Minalpher v1.1

Designers
Yu Sasaki, Yosuke Todo, Kazumaro Aoki,
Yusuke Naito, Takeshi Sugawara, Yumiko Murakami, Mitsuru Matsui,
Shoichi Hirose

Submitters
NTT Secure Platform Laboratories,
Mitsubishi Electric Corporation,
University of Fukui

Contact
minalpher@lab.ntt.co.jp

Version 1.1: 29 August 2015

Changes

From v1.0 to v1.1

- Clarified the handling of illegal input length, specially, when a ciphertext length is not a multiple of a block.
- Updated the numbers for software implementation on x86_64.
- Added references which are found after the submission of v1.0.
- Corrected typos.

1 Specification

Section 1.1 specifies the mode of operation of Minalpher. Section 1.2 shows the exact parameters in our design. Section 1.3 defines the maximum input message length to Minalpher. Section 1.4 specifies the underlying primitive in Minalpher.

1.1 Mode of Operation

Minalpher supports two functionalities: authenticated encryption with associated data (AEAD) and message authentication code (MAC). In this section, we specify these modes of operation.

Subsection 1.1.1 gives notations used in the modes of operation. Subsection 1.1.2 specifies a padding rule used in the modes of operation. Subsection 1.1.3 specifies a primitive which we call tweakable Even-Mansour. The construction dates back to Kurosawa [27, 28], who builds a tweakable block-cipher from a permutation-based block-cipher proposed by Even and Mansour [18]. Tweakable block-ciphers are originated by Liskov, Rivest and Wagner [33]. Subsection 1.1.4 specifies the AEAD mode of operation. Subsection 1.1.5 specifies the MAC mode of operation. These modes are inspired by OCB [45] and PMAC [9].

1.1.1 Notations

We denote the length in bits of a bit string a by $|a|$. We denote the i-th bit of a bit string a by a_i where i is from 0 to $|a| - 1$. For example, $a = a_{n-1} \cdots a_1 a_0$ when $|a| = n$. We denote an empty string by ε.

Let $\mathrm{GF}(2^n)$ denote the field with 2^n points. To add two points in $\mathrm{GF}(2^n)$, take their bitwise xor. We denote this operation by $a \oplus b$. To multiply two points a, b in $\mathrm{GF}(2^n)$, fix a primitive polynomial $g(\mathrm{y})$. We denote this operation ab.

1.1.2 Padding Rule

We use the following padding rule: the padding of a bit string X to a sequence of n-bit blocks is denoted by $X \| \mathsf{pad}(|X|)$, where $\left|X \| \mathsf{pad}(|X|)\right|$ is a multiple of n.

The modes of operation of AEAD and of MAC make use of the following padding.

Definition 1. *The padding appends a single bit* 1 *followed by the minimum number of bits* 0 *such that the length of the result is a multiple of the block length.*

1.1.3 Tweakable Even-Mansour

The modes of operation of Minalpher are based on a primitive which we call tweakable Even-Mansour.

It is a tweakable block-cipher based on a permutation, denoted by P, which operates on n bits such that $n \bmod 2 = 0$. We denote the inverse by P^{-1}. It consists of two algorithms, the encryption algorithm, denoted by TEM_Enc, and the decryption algorithm, denoted by TEM_Dec.

TEM_Enc is specified in Algorithm 1, which accepts the following inputs,

- a secret key, denoted by K, such that $K \in \{0,1\}^{n/2}$,
- tweaks, denoted by (flag, N, i, j), such that
  - $\mathsf{flag} \in \{0,1\}^s$
  - $N \in \{0,1\}^{n/2-s}$
  - i and j are integers such that $(i,j) \neq (0,0)$, and
- a message, denoted by M, such that $M \in \{0,1\}^n$,

and returns an n-bit value C.

TEM_Dec is specified in Algorithm 2, which is used in the decryption algorithm of the AEAD mode of operation. TEM_Dec accepts the following inputs,

- a secret key, denoted by K, such that $K \in \{0,1\}^{n/2}$,
- tweaks, denoted by (flag, N, i, j), such that
  - $\mathsf{flag} \in \{0,1\}^s$
  - $N \in \{0,1\}^{n/2-s}$
  - i and j are integers such that $(i,j) \neq (0,0)$, and
- a ciphertext, denoted by C, such that $C \in \{0,1\}^n$,

and returns an n-bit value M.

Algorithm 1 Encryption Algorithm of Tweakable Even-Mansour

procedure TEM_Enc($K, \mathsf{flag}, N, i, j, M$)
  $L \leftarrow (K\|\mathsf{flag}\|N) \oplus P(K\|\mathsf{flag}\|N)$
  $C \leftarrow \mathsf{y}^i(\mathsf{y}+1)^j L \oplus P(M \oplus \mathsf{y}^i(\mathsf{y}+1)^j L)$
  return C
end procedure

Algorithm 2 Decryption Algorithm of Tweakable Even-Mansour

procedure TEM_Dec($K, \mathsf{flag}, N, i, j, C$)
  $L \leftarrow (K\|\mathsf{flag}\|N) \oplus P(K\|\mathsf{flag}\|N)$
  $M \leftarrow \mathsf{y}^i(\mathsf{y}+1)^j L \oplus P^{-1}(C \oplus \mathsf{y}^i(\mathsf{y}+1)^j L)$
  return M
end procedure

1.1.4 AEAD Mode

The AEAD mode of operation consists of two algorithms, the encryption algorithm, denoted by AEAD_Enc, and the decryption algorithm, denoted by AEAD_Dec. Let ℓ be the tag size such that $0 \leq \ell \leq n$. Let $\mathsf{flag}_{\mathsf{ad}}$ and $\mathsf{flag}_{\mathsf{m}}$ be constant values of s bits such that $\mathsf{flag}_{\mathsf{ad}} \neq \mathsf{flag}_{\mathsf{m}}$.

AEAD_Enc is specified in Algorithm 5. It uses two algorithms, AEAD_CGen specified in Algorithm 3 and AEAD_TGen specified in Algorithm 4. AEAD_Enc accepts the following inputs,

- a secret key, denoted by K, such that $K \in \{0,1\}^{n/2}$,
- a nonce, denoted by N, such that $N \in \{0,1\}^{n/2-s}$,
- associated data, denoted by A, such that $A \in \{0,1\}^*$, and
- a message, denoted by M, such that $M \in \{0,1\}^*$,

and returns either

- the ciphertext, denoted by C, and the tag, denoted by tag, or
- the reject symbol, denoted by $\perp$.

Note that AEAD_Enc does not accept inputs that do not satisfy the above conditions. For the inputs, AEAD_Enc returns $\perp$.

Algorithm 3 Ciphertext Generation Algorithm of AEAD Mode

procedure AEAD_CGen(K, N, M)
  Parse $M\|\,\mathsf{pad}(|M|)$ into n-bit blocks $(M[1], \ldots, M[m])$
  for $i = 1$ to m **do**
    $C[i] \leftarrow$ TEM_Enc($K, \mathsf{flag}_{\mathsf{m}}, N, 2i-1, 0, M[i]$)
  end for
  $C \leftarrow C[1]\| \cdots \|C[m]$
  return C
end procedure

AEAD_Dec is specified in Algorithm 7. It uses two algorithms, AEAD_MGen specified in Algorithm 6 and AEAD_TGen specified in Algorithm 4. AEAD_Dec accepts the following five inputs,

