)$, a, b, c, d are all 32-bit strings.

2) Processing associated data: After padding and partition, the associated data A becomes $A_1A_2 \cdots A_s$, a string of 128-bit blocks. Use the block A_i to update $State$ with StateUpdate_1. More specifically,

```
for $i = 1$ to $s$
  $State \leftarrow \texttt{StateUpdate}_1(State, A_i)$
```

3) Processing plaintext: After padding and partition, the plaintext P becomes $P_1P_2 \cdots P_m$, a string of 128-bit blocks. Output the XOR of S_3 and its previous block S'_3 as a key stream to encrypt P_i, and then use the block P_i to update $State$ with StateUpdate_1. More specifically,

```
$State \leftarrow \texttt{StateUpdate}_1(State, 0)$
for $i = 1$ to $m$
  $R_i \leftarrow S_3 \oplus S'_3$
  $C_i \leftarrow P_i \oplus R_i$
  $State \leftarrow \texttt{StateUpdate}_1(State, P_i)$
```

We illustrate this step in Fig. 2.2.

4) Finalization: Use $tempt = adlen||mslen$ to update $State$ with StateUpdate_0 5 times, and then output the XOR of the last two state blocks as the authentication tag T, where $adlen$ and $mslen$, both 64-bit, are the bit-length of the associated data and the plaintext respectively. More specifically,

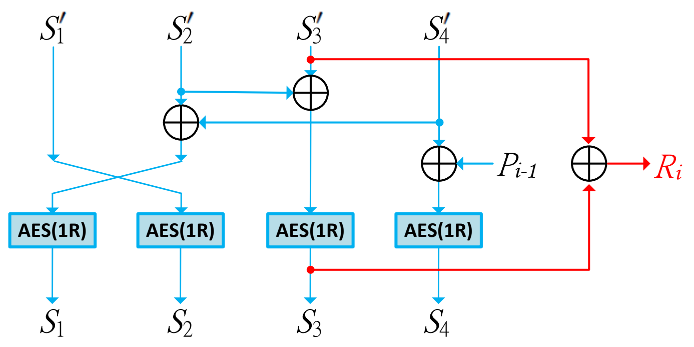

Figure 2.2: Processing plaintext in PAES-4, where $P_0 = 0$, $R_i, i = 1, 2, \cdots m$ is the key stream.

```
for $i = 1$ to $5$
  $State \leftarrow$ StateUpdate$_0(State, tempt)$
$T \leftarrow S_3 \oplus S_4$.
```

2.1.3 The decryption: PAES-4.Dec$_K(N, A, C, T)$

The steps of initialization and processing associated data are the same as in the encryption. After padding and partition, the ciphertext C becomes $C_1C_2 \cdots C_m$,

```
$State \leftarrow$ StateUpdate$_1(State, 0)$
for $i = 1$ to $m$
  $R_i \leftarrow S_3 \oplus S_3'$
  $P_i \leftarrow C_i \oplus R_i$
  $State \leftarrow$ StateUpdate$_1(State, P_i)$
```

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.

2.2 Specification of PAES-8

2.2.1 State update functions of PAES-8

In the encryption and decryption of PAES, a state $State$ consisting of 8 128-bit blocks S_i, $i = 1, 2, \cdots, 8$ is always maintained. Two similar state update functions StateUpdate$_0$ and StateUpdate$_1$ are used in PAES. The state update functions use the block M to update $State$. More specifically,

```
Algorithm StateUpdate$_i(State, M)$
  if $i = 0$
     $S_8 \leftarrow S_8 \oplus S_7$
  $V_1 \leftarrow AESRound(S_6 \oplus S_8)$
  $V_2 \leftarrow AESRound(S_1)$
  $V_3 \leftarrow AESRound(S_2)$
  $V_4 \leftarrow AESRound(S_3)$
  $V_5 \leftarrow AESRound(S_4)$
  $V_6 \leftarrow AESRound(S_5)$
  $V_7 \leftarrow AESRound(S_7 \oplus S_6)$
```