- a secret key, denoted by K, such that $K \in \{0,1\}^{n/2}$,
- a nonce, denoted by N, such that $N \in \{0,1\}^{n/2-s}$,
- associated data, denoted by A, such that $A \in \{0,1\}^*$,
- a ciphertext, denoted by C, such that $|C| \bmod n = 0$ and $|C| \geq n$, and
- a tag, denoted by tag, such that $|tag| = \ell$,

and returns either

- the message, denoted by M, or

Algorithm 4 Tag Generation Algorithm of AEAD Mode

procedure AEAD_TGEN(K, N, A, C)
  $t \leftarrow 0^n$
  if $A \neq \varepsilon$ **then**
    if $|A| \bmod n = 0$ **then**
      Parse A into n-bit blocks $(A[1], \ldots, A[a])$
    else
      Parse $A \| \mathsf{pad}(|A|)$ into n-bit blocks $(A[1], \ldots, A[a])$
    end if
    for $i = 1$ to $a - 1$ **do**
      $t \leftarrow t \oplus \text{TEM\_ENC}(K, \mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, i, 0, A[i])$
    end for
    if $|A| \bmod n = 0$ **then**
      $t \leftarrow \text{TEM\_ENC}(K, \mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, a-1, 1, A[a] \oplus t)$
    else
      $t \leftarrow \text{TEM\_ENC}(K, \mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, a-1, 2, A[a] \oplus t)$
    end if
  end if
  Parse C into n-bit blocks $(C[1], \ldots, C[m])$
  for $i = 1$ to $m - 1$ **do**
    $t \leftarrow t \oplus \text{TEM\_ENC}(K, \mathsf{flag}_{\mathsf{m}}, N, 2i, 0, C[i])$
  end for
  $t \leftarrow \text{TEM\_ENC}(K, \mathsf{flag}_{\mathsf{m}}, N, 2m-1, 1, t \oplus C[m])$
  $tag \leftarrow t_{n-1} \cdots t_{n-\ell+1} t_{n-\ell}$
  return tag
end procedure

Algorithm 5 Encryption Algorithm of AEAD Mode

procedure AEAD_ENC(K, N, A, M)
  $C \leftarrow \text{AEAD\_CGEN}(K, N, M)$
  $tag \leftarrow \text{AEAD\_TGEN}(K, N, A, C)$
  return (C, tag)
end procedure

– the reject symbol, denoted by $\perp$.

Note that AEAD_Dec does not accept inputs that do not satisfy the above conditions. For the inputs, AEAD_Dec returns $\perp$.

Fig. 1, Fig. 2, Fig. 3, and Fig. 4 illustrate the procedures of the AEAD mode with non-empty associated data and non-empty message, with non-empty associated data and empty message, with empty associated data and empty message, and with empty associated data and non-empty message, respectively.

Algorithm 6 Message Generation Algorithm of AEAD Mode

procedure AEAD_MGen(K, N, C)
  Parse C into n-bit blocks $(C[1], \ldots, C[m])$
  for $i = 1$ to m **do**
    $M[i] \leftarrow$ TEM_Dec($K, \mathsf{flag_m}, N, 2i-1, 0, C[i]$)
  end for
  $M' \leftarrow M[1] \| \cdots \| M[m]$
  if M' has the form from pad **then**
    Parse M' into M and the padding value
  else
    $M \leftarrow \perp$
  end if
  return M
end procedure

Algorithm 7 Decryption Algorithm of AEAD Mode

procedure AEAD_Dec(K, N, A, C, tag)
  $tag^* \leftarrow$ AEAD_TGen(K, N, A, C)
  if $tag = tag^*$ **then**
    $M^* \leftarrow$ AEAD_MGen(K, N, C)
  else
    $M^* \leftarrow \perp$
  end if
  return M^*
end procedure

1.1.5 MAC Mode

The MAC mode of operation consists of two algorithms, the tag generation algorithm, denoted by MAC_Gen, and the tag verification algorithm, denoted by MAC_Verif. Let ℓ be the tag size such that $0 \leq \ell \leq n$. Let $\mathsf{flag_{mac}}$ be a constant vale of s bits such that $\mathsf{flag_{mac}} \neq \mathsf{flag_{ad}}$ and $\mathsf{flag_{mac}} \neq \mathsf{flag_m}$.

MAC_Gen is specified in Algorithm 8, which accepts the following inputs,

– a secret key, denoted by K, such that $K \in \{0,1\}^{n/2}$ and
– a message, denoted by A, such that $A \in \{0,1\}^*$,

and returns the tag, denoted by tag. Note that MAC_Gen does not accept inputs that do not satisfy the above conditions. For the inputs, it returns $\perp$.

MAC_Verif is specified in Algorithm 9, which accepts the following three inputs,

– a secret key, denoted by K, such that $K \in \{0,1\}^{n/2}$,
– a message, denoted by A, such that $A \in \{0,1\}^*$ and
– a tag, denoted by tag, such that $|tag| = \ell$,

and returns either

Algorithm 8 Tag Generation Algorithm of MAC Mode

procedure MAC_Gen(K, A)
  $t \leftarrow 0^n$
  if $|A| \bmod n = 0$ and $|A| > 0$ **then**
    Parse A into n-bit blocks $(A[1], \ldots, A[a])$
  else
    Parse $A \| \mathsf{pad}(|A|)$ into n-bit blocks $(A[1], \ldots, A[a])$
  end if
  for $i = 1$ to $a - 1$ **do**
    $t \leftarrow t \oplus$ TEM_Enc$(K, \mathsf{flag}_{\mathsf{mac}}, 0^{n/2-s}, i, 0, A[i])$
  end for
  if $|A| \bmod n = 0$ and $|A| > 0$ **then**
    $t \leftarrow$ TEM_Enc$(K, \mathsf{flag}_{\mathsf{mac}}, 0^{n/2-s}, a-1, 1, A[a] \oplus t)$
  else
    $t \leftarrow$ TEM_Enc$(K, \mathsf{flag}_{\mathsf{mac}}, 0^{n/2-s}, a-1, 2, A[a] \oplus t)$
  end if
  $tag \leftarrow t_{n-1} \cdots t_{n-\ell+1} t_{n-\ell}$
  return tag
end procedure

– the accept symbol, denoted by *accept*, or
– the reject symbol, denoted by $\perp$.

Note that MAC_Verif does not accept inputs that do not satisfy the above conditions. For the inputs, it returns $\perp$.

Fig. 5 and Fig. 6 illustrate the procedures of the MAC mode with non-empty message and with empty message, respectively.

Algorithm 9 Tag Verification Algorithm of MAC Mode

procedure MAC_Verif(K, A, tag)
  $tag^* \leftarrow$ MAC_Gen(K, A, tag)
  if $tag = tag^*$ **then**
    $result \leftarrow accept$
  else
    $result \leftarrow \perp$
  end if
  return $result$
end procedure

1.2 Parameters

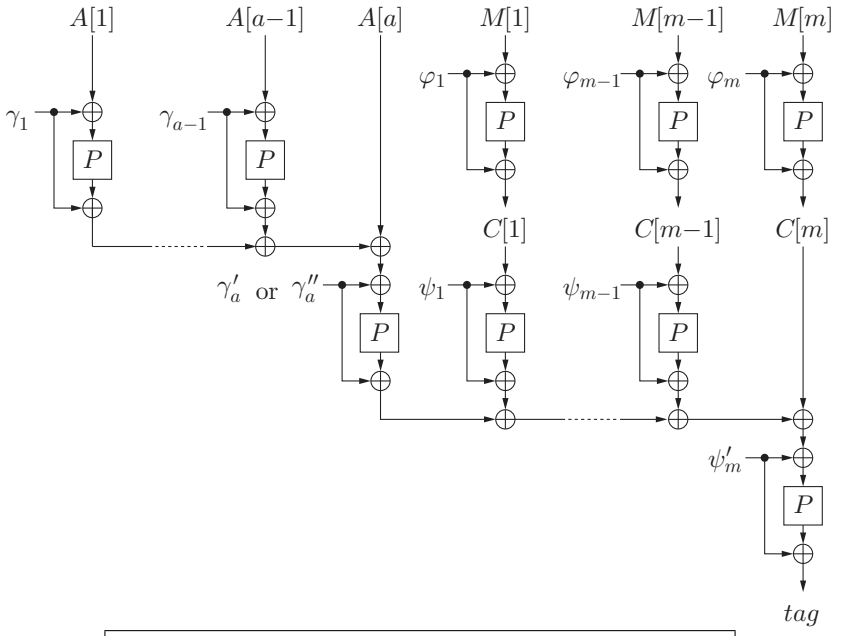
We define the parameters n and s as $n = 256$ and $s = 24$. Namely, the size of a nonce is 104 bits and $\ell = 128$. We define the constant values $\mathsf{flag}_{\mathsf{ad}}$, $\mathsf{flag}_{\mathsf{m}}$ and $\mathsf{flag}_{\mathsf{mac}}$ as $\mathsf{flag}_{\mathsf{ad}} = 0^{24}$, $\mathsf{flag}_{\mathsf{m}} = 01\|0^{22}$, and $\mathsf{flag}_{\mathsf{mac}} = 10\|0^{22}$.

We represent $\mathrm{GF}(2^{256})$ with a tower of extensions using irreducible polynomials in the following way.