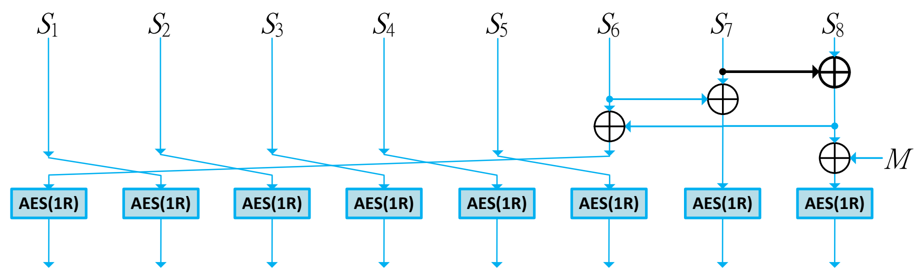

Figure 2.3: The state update function StateUpdate_0, and StateUpdate_1 is illustrated without the bold line.

```
  $V_8 \leftarrow AESRound(S_8 \oplus M)$
  return $(V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8)$
```

We illustrate the two functions in Fig. 2.3.

2.2.2 The encryption: PAES-8.Enc$_K(N, A, P)$

The encryption is divided into the following 4 steps.

1) Initialization: Upload $State$ with the nonce N and the key K, then run the update function StateUpdate_0 10 times, and then XOR the key to each block of the state. More specifically,

```
$S_1 \leftarrow K \oplus N$
$S_2 \leftarrow L(K) \oplus L^3(N)$
$S_3 \leftarrow L^2(K) \oplus L(N)$
$S_4 \leftarrow L^3(K) \oplus L^2(N)$
$S_5 \leftarrow L^4(K) \oplus L^7(N)$
$S_6 \leftarrow L^5(K) \oplus L^4(N)$
$S_7 \leftarrow L^6(K) \oplus L^5(N)$
$S_8 \leftarrow L^7(K) \oplus L^6(N)$
for $i = 1$ to $10$
  $State \leftarrow \texttt{StateUpdate}_0(State, 0)$
for $i = 1$ to $8$
  $S_i \leftarrow S_i \oplus K$
```

where L is the same as defined in PAES-4.

2) Processing associated data: After padding and partition, the associated data A becomes $A_1A_2 \cdots A_s$. Use the block A_i to update $State$ with StateUpdate_1. More specifically,

```
for $i = 1$ to $s$
  $State \leftarrow \texttt{StateUpdate}_1(State, A_i)$
```

3) Processing plaintext: After padding and partition, the plaintext P becomes $P_1P_2 \cdots P_m$. Output the XOR of S_7 and its previous block S'_7 as a key stream to encrypt the plaintext P_i, and then use the block P_i to update $State$ with StateUpdate_1. More specifically,

```
$State \leftarrow \texttt{StateUpdate}_1(State, 0)$
for $i = 1$ to $m$
  $R_i \leftarrow S_7 \oplus S'_7$
  $C_i \leftarrow P_i \oplus R_i$
  $State \leftarrow \texttt{StateUpdate}_1(State, P_i)$
```

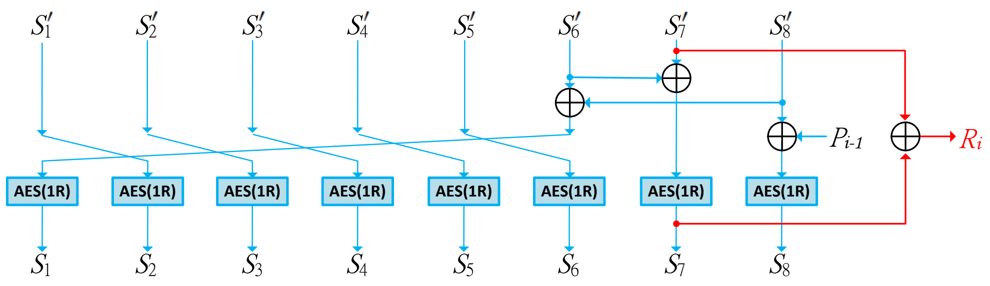

Figure 2.4: Processing plaintext in PAES-8.

We illustrate this step in Fig. 2.4.

4) Finalization: Use $tempt = adlen||mslen$ to update S with StateUpdate$_0$ 10 times, and then output the XOR of the last two state blocks as the authentication tag T, where $adlen$ and $mslen$, both 64-bit, are the bit-length of the associated data and the plaintext respectively. More specifically,

```
for $i = 1$ to $10$
  $State \leftarrow \texttt{StateUpdate}_0(State, tempt)$
$T \leftarrow S_7 \oplus S_8$
```

2.2.3 The decryption: PAES-8.Dec$_K(N, A, C, T)$

The steps of initialization and processing associated data are the same as those in the encryption. After padding and partition, the ciphertext C becomes $C_1 C_2 \cdots C_m$,

```
$State \leftarrow \texttt{StateUpdate}_1(State, 0)$
for $i = 1$ to $m$
  $R_i \leftarrow S_7 \oplus S'_7$
  $P_i \leftarrow C_i \oplus R_i$
  $State \leftarrow \texttt{StateUpdate}_1(State, P_i)$
```

The generation of authentication tag in the finalization in 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.

Chapter 3

Security goals

The security goals of PAES are listed in Table 3.1.

Goal	Nonce-respecting Model	Nonce-repeating Model	
	PAES-4/8	PAES-4	PAES-8
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

Table 3.1: Bits of security goals in two models

Assumptions. We assume that the key of PAES 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 PAES 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 PAES fails.

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

1) Recovering the key or the state of PAES;
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: PAES-4 in Model 1, PAES-8 in Model 1 and 2.

2) Resistance against state recovery: PAES-4 in Model 1, PAES-8 in Model 1 and 2.
3) Resistance against plaintext recovery: PAES-4 in Model 1, PAES-8 in Model 1 and 2.
4) Resistance against linear distinguishing attack: PAES-4 in Model 1, PAES-8 in Model 1.
5) Resistance against forgery attack: PAES-4 in Model 1, and PAES-8 in Model 1 and 2.
6) All the above resistances have 128 bits of security.

Details of security analysis are given in the next chapter.

Chapter 4

Security analysis

The main result of this section is that the total number of active AES S-boxes is at least $25/50$ in linear or differential attacks against PAES-4/8, since a sequence of chained four/eight AES round functions is involved.

4.1 Linear analysis

A linear attack [6] exploits the fact that some linear combinations of the input and output of the non-linear component are biased. These linear combinations are called *linear approximations* of the non-linear function. We view the AES round function as the non-linear component in PAES. 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 against stream cipher 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 the key stream and τ_i are the linear masks, which are obtained by using some linear approximations of the AES round functions. Assume that the state during the encryption of PAES-4 is $State_i = (S_{1i}, S_{2i}, S_{3i}, S_{4i})$, $i = 1, 2, \cdots$, the key stream $R_i = S_{3i} \oplus S_{3(i+1)}$, $i = 1, 2, \cdots$, and the linear masks of input and output of AES round functions are α_i, α_i', β_i, β_i', γ_i, γ_i', δ_i, δ_i', $i = 1, 2, \cdots$, which are illustrated in Fig. 4.1.

We turn the idea above into the following equation in terms of the variables of state:

$$\begin{aligned}\sum_{i=1}^{s} \tau_i \cdot (S_{3i} + S_{3(i+1)}) = \sum_{i=1}^{s} (&\alpha_i \cdot (S_{2i} + S_{4i}) + \alpha_i' \cdot S_{1(i+1)} \\ &+ \beta_i \cdot S_{1i} + \beta_i' \cdot S_{2(i+1)} \\ &+ \gamma_i \cdot (S_{2i} + S_{3i}) + \gamma_i' \cdot S_{3(i+1)} \\ &+ \delta_i \cdot S_{4i} + \delta_i' \cdot S_{4(i+1)}),\end{aligned}$$