– $\mathrm{GF}(2^8) = \mathrm{GF}(2)[\mathtt{x}]/(f(\mathtt{x}))$ where $f(\mathtt{x}) = \mathtt{x}^8 + \mathtt{x}^7 + \mathtt{x}^5 + \mathtt{x} + 1$.
– $\mathrm{GF}(2^{256}) = \mathrm{GF}(2^8)[\mathtt{y}]/(g(\mathtt{y}))$ where $g(\mathtt{y}) = \mathtt{y}^{32} + \mathtt{y}^3 + \mathtt{y}^2 + \mathtt{x}$.

We interchangeably think of a point a in $\mathrm{GF}(2^{256})$ as:

– a 256-bit string $a_{255} \ldots a_1 a_0$,
– a formal polynomial $a(\mathtt{x}) = (a_{255}\mathtt{x}^7 + \cdots + a_{249}\mathtt{x} + a_{248})\mathtt{y}^{31} + \cdots + (a_{15}\mathtt{x}^7 + \cdots + a_9\mathtt{x} + a_8)\mathtt{y} + (a_7\mathtt{x}^7 + \cdots + a_1\mathtt{x} + a_0)$, or
– an integer $\sum_{i=0}^{255} a_i 2^i$ between 0 to $2^{256} - 1$.



$$
\begin{aligned}
\varphi_i &= \mathrm{y}^{2i-1}L \\
\psi_i &= \mathrm{y}^{2i}L \\
\psi'_m &= \mathrm{y}^{2m-1}(\mathrm{y}+1)L \\
\gamma_i &= \mathrm{y}^{i}L' \\
\gamma'_a &= \mathrm{y}^{a-1}(\mathrm{y}+1)L' \\
\gamma''_a &= \mathrm{y}^{a-1}(\mathrm{y}+1)^2L' \ ,
\end{aligned}
$$

where

$$
\begin{aligned}
L &= (K\|\mathsf{flag_m}\|N) \oplus P(K\|\mathsf{flag_m}\|N) \\
L' &= (K\|\mathsf{flag_{ad}}\|0^{n/2-s}) \oplus P(K\|\mathsf{flag_{ad}}\|0^{n/2-s}) \ .
\end{aligned}
$$

Fig. 1. AEAD Mode with Non-empty Associated Data and Non-empty Message

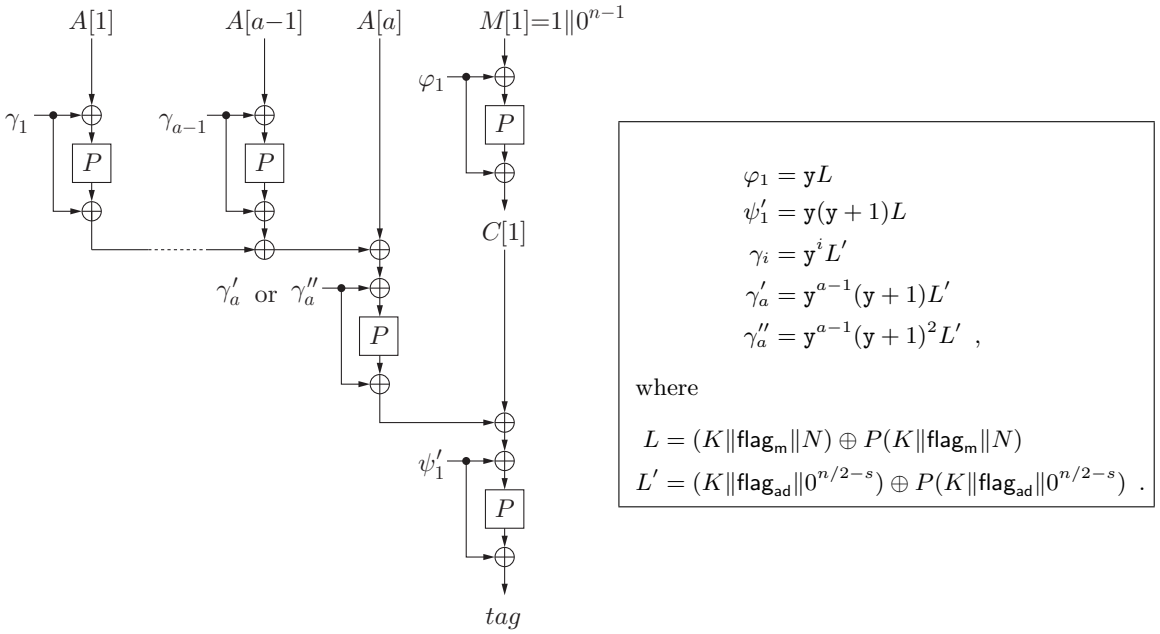


Fig. 2. AEAD Mode with Non-empty Associated Data and Empty Message

1.3 Limitations

In the AEAD mode, we restrict the sum of the sizes of an associated data and of a message at most $2^{104} - 1$ bits, and the sum of the sizes of an associated data and of a ciphertext accordingly. In the MAC mode, we restrict the size of a message at most $2^{104} - 1$ bits.

1.4 Primitive

Minalpher-P is a concrete design of permutation P which maps a 256-bit input value to a 256-bit output value. In this section, we specify the permutation Minalpher-P and the inverse Minalpher-P^{-1}.

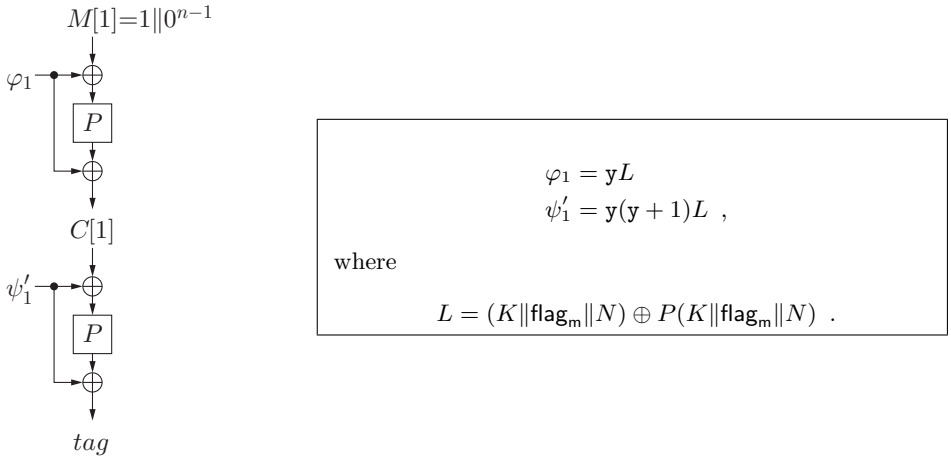
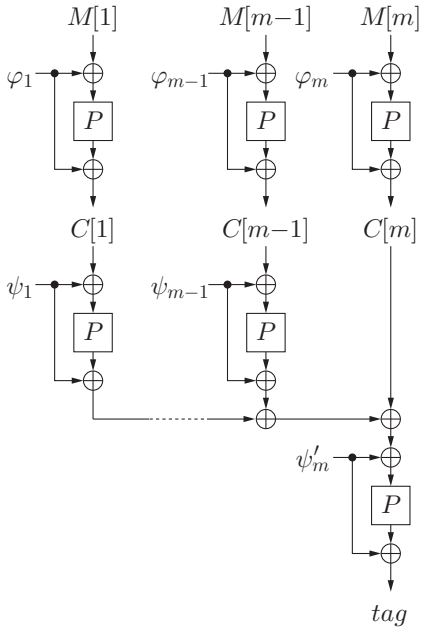


Fig. 3. AEAD Mode with Empty Associated Data and Empty Message

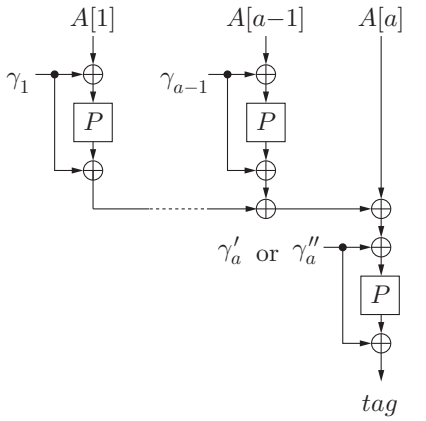


$$\varphi_i = \mathtt{y}^{2i-1}L$$
$$\psi_i = \mathtt{y}^{2i}L$$
$$\psi'_m = \mathtt{y}^{2m-1}(\mathtt{y}+1)L \ ,$$

where

$$L = (K\|\mathsf{flag_m}\|N) \oplus P(K\|\mathsf{flag_m}\|N) \ .$$

Fig. 4. AEAD Mode with Empty Associated Data and Non-empty Message



$$\gamma_i = \mathtt{y}^i L'$$
$$\gamma'_a = \mathtt{y}^{a-1}(\mathtt{y}+1)L'$$
$$\gamma''_a = \mathtt{y}^{a-1}(\mathtt{y}+1)^2 L' \ ,$$

where

$$L' = (K\|\mathsf{flag_{mac}}\|0^{n/2-s}) \oplus P(K\|\mathsf{flag_{mac}}\|0^{n/2-s}) \ .$$

Fig. 5. MAC Mode with Non-empty Message

1.4.1 Structure of Minalpher-P

1.4.1.1 Specification of Minalpher-P (Forward Procedure)

Minalpher-P has a round function R which consists of an S-function, a T-function, an M-function and an E-function. Figure 7 shows the structure of the round function R for the forward procedure.