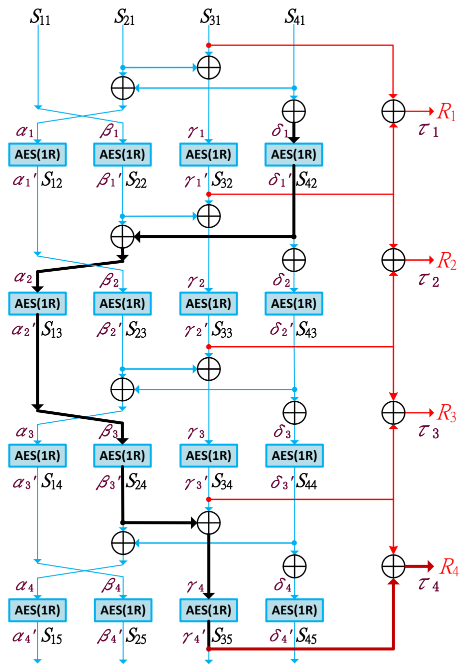

Figure 4.1: Notations in the linear attack against PAES-4.

or

$$\begin{aligned}
&\beta_1 \cdot S_{11} + (\alpha_1+\gamma_1)\cdot S_{21} + (\gamma_1+\tau_1)\cdot S_{31} + (\alpha_1+\delta_1)\cdot S_{41}\\
&+\sum_{i=1}^{s-1}((\alpha'_i+\beta_{i+1})\cdot S_{1(i+1)} + (\beta'_i+\alpha_{i+1}+\gamma_{i+1})\cdot S_{2(i+1)}\\
&\qquad + (\gamma'_i+\tau_i+\gamma_{i+1}+\tau_{i+1})\cdot S_{3(i+1)} + (\delta'_i+\alpha_{i+1}+\delta_{i+1})\cdot S_{4(i+1)})\\
&+\alpha'_s\cdot S_{1(s+1)} + \beta'_s\cdot S_{2(s+1)} + (\gamma'_s+\tau_s)\cdot S_{3(s+1)} + \delta'_s\cdot S_{4(s+1)} = 0
\end{aligned}$$

In order to get the approximation, the variables of state must vanish. We have the following equations:

$$\begin{cases} \beta_1 = 0\\ \alpha_1+\gamma_1=0\\ \gamma_1+\tau_1=0\\ \alpha_1+\delta_1=0, \end{cases} \quad \begin{cases} \alpha'_s=0\\ \beta'_s=0\\ \gamma'_s+\tau_s=0\\ \delta'_s=0, \end{cases} \quad \begin{cases} \alpha'_i+\beta_{i+1}=0\\ \beta'_i+\alpha_{i+1}+\gamma_{i+1}=0\\ \gamma'_i+\tau_i+\gamma_{i+1}+\tau_{i+1}=0\\ \delta'_i+\alpha_{i+1}+\delta_{i+1}=0, \quad i=1,\cdots,s-1. \end{cases}$$

Suppose that $\tau_s \neq 0$, then we can deduce from the above equations that $s \geq 4$, $\gamma_s = \beta'_{s-1}$, $\beta_{s-1} = \alpha'_{s-2}$, $\alpha_{s-2} = \delta'_{s-3}$, which define a sequence of chained four AES round functions (as illustrated in Fig. 4.1 by the bold line when $s = 4$). Therefore at least 25 active AES S-boxes are involved in the linear attack.

As to PAES-8, a similar argument can prove that in the linear attack against PAES-8, at least 50 active AES S-boxes are involved.

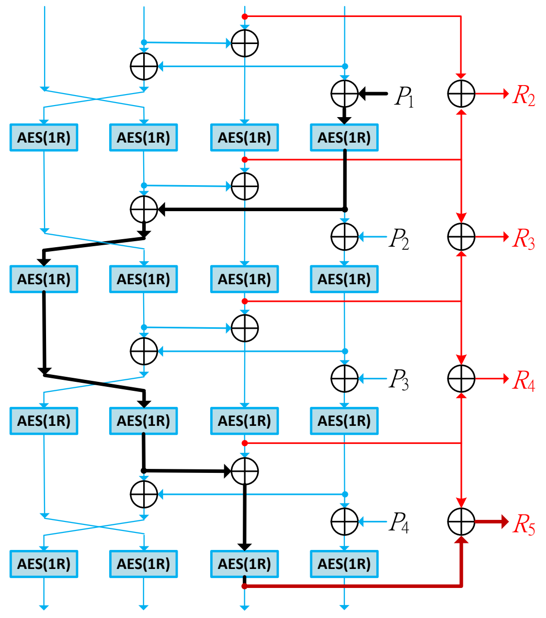

Figure 4.2: Differential path of PAES-4.

4.2 Differential analysis

In PAES-4, the difference of any plaintext block, e.g., P_1, will go through a sequence of four AES round functions until it reaches the output ciphertext and the tag. The bold line in Fig. 4.2 illustrates the differential path starting from P_1.

The differential path has two meanings. In nonce-respecting model, although the adversary can not get the difference from plaintext to ciphertext, or the difference of (P, C, T) for short, under the same nonce, he can exploit the difference of (C, T) to make a forgery attack. In nonce-repeating model, the adversary can choose arbitrary plaintext under the same nonce, and use the difference of (P, C, T) to recover the key or the state.

In AEGIS, the difference of (C, T) involves a sequence of five AES round functions, but the difference of (P, C) only involves one AES round function. That is why AEGIS is not secure in nonce-repeating model.

PAES-4 is not strong enough for state recovery attack in nonce-repeating model. That is why we propose PAES-8, in which any difference of the plaintext will go through a sequence of eight AES round functions, as illustrated in Fig. 4.3. If we regard the unknown state blocks related to this differential path as sub-keys, the state recovery attack is just the differential attack against 8-round AES. The best attack against AES-128 that does not exploit the property of key scheme is 7 rounds [11, 7] as we know, and so we believe that PAES-8 is secure against state or key recovery attack.