In the forward procedure, first the 256-bit input value IN is copied to a 256-bit value X_0 which is an input to the round function. Let the number of rounds be $r + 0.5$. Then, the following operations are performed from $i = 1$ to r,

$$X_i \leftarrow R(X_{i-1}, E(i-1)),$$

where i denotes the round number and $R(X, E(i))$ is calculated as follows:

$$R(X, E(i)) \leftarrow M \circ T \circ S(X) \oplus E(i).$$

Finally, X_{r+1} is calculated from X_r as follows,

$$X_{r+1} \leftarrow T \circ S(X_r),$$

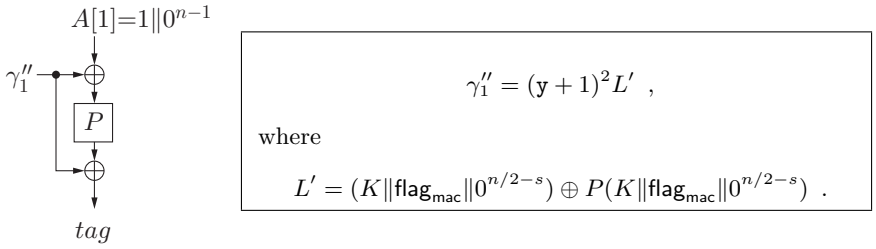


Fig. 6. MAC Mode with Empty Message

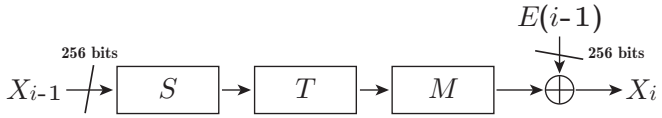


Fig. 7. Structure of Round Functions

and X_{r+1} is copied to the 256-bit output value OUT. We call composition $T \circ S$ 0.5 round. The recommended parameter in Minalpher-P is $r = 17$, then Minalpher-P has 17.5 rounds.

1.4.1.2 Specification of Minalpher-P^{-1} (Backward Procedure)

In the backward procedure, first the 256-bit input value IN is copied to a 256-bit value X_0 which is an input to the round function. Then, following operations are performed from $i = 1$ to r,

$$X_i \leftarrow R(X_{i-1}, M \circ E(r - i)),$$

where the function R is the same function as that of the forward procedure. Finally, X_{r+1} is calculated from X_r as follows,

$$X_{r+1} \leftarrow T \circ S(X_r),$$

and X_{r+1} is copied to the 256-bit output value OUT. Minalpher-P uses $r = 17$, then Minalpher-P has 17.5 rounds.

1.4.2 State and Mapping

The operation of round function of Minalpher-P is performed on 2 two-dimensional matrices $A \in \left\{\{0,1\}^4\right\}^{4\times 8}$ and $B \in \left\{\{0,1\}^4\right\}^{4\times 8}$. Each matrix has 4×8 elements and the size of each element is 4 bits. Hereafter, a 4-bit value is called a nibble. The 256-bit input value to Minalpher-P or Minalpher-P^{-1} is mapped to two matrices A and B. Let $X \in \{0,1\}^{256}$ be the 256-bit input value. In this mapping, X is split into 64 nibbles as $X = X[0]\|X[1]\|\cdots\|X[63]$. Each nibble $X[i]$ is copied into each element of matrix $A \in \left\{\{0,1\}^4\right\}^{4\times 8}$ and matrix $B \in \left\{\{0,1\}^4\right\}^{4\times 8}$ as illustrated in Fig. 8. After calculation of the round function, the result in matrices A and B is mapped to the 256-bit output value OUT as illustrated in Fig. 8. In this document, $A[i][j]$ and $B[i][j]$ denote the nibble in row i and column j of A and B, respectively.

1.4.3 Round Function

The round function consists of the S-function, the T-function and the M-function. Moreover the S-function consists of SubNibbles (SN), and the T-function consists of ShuffleRows (SR) and SwapMatrices (SM), and the M-function consists of XorMatrix (XM) and MixColumns (MC), where SN, SR and MC are functions from $\left\{\{0,1\}^4\right\}^{4\times 8}$ to $\left\{\{0,1\}^4\right\}^{4\times 8}$. In the end of the round function, the state is XORed with the round constant which is calculated from the round number i and the E-function.

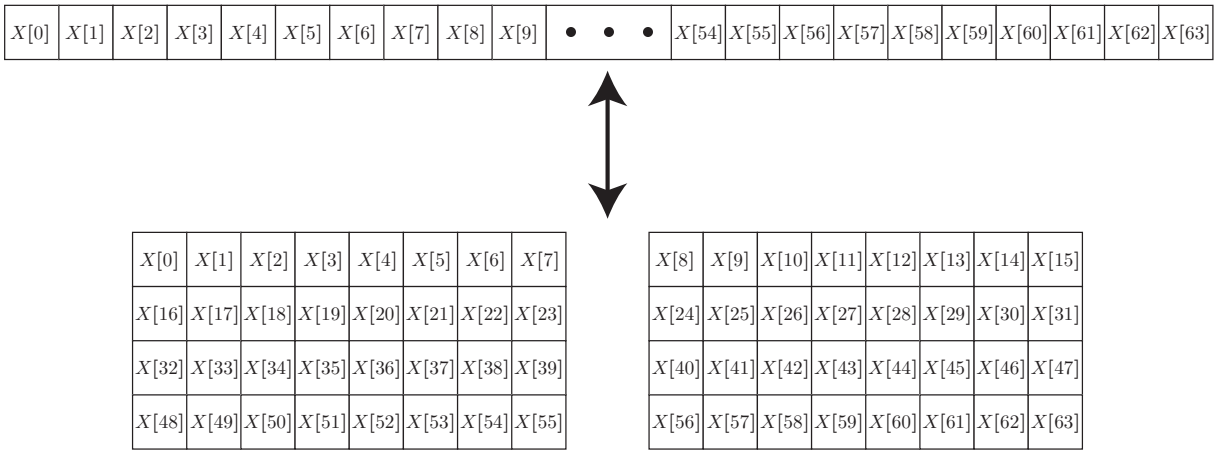


Fig. 8. Mapping between the State and the Input

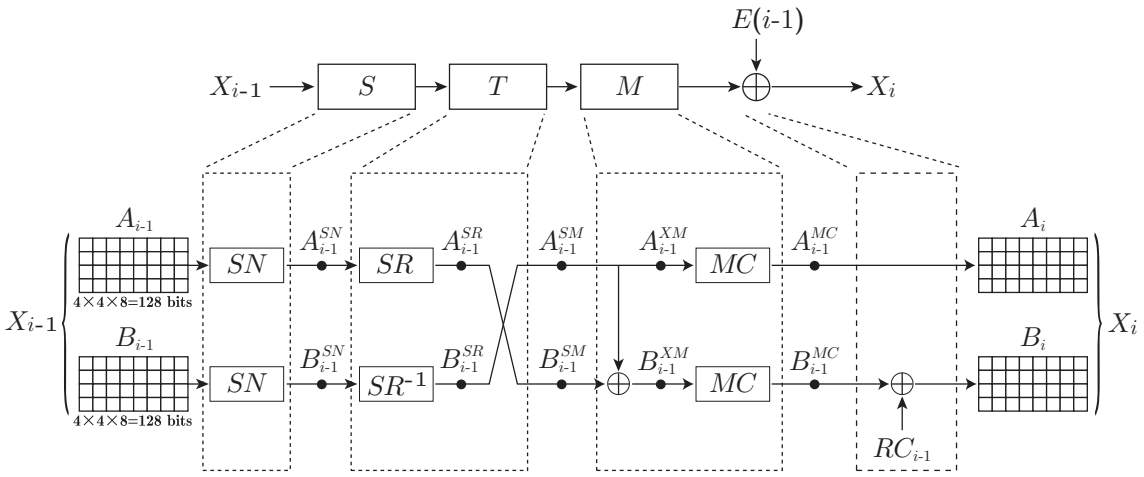


Fig. 9. Round Function

Let X_{i-1} and X_i be input and output of the round function for round i. X_{i-1} consists of two 4×8 nibble-wise matrices A_{i-1} and B_{i-1}. Similarly X_i consists of A_i and B_i. A_i and B_i are updated from A_{i-1} and B_{i-1} as follows:

$$
\begin{aligned}
A_{i-1}^{SN} &\leftarrow SN(A_{i-1}), & B_{i-1}^{SN} &\leftarrow SN(B_{i-1}),\\
A_{i-1}^{SR} &\leftarrow SR(A_{i-1}^{SN}), & B_{i-1}^{SR} &\leftarrow SR^{-1}(B_{i-1}^{SN}),\\
A_{i-1}^{SM} &\leftarrow B_{i-1}^{SR}, & B_{i-1}^{SM} &\leftarrow A_{i-1}^{SR},\\
A_{i-1}^{XM} &\leftarrow A_{i-1}^{SM}, & B_{i-1}^{XM} &\leftarrow A_{i-1}^{SM} \oplus B_{i-1}^{SM},\\
A_{i-1}^{MC} &\leftarrow MC(A_{i-1}^{XM}), & B_{i-1}^{MC} &\leftarrow MC(B_{i-1}^{XM}),\\
A_i &\leftarrow A_{i-1}^{MC}, & B_i &\leftarrow B_{i-1}^{MC} \oplus RC_{i-1}.
\end{aligned}
$$

In this document, A^{op} and B^{op} denote $\left\{\{0,1\}^4\right\}^{4\times 8}$ data after operation op. Fig. 9 shows the structure of the round function.

1.4.3.1 SubNibbles (SN)

SN substitutes each nibble in the state into another value by using 4-bit S-box s, where s is the permutation from $\{\texttt{0x0}, \ldots, \texttt{0xF}\}$ to $\{\texttt{0x0}, \ldots, \texttt{0xF}\}$ defined as follows:

x	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
$s(x)$	0xB	0x3	0x4	0x1	0x2	0x8	0xC	0xF	0x5	0xD	0xE	0x0	0x6	0x9	0xA	0x7

The S-box of Minalpher-P is an involution S-box. For the element in row i and column j of A and B, SN performs the following transformation:

$$A^{SN}[i][j] \leftarrow s(A[i][j]), \quad 0 \leq i < 4, \ 0 \leq j < 8,$$
$$B^{SN}[i][j] \leftarrow s(B[i][j]), \quad 0 \leq i < 4, \ 0 \leq j < 8.$$

1.4.3.2 ShuffleRows (SR)

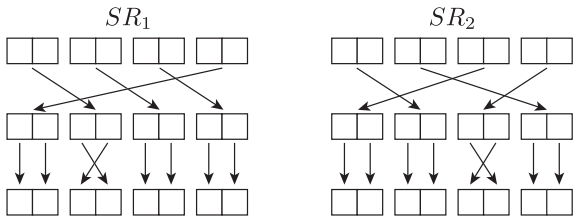


Fig. 10. ShuffleRows

SR shuffles nibble positions within each row. SR consists of 2 different shuffle functions SR_1 and SR_2, where SR_1 and SR_2 are permutations from $\{0, \ldots, 7\}$ to $\{0, \ldots, 7\}$ defined as follows:

i	0	1	2	3	4	5	6	7
$SR_1(i)$	6	7	1	0	2	3	4	5
$SR_2(i)$	4	5	0	1	7	6	2	3
$SR_1^{-1}(i)$	3	2	4	5	6	7	0	1
$SR_2^{-1}(i)$	2	3	6	7	0	1	5	4

Fig. 10 shows structures of SR_1 and SR_2. For elements in column j of A, SR performs the following transformation:

$$A^{SR}[0][j] \leftarrow A[0][SR_1(j)], \quad 0 \leq j < 8,$$
$$A^{SR}[1][j] \leftarrow A[1][SR_2(j)], \quad 0 \leq j < 8,$$
$$A^{SR}[2][j] \leftarrow A[2][SR_1^{-1}(j)], \quad 0 \leq j < 8,$$
$$A^{SR}[3][j] \leftarrow A[3][SR_2^{-1}(j)], \quad 0 \leq j < 8.$$

For elements in column j of B, SR^{-1} performs the following transformation:

$$B^{SR}[0][j] \leftarrow B[0][SR_1^{-1}(j)], \quad 0 \leq j < 8,$$
$$B^{SR}[1][j] \leftarrow B[1][SR_2^{-1}(j)], \quad 0 \leq j < 8,$$
$$B^{SR}[2][j] \leftarrow B[2][SR_1(j)], \quad 0 \leq j < 8,$$
$$B^{SR}[3][j] \leftarrow B[3][SR_2(j)], \quad 0 \leq j < 8.$$

1.4.3.3 SwapMatrices (SM)

SM swaps the matrix A_i for the matrix B_i, namely , SM performs the following transformation:

$$A^{SM} \leftarrow B,$$
$$B^{SM} \leftarrow A.$$

1.4.3.4 XorMatrix (XM)

XM is a linear function, and the matrix B is XORed with the matrix A, namely, XM performs the following transformation:

$$A^{XM} \leftarrow A,$$
$$B^{XM} \leftarrow A \oplus B.$$

1.4.3.5 MixColumns (MC)

MC is a linear function within each column. MC is expressed as a multiplication by the following matrix:

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

For elements in column j of A and B, MC performs the following transformation:

$$A^{MC}[0][j] \leftarrow A[0][j] \oplus A[1][j] \oplus A[3][j], \quad 0 \le j < 8,$$
$$A^{MC}[1][j] \leftarrow A[1][j] \oplus A[2][j] \oplus A[0][j], \quad 0 \le j < 8,$$
$$A^{MC}[2][j] \leftarrow A[2][j] \oplus A[3][j] \oplus A[1][j], \quad 0 \le j < 8,$$
$$A^{MC}[3][j] \leftarrow A[3][j] \oplus A[0][j] \oplus A[2][j], \quad 0 \le j < 8,$$

and

$$B^{MC}[0][j] \leftarrow B[0][j] \oplus B[1][j] \oplus B[3][j], \quad 0 \le j < 8,$$
$$B^{MC}[1][j] \leftarrow B[1][j] \oplus B[2][j] \oplus B[0][j], \quad 0 \le j < 8,$$
$$B^{MC}[2][j] \leftarrow B[2][j] \oplus B[3][j] \oplus B[1][j], \quad 0 \le j < 8,$$
$$B^{MC}[3][j] \leftarrow B[3][j] \oplus B[0][j] \oplus B[2][j], \quad 0 \le j < 8.$$

1.4.3.6 Round Constant

The round constant RC_{i-1} is calculated from the round number i and it is a matrix on $\left\{\{0,1\}^4\right\}^{4\times 8}$. $r \in [0, 15]$ is calculated from i as $r = i - 1 \bmod 16$. RC_{i-1} is generated as follows:

$r \oplus 0$	$r \oplus 1$	$r \oplus 2$	$r \oplus 3$	0	0	0	0
$r \oplus 1$	$r \oplus 0$	$r \oplus 3$	$r \oplus 2$	0	0	0	0
$r \oplus 2$	$r \oplus 3$	$r \oplus 0$	$r \oplus 1$	0	0	0	0
$r \oplus 3$	$r \oplus 2$	$r \oplus 1$	$r \oplus 0$	0	0	0	0

2 Security Goals

Before starting the discussion of security goals, we state that users are required to use the public message number as a nonce and the secret message number is absent.