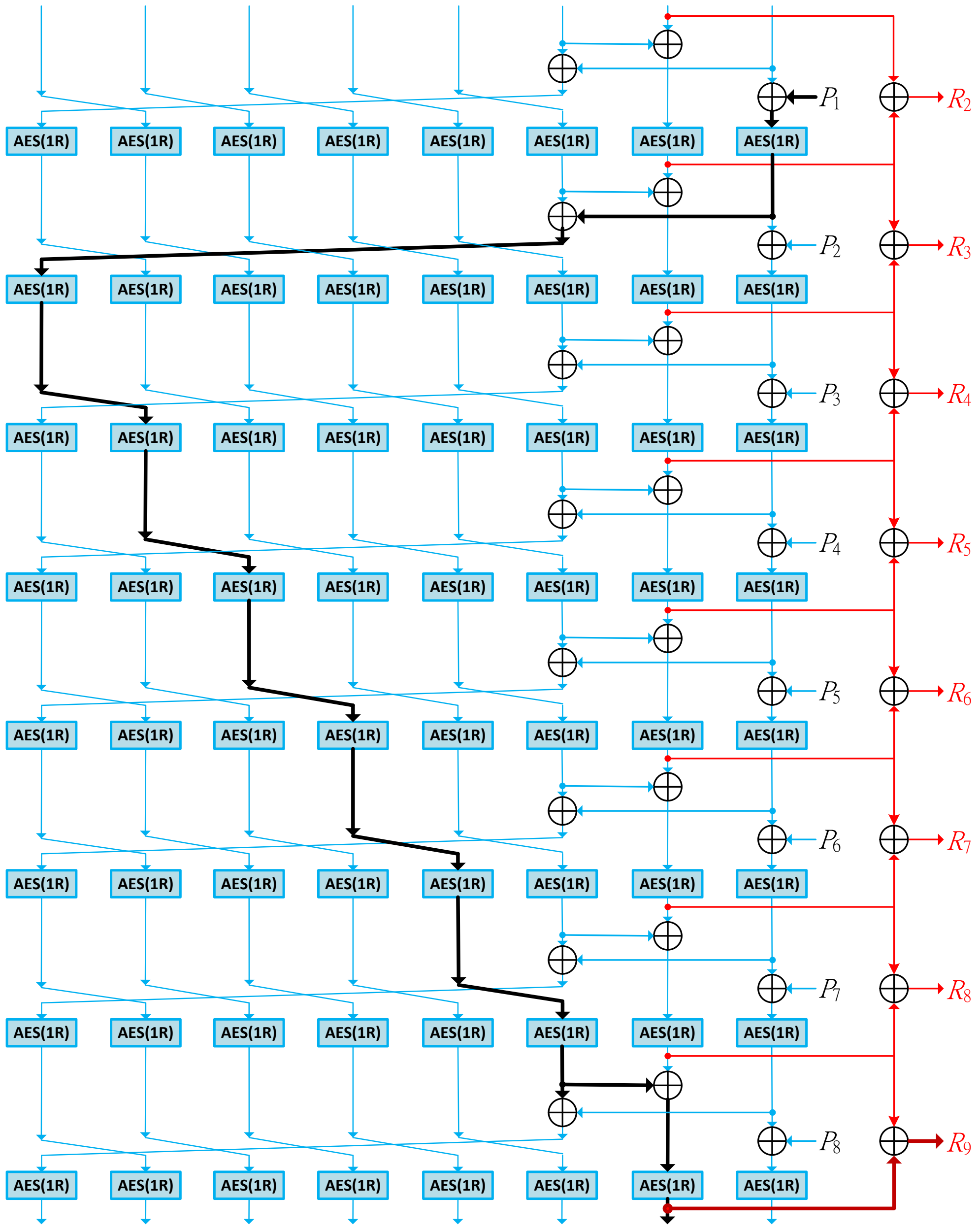

Figure 4.3: Differential path of PAES-8.

Chapter 5

Features

We compare AEGIS, PAES and GCM in Table 5.1.

	State	Efficiency	Security analysis		Misuse resistance	Birthday bound
			linear analysis	differential analysis		
AEGIS-128	5	5	No	partial	No	No
AEGIS-256	6	6	No	partial	No	No
AEGIS-128L	8	4	No	partial	No	No
PAES-4	4	4	Yes	YES	No	No
PAES-8	8	8	Yes	YES	partial	No
GCM	1	10	Reduction proof		No	Yes

Table 5.1: Comparison of AEGIS, PAES and GCM. Here the state is measured by the number of 128-bit blocks, the efficiency is measured by the number of AES round functions needed to handle a message block, and robustness is measured in the nonce-repeating model.

PAES preserves the advantages of AEGIS such as parallel structure, fast software performance, etc. There are two features we want to emphasize.

Security analysis. PAES 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. A sequence of chained AES round functions is involved our analyses, which makes the security proof straightforward by using some property of AES.

Misuse resistance. The design of PAES-8 provides some intermediate level of robustness against nonce reuse. Although, as a steam cipher, the security of confidentiality no longer exits when the nonce is reused, the differential property of PAES-8 makes it strong enough against forgery attacks.

Chapter 6

Design rationale

We designed the PAES scheme according to the following rationales:

Use of the pipeline of AES-NI. AES-NI, short for Advanced Encryption Standard New Instructions, greatly improves the software performance of cryptographic systems. Moreover, the parallel calls of the same instruction in a pipeline can further accelerate the performance. In Intel's white paper on AES-NI [13], the experiment on Westmere CPU shows that the performance of AES Encryption in the non-parallel CBC mode and the parallel CTR mode are about 1.38 and 4.15 cpb respectively. In recent Sandy and Ivy Bridge CPUs [12], the latency and throughput of the instruction AESDES corresponding to the AES round function used in our schemes are 8 and 1 cycles respectively. So eight divided instructions of AESDES take $(8 \times 8) = 64$ cycles, but eight instructions of AESDES in pipeline take only $(8 + 1 \times 7 =) 15$ cycles. Our design always puts the AES round functions in parallel, in order to take advantage of the pipeline of AES-NI.