Minalpher is designed to provide 128-bit security for confidentiality and integrity. In addition to the basic security, it is designed to provide some level of security against two kinds of misuses: nonce reuse and unverified plaintext release. We explain these security notions in the following subsection.

2.1 Security Definition

We specify security definitions for privacy and for authenticity of an AEAD scheme $\Pi = (\mathsf{E}, \mathsf{D}, \mathsf{M}, \mathsf{V})$ such that

- $\mathsf{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{C} \times \mathcal{TG}$ is the encryption algorithm of the AEAD mode,
- $\mathsf{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \times \mathcal{TG} \to \mathcal{RE}$ is the decryption algorithm of the AEAD mode,
- $\mathsf{M} : \mathcal{K} \times \mathcal{M}_{\mathsf{mac}} \to \mathcal{TG}_{\mathsf{mac}}$ is the tag generation algorithm of the MAC mode,
- $\mathsf{V} : \mathcal{K} \times \mathcal{M}_{\mathsf{mac}} \times \mathcal{TG}_{\mathsf{mac}} \to \mathcal{RE}_{\mathsf{mac}}$ is the tag verification algorithm of the MAC mode,

where $\mathcal{K}$ is a key set, $\mathcal{N}$ is a nonce set, $\mathcal{AD}$ is an associated data set, $\mathcal{M}$ is a message set of the AEAD mode, $\mathcal{C}$ is a ciphertext set, $\mathcal{TG}$ is a tag set of the AEAD mode, $\mathcal{M}_{\mathsf{mac}}$ is a message set of the MAC mode, $\mathcal{TG}_{\mathsf{mac}}$ is a tag set of the MAC mode, $\mathcal{RE}$ is a result set of the decryption algorithm, and $\mathcal{RE}_{\mathsf{mac}}$ is a result set of the tag verification algorithm.

Hereafter, $\mathcal{AD} = \{0,1\}^*$, $\mathcal{M} = \{0,1\}^*$, $\mathcal{M}_{\mathsf{mac}} = \{0,1\}^*$, $\mathcal{C} = (\{0,1\}^n)^{\geq 1}$, $\mathcal{TG} = \{0,1\}^\ell$, $\mathcal{TG}_{\mathsf{mac}} = \{0,1\}^\ell$, and $\mathcal{RE}_{\mathsf{mac}} = \{accept, \perp\}$, where $(\{0,1\}^n)^{\geq 1}$ is a set of bit strings such that the lengths are multiples of n and an empty string is not included. $\mathcal{RE}$ is specified in each security definition.

We denote by $K \xleftarrow{R} \mathcal{K}$ choosing a key K uniformly at random from $\mathcal{K}$.

2.1.1 Privacy

The goal of privacy is that for a key K, E_K behaves like "Block-wise Random Function", denoted by \$, and M_K behaves like a random function, denoted by $\mathcal{R}$, where K is chosen uniformly at random from $\{0,1\}^{n/2}$.

Roughly speaking, for an input, \$ defines its output uniformly at random on block-wise, that is, an output block depends on a nonce, a block number, and a message block but not on other message blocks. Concretely, \$ is defined as follows.

Definition 2 (Block-wise Random Function). *Let $\mathcal{R}_1 : \mathcal{N} \times \mathbb{Z} \times \{0,1\}^n \to \{0,1\}^n$ and $\mathcal{R}_2 : \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{TG}$ be random functions. For query $(N, A, M) \in \mathcal{N} \times \mathcal{AD} \times \mathcal{M}$, a block-wise random function defines the response as follows.*

1. *Parse $M \| \mathsf{pad}(|M|)$ into n-bit blocks $(M[1], \ldots, M[m])$,*
2. *For each block number i, $C[i] \leftarrow \mathcal{R}_1(N, i, M[m])$,*
3. *$tag \leftarrow \mathcal{R}_2(N, A, M)$,*
4. *$(C[1] \| \cdots \| C[m], tag)$ is the response.*

We define "behave like" as the indistinguishability from $(\$, \mathcal{R})$. The advantage is defined as follows.

Definition 3 (Privacy). *Let \$ and $\mathcal{R}$ be a block-wise random function and a random function, respectively. The priv-advantage of an adversary $\mathcal{A}$ is defined as*

$$\mathrm{Adv}^{\mathrm{priv}}_{\Pi}(\mathcal{A}) = \Pr[K \xleftarrow{R} \mathcal{K} : \mathcal{A}^{\mathsf{E}_K, \mathsf{M}_K} \Rightarrow 1] - \Pr[\mathcal{A}^{\$, \mathcal{R}} \Rightarrow 1] \ .$$

We consider two types of adversary $\mathcal{A}$. One is a nonce-respecting adversary, which is not permitted to make queries with reused nonces. The other is a nonce-reusing adversary, which is permitted to make the queries.

When $\mathcal{A}$ is a nonce-respecting adversary, this definition is the indistinguishability from a pair of random functions. For confidentiality, we say Π has k-bit priv-security if for a key $K \xleftarrow{R} \mathcal{K}$ and any nonce-respecting adversary, the complexity to distinguish $(\mathsf{E}_K, \mathsf{M}_K)$ from $(\$, \mathcal{R})$ is at least 2^k.

When $\mathcal{A}$ is a nonce-reusing adversary, this definition is the indistinguishability from a pair of a block-wise random function and a random function. Since $(\$, \mathcal{R})$ leaks no information of inputs except for collisions of message blocks with the same block number, so is $(\mathsf{E}_K, \mathsf{M}_K)$ which is indistinguishable from $(\$, \mathcal{R})$. For confidentiality in the nonce reuse setting, we say Π has k-bit block-wise priv-security if for a key $K \xleftarrow{R} \mathcal{K}$ and any nonce-reusing adversary, the complexity to distinguish this scheme from $(\$, \mathcal{R})$ is at least 2^k.

We use the following notations interchangeably: k-bit priv-security and 2^k priv-security. This rule is also applied to other security notions.

2.1.2 Authenticity

The goal of authenticity is that for a key $K \xleftarrow{R} \mathcal{K}$, no adversary makes a query to D_K or V_K such that *accept* is returned even with nonce reuse and unverified plaintext release. Unverified plaintext release (UPR) means that decrypted messages are returned even if the corresponding tags are invalid, that is, in this setting, $\mathcal{RE} = \mathcal{M} \times \{accept, \perp\}$. If the UPR setting is not considered, $\mathcal{RE} = \mathcal{M} \cup \{\perp\}$. The advantage is defined as follows.

Definition 4 (Authenticity). *The auth-advantage of an adversary $\mathcal{A}$ is defined as*

$$\mathrm{Adv}^{\mathrm{auth}}_{\Pi}(\mathcal{A}) = \Pr[K \xleftarrow{R} \mathcal{K} : \mathcal{A}^{\mathsf{E}_K,\mathsf{D}_K,\mathsf{M}_K,\mathsf{V}_K} \textit{ forges}],$$

where $\mathcal{RE} = \mathcal{M} \times \{accept, \perp\}$ if in the UPR setting, and $\mathcal{RE} = \mathcal{M} \cup \{\perp\}$ otherwise. The goal of $\mathcal{A}$ is to make a query to D_K or V_K such that accept is returned. Note that $\mathcal{A}$ cannot make trivial queries listed bellow.

- *A query (N, A, C, tag) to D_K which was obtained from E_K.*
- *A query (A, tag) to V_K which was obtained from M_K.*

We consider two types of adversary $\mathcal{A}$. One is a nonce-respecting adversary which is not permitted to make queries with reused nonces to E_K and the other is a nonce-reusing adversary which is permitted to make the queries.

For integrity, we say Π has k-bit auth-security if for a key $K \xleftarrow{R} \mathcal{K}$ and any nonce-respecting adversary, the complexity to forge a tag is at least 2^k without UPR.

For integrity in the nonce reuse and UPR setting, we say Π has k-bit auth-security in this setting if for a key $K \xleftarrow{R} \mathcal{K}$ and any nonce-reusing adversary, the complexity to forge a tag is at least 2^k with UPR.

2.1.3 (Strong) Tweakable Pseudorandom Permutation

Let $\mathcal{T}$ be a tweak set. We give definitions of Tweakable PseudoRandom Permutation (TPRP) and of Strong Tweakable PseudoRandom Permutation (STPRP) for tweakable block-cipher $\mathsf{TBC} = (\mathsf{TBC\_E}, \mathsf{TBC\_D})$, where $\mathsf{TBC\_E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is the encryption function and $\mathsf{TBC\_D} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is the decryption function.

Let $\mathsf{TRP} = (\mathsf{TRP\_F}, \mathsf{TRP\_B})$ be a tweakable random permutation which is chosen uniformly at random from all tweakable permutations of n bits with the tweak set $\mathcal{T}$, where $\mathsf{TRP\_F} : \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is the forward oracle and $\mathsf{TRP\_B} : \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is its backward oracle.

Definition 5 (TPRP). *The* tprp*-advantage of a distinguisher $\mathcal{D}$ for TBC is defined as*

$$\mathrm{Adv}^{\mathrm{tprp}}_{\mathsf{TBC}}(\mathcal{D}) = \Pr[K \xleftarrow{R} \mathcal{K} : \mathcal{D}^{\mathsf{TBC\_E}_K} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{TRP\_F}} \Rightarrow 1] \ .$$

Definition 6 (STPRP). *The* stprp*-advantage of a distinguisher $\mathcal{D}$ for TBC is defined as*

$$\mathrm{Adv}^{\mathrm{stprp}}_{\mathsf{TBC}}(\mathcal{D}) = \Pr[K \xleftarrow{R} \mathcal{K} : \mathcal{D}^{\mathsf{TBC\_E}_K,\mathsf{TBC\_D}_K} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{TRP\_F},\mathsf{TRP\_B}} \Rightarrow 1] \ .$$

2.2 Basic Security Claims

We make the following claims on the security of Minalpher without nonce reuse and UPR.

Claim (Confidentiality). Minalpher has 128-bit priv-security.

Claim (Integrity). Minalpher has 128-bit auth-security.

These claims are summarized in Table 1.

2.3 Additional Security Claims

In addition to the basic security claims, we claim that Minalpher has the block-wise priv-security for confidentiality in the nonce reuse setting and the auth-security for integrity in the nonce reuse and UPR setting.

Claim (Confidentiality in Nonce Reuse Setting). Minalpher has 128-bit block-wise priv-security in the nonce reuse setting.

Claim (Integrity in Nonce Reuse and UPR Setting). Minalpher has 128-bit auth-security in the nonce reuse and UPR setting.

Table 1. The Intended Numbers of Bits of Security of Minalpher for the Recommended Parameter $n = 256$ and $\ell = 128$

category	the number of bits of security
confidentiality for the plaintext	128
confidentiality for the secret message number	N/A
integrity for the plaintext	128
integrity for the associated data	128
integrity for the secret message number	N/A
integrity for the public message number	128

2.4 Security Claims of the Mode of Operation and of the Tweakable Even-Mansour

The above claims are based on security claims of the Minalpher mode of operation and of the tweakable Even-Mansour using Minalpher-P.

We claim the security of the mode of operation as follows.

Claim (Confidentiality). If tweakable Even-Mansour has k-bit tprp-security, then the Minalpher mode has $\min\{n/2, k\}$-bit priv-security.

Claim (Integrity). If tweakable Even-Mansour has k-bit stprp-security, then the Minalpher mode has $\min\{n/2, k, \ell\}$-bit auth-security.

Claim (Confidentiality in Nonce Reuse Setting). If tweakable Even-Mansour has k-bit tprp-security, then the Minalpher mode has $\min\{n/2, k\}$-bit block-wise priv-security in the nonce reuse setting.

Claim (Integrity in Nonce Reuse and UPR Setting). If tweakable Even-Mansour has k-bit stprp-security, then the Minalpher mode has $\min\{n/2, k, \ell\}$-bit auth-security in the nonce reuse and UPR setting.

We also claim the security of tweakable Even-Mansour with Minalpher-P. Let $\mathbb{Z}_d$ be the set of integers from 0 to $d-1$ inclusive.

Claim. Tweakable Even-Mansour with Minalpher-P has 128-bit stprp-security, where the tweak set $\mathcal{T}$ is defined as

$$\mathcal{T} = \{\mathsf{flag}_{\mathsf{ad}}, \mathsf{flag}_{\mathsf{m}}, \mathsf{flag}_{\mathsf{mac}}\} \times \{0,1\}^{104} \times \mathbb{Z}_{2^{104}} \times \mathbb{Z}_3.$$

The combination of these claims offers the claims given in Subsections 2.2 and 2.3.

3 Security Analysis

The claims in Section 2 are based on security analyses of the Minalpher mode of operation and of tweakable Even-Mansour using Minalpher-P.

Section 3.1.2 shows the security proof of the Minalpher mode of operation: if tweakable Even-Mansour with an n-bit permutation has $\mathcal{O}(2^{n/2})$ (s)tprp-security, then the mode of operation has $\mathcal{O}(2^{n/2})$ (block-wise) priv-security and $\mathcal{O}(2^{\min\{n/2,\ell\}})$ auth-security.

To claim the security of tweakable Even-Mansour with Minalpher-P, Section 3.1.3 shows that tweakable Even-Mansour with an n-bit permutation has $\mathcal{O}(2^{n/2})$ security in the ideal permutation model. This proof ensures that there are no generic attacks on tweakable Even-Mansour.

Section 3.2 and Section 3.3 show analyses of Minalpher-P. Section 3.2 shows the proof of the minimum number of active S-boxes for the primitive which is derived theoretically or experimentally. Section 3.3 shows the initial cryptanalysis for Minalpher.

3.1 Security Proofs

For the sake of completeness we provide self-contained proofs for essentially all lemmas and theorems. Our proofs are based on the code-based game-playing technique [2].

3.1.1 Preliminaries

Pseudorandom Function. Let $\mathcal{G} : \mathcal{K} \times \boldsymbol{D} \to \boldsymbol{R}$ be a keyed function, where $\mathcal{K}$ is a key set.

Definition 7 (PRF). *The prf-advantage of a distinguisher $\mathcal{D}$ for $\mathcal{G}$ is defined as*

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathcal{G}}(\mathcal{D}) = \Pr\left[K \xleftarrow{R} \mathcal{K} : \mathcal{D}^{\mathcal{G}_K} \Rightarrow 1\right] - \Pr\left[\mathcal{D}^{\mathcal{R}} \Rightarrow 1\right] \ ,$$

where $\mathcal{R}$ is a function chosen uniformly at random from the set of functions with the same domain and range as $\mathcal{G}_K$.

3.1.2 Security of Mode of Operation

Let TEM = (TEM_Enc, TEM_Dec) be tweakable Even-Mansour defined in Algorithm 1 and Algorithm 2. In this subsection, we prove the followings.

- If TEM has $\mathcal{O}(2^k)$ tprp-security, then the Minalpher mode of operation has $\mathcal{O}(2^{\min\{n/2,k\}})$ priv-security (Theorem 1).
- If TEM has $\mathcal{O}(2^k)$ stprp-security, then the Minalpher mode of operation has $\mathcal{O}(2^{\min\{n/2,k,\ell\}})$ auth-security (Theorem 2).
- If TEM has $\mathcal{O}(2^k)$ tprp-security, then the Minalpher mode of operation has $\mathcal{O}(2^{\min\{n/2,k\}})$ block-wise priv-security in the nonce reuse setting (Theorem 3).
- If TEM has $\mathcal{O}(2^k)$ stprp-security, then the Minalpher mode of operation has $\mathcal{O}(2^{\min\{n/2,k,\ell\}})$ auth-security in the nonce reuse and UPR setting (Theorem 4).

Theorem 1 (Confidentiality). *Let $\mathcal{A}$ be a nonce-respecting adversary. Suppose that the number of queries made by $\mathcal{A}$ is at most q and the number of invocations of* TEM *induced by the queries of $\mathcal{A}$ is at most σ. Then, for any $\mathcal{A}$, there exists a distinguisher $\mathcal{D}$ such that*

$$\mathrm{Adv}^{\mathrm{priv}}_{\mathrm{Minalpher}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{tprp}}_{\mathrm{TEM}}(\mathcal{D}) + \frac{\sigma^2}{2^{n+1}} + \frac{q^2}{2^n} \ ,$$

where $\mathcal{D}$ makes at most σ queries and the running time of $\mathcal{D}$ is sum of the running time of $\mathcal{A}$ and the time required to execute the Minalpher mode using an oracle for tweakable Even-Mansour to answer to the queries made by $\mathcal{A}$.