Use of characteristics of AES. One of advantages of AES is that its security against some classic attacks, such as linear or differential attacks, can be proven in a standard way. Following the wide trail design strategy, it is showed in [8] that 4-round AES has at least 25 active S-boxes in linear or differential attacks. Our design makes the sequence of the AES round functions appear in the classic attacks as long as possible. It is 4 rounds in PAES-4 and 8 rounds in PAES-8 under both linear attack and differential attack, which makes the security proof straightforward.

Achieving robustness from architecture. AEGIS does not provide any robustness against nonce reuse attack. It is easy to recover the internal states of AEGIS [21] when nonce is repeated, leading to successful attacks against both confidentiality and integrity. Our design aims at providing some robustness in this case. The crucial point of the design goes to how to choose a proper architecture for the scheme. The final choice of architecture has some flexibility in size of the state. The robustness increases as the state becomes larger.

The designers have not hidden any weaknesses in PAES.

Chapter 7

Intellectual property

We have not applied for any patents on PAES. We explicitly release any intellectual property rights to PAES 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 PAES.

Bibliography

[1] Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of Grain-128 with optional authentication. IJWMC 5(1), 48–59 (2011) 1

[2] 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

[3] 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

[4] Biryukov, A.: A new 128-bit key stream cipher LEX. ECRYPT stream cipherproject report 2005/013 (2005) 1

[5] Bogdanov, A., Mendel, F., Regazzoni, F., Tischhauser, E., Rijmen, V.: ALE: AES-based lightweight authenticated encryption. Lecture Notes in Computer Science (2013) 1

[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) 10

[7] Daemen, J., Rijmen, V.: The first 10 years of advanced encryption. Security Privacy, IEEE 8(6), 72–74 (Nov 2010) 12

[8] Daemen, J., Rijmen, V.: The design of Rijndael: AES-the advanced encryption standard. Springer (2002) 15

[9] Daemen, J., Rijmen, V.: The Pelican MAC function. IACR Cryptology ePrint Archive 2005, 88 (2005) 1

[10] ETSI/SAGE: Specification of the 3GPP confidentiality and integrity algorithms 128-EEA3 & 128-EIA3. Document 1: 128-EEA3 and 128-eia3 specification. Version 1.7, July 2012 (2012) 1

[11] Gilbert, H., Minier, M.: A collision attack on 7 rounds of rijndael. In: AES Candidate Conference. pp. 230–241 (2000) 12

[12] Intel: 64 and IA-32 architectures optimization reference manual (2012), http://www.intel.com/content/dam/doc/manual/64-ia-32-architectures-optimization-manual.pdf 15

[13] Intel: Advanced Encryption Standard (AES) instructions set - rev 3.01 (2012), http://software.intel.com/sites/default/files/article/165683/aes-wp-2012-09-22-v01.pdf 15

[14] 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) 1

[15] Jr., M.A.S., d'Aquino F. F. S. Barbuda, P., Barreto, P.S.L.M., Carvalho, T.C.M.B., Margi, C.B.: The marvin message authentication code and the lettersoup authenticated encryption scheme. Security and Communication Networks 2(2), 165–180 (2009) 1

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

[17] 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

[18] Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 4004, pp. 373–390. Springer (2006) 1

[19] Whiting, D., Housley, R., Ferguson, N.: Counter with CBC-MAC (CCM) (2002), http://csrc.nist.gov/groups/ST/toolkit/BCM/ 1

[20] Wu, H., Preneel, B.: AEGIS: A fast authenticated encryption algorithm. In: SAC 2013. Springer (2013) 1

[21] 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 1, 2, 15