Theorem 2 (Integrity). *Let $\mathcal{A}$ be a nonce-respecting adversary. Suppose that the number of queries made by $\mathcal{A}$ is at most q and the number of invocations of* TEM *induced by the queries of $\mathcal{A}$ is at most σ. Then, for any $\mathcal{A}$, there exists a distinguisher $\mathcal{D}$ such that*

$$\mathrm{Adv}^{\mathrm{auth}}_{\mathrm{Minalpher}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{stprp}}_{\mathrm{TEM}}(\mathcal{D}) + \frac{q}{2^\ell} + \frac{\sigma^2}{2^{n+1}} + \frac{q^2}{2^n} \ ,$$

where $\mathcal{D}$ makes at most σ queries and the running time of $\mathcal{D}$ is sum of the running time of $\mathcal{A}$ and the time required to execute the Minalpher mode using an oracle for tweakable Even-Mansour to answer to the queries made by $\mathcal{A}$.

Theorem 3 (Confidentiality in Nonce Reuse Setting). *Let $\mathcal{A}$ be a nonce-reusing adversary. Suppose that the number of queries made by $\mathcal{A}$ is at most q and the number of invocations of* TEM *induced by the queries of $\mathcal{A}$ is at most σ. Then, for any $\mathcal{A}$, there exists a distinguisher $\mathcal{D}$ such that*

$$\mathrm{Adv}^{\mathrm{priv}}_{\mathrm{Minalpher}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{tprp}}_{\mathrm{TEM}}(\mathcal{D}) + \frac{\sigma^2 + q^2}{2^n} \ ,$$

where $\mathcal{D}$ makes at most σ queries and the running time of $\mathcal{D}$ is sum of the running time of $\mathcal{A}$ and the time required to execute the Minalpher mode using an oracle for tweakable Even-Mansour to answer to the queries made by $\mathcal{A}$.

Theorem 4 (Integrity in Nonce Reuse and UPR Setting). *Let $\mathcal{A}$ be a nonce-reusing adversary. Suppose that the number of queries made by $\mathcal{A}$ is at most q and the number of invocations of* TEM *induced by the queries of $\mathcal{A}$ is at most σ. Then, for any $\mathcal{A}$, there exists a distinguisher $\mathcal{D}$ such that*

$$\mathrm{Adv}^{\mathrm{auth}}_{\mathrm{Minalpher}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{stprp}}_{\mathrm{TEM}}(\mathcal{D}) + \frac{q}{2^\ell} + \frac{\sigma^2 + q^2}{2^n} \ ,$$

where $\mathcal{D}$ makes at most σ queries and the running time of $\mathcal{D}$ is sum of the running time of $\mathcal{A}$ and the time required to execute the Minalpher mode using an oracle for tweakable Even-Mansour to answer to the queries made by $\mathcal{A}$.

3.1.2.1 Road of Proofs

Let $\mathsf{TGen}_{\mathrm{AE}}$ be a function chosen uniformly at random from functions with the same domain and range as AEAD_TGEN$(K, \cdot, \cdot, \cdot)$. Let $\mathsf{TGen}_{\mathrm{MAC}}$ be a function chosen uniformly at random from functions with the same domain and range as MAC_GEN$(K, \cdot)$. Let $\mathsf{TRP} = (\mathsf{TRP\_F}, \mathsf{TRP\_B})$ be a tweakable permutation chosen uniformly at random from tweakable permutations with the same domain and range as TEM.

We prove Theorems 1, 2, 3, and 4 by the following strategy.

Step 1. We prove the security of Minalpher*, which is the mode obtained from Minalpher by replacing
- TEM_ENC in AEAD_CGEN with $\mathsf{TRP\_F}$,
- AEAD_TGEN with $\mathsf{TGen}_{\mathrm{AE}}$, and
- MAC_GEN with $\mathsf{TGen}_{\mathrm{MAC}}$.

Note that the corresponding changes should also be applied to AEAD_DEC and MAC_VERIF.

Step 2. We prove that
- AEAD_TGEN using TRP is a PRF (indistinguishable from $\mathsf{TGen}_{\mathrm{AE}}$), and
- MAC_GEN using TRP is a PRF (indistinguishable from $\mathsf{TGen}_{\mathrm{MAC}}$).

Suppose that TEM, AEAD_TGEN, and MAC_GEN are instantiated with TRP. Then, they are independent since the set of tweaks for TEM, the set of tweaks for AEAD_TGEN and the set of tweaks for MAC_GEN are disjoint. Thus, by combining the results of Step 1 and Step 2, the security of the mode of operation is obtained under the assumption that TEM is (S)TPRP (Theorems 1, 2, 3, and 4).

3.1.2.2 Step 1: Security of Minalpher*

Lemma 1 (Confidentiality). *For any nonce-respecting adversary $\mathcal{A}$,*

$$\mathrm{Adv}^{\mathrm{priv}}_{\mathrm{Minalpher}^*}(\mathcal{A}) = 0 \ .$$

Proof. Without nonce reuse, a new invocation of TRP gives a new tweak to TRP. □

Lemma 2 (Integrity). *For any nonce-respecting adversary $\mathcal{A}$ making at most q queries to $\mathsf{TGen}_{\mathrm{AE}}$ and $\mathsf{TGen}_{\mathrm{MAC}}$ in total,*

$$\mathrm{Adv}^{\mathrm{auth}}_{\mathrm{Minalpher}^*}(\mathcal{A}) \leq \frac{q}{2^{\ell}} \ .$$

Proof. $\mathcal{A}$ has to predict an output of $\mathsf{TGen}_{\mathrm{AE}}$ or $\mathsf{TGen}_{\mathrm{MAC}}$ to a new input to succeed in forgery. □

Lemma 3 (Confidentiality in Nonce Reuse Setting). *For any nonce-reusing adversary $\mathcal{A}$,*

$$\mathrm{Adv}^{\mathrm{priv}}_{\mathrm{Minalpher}^*}(\mathcal{A}) \leq \frac{\sigma^2}{2^{n+1}} \ ,$$

where σ is the number of invocations of TRP induced by the queries of $\mathcal{A}$.

Proof. With nonce reuse, message blocks at the same position may be encrypted with the same tweak. Message blocks at different positions are encrypted with different tweaks. Since TRP is a random permutation on $\{0,1\}^n$ for each tweak, from PRP/PRF switching lemma [2], this lemma holds. □

Lemma 4 (Integrity in Nonce Reuse and UPR Setting). *For any nonce-reusing adversary $\mathcal{A}$ making at most q queries to $\mathsf{TGen}_{\mathrm{AE}}$ and $\mathsf{TGen}_{\mathrm{MAC}}$ in total,*

$$\mathrm{Adv}^{\mathrm{auth}}_{\mathrm{Minalpher}^*}(\mathcal{A}) \leq \frac{q}{2^{\ell}}$$

in the UPR setting.

Proof. Even with nonce reuse and/or UPR, $\mathcal{A}$ has to predict an output of $\mathsf{TGen}_{\mathrm{AE}}$ or $\mathsf{TGen}_{\mathrm{MAC}}$ to a new input to succeed in forgery. □

3.1.2.3 Step 2: Security of AEAD_TGen and of MAC_Gen

Let $\mathcal{G}_{\mathrm{AE}}$ be the tag generation function AEAD_TGen instantiated with TRP.

Lemma 5. *For any distinguisher $\mathcal{D}$,*

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathcal{G}_{\mathrm{AE}}}(\mathcal{D}) \leq \frac{\sigma^2}{2^{n+1}} + \frac{q^2}{2^n} \ ,$$

where q is the number of the queries made by $\mathcal{D}$ and σ is the number of invocations of TRP *induced by the queries.*

Proof. This proof utilizes the game playing technique [2].

Without loss of generality, assume that $\mathcal{D}$ makes no repeated queries. The game Gm0 in Fig. 11 implements $\mathcal{G}_{\mathrm{AE}}$. Thus,

$$\Pr\left[\mathcal{D}^{\mathcal{G}_{\mathrm{AE}}} \Rightarrow 1\right] = \Pr\left[\mathcal{D}^{\mathrm{Gm0}} \Rightarrow 1\right] \ .$$

The game Gm1 in Fig. 12 is obtained simply by replacing TRP_F in Gm0 with ρ chosen uniformly at random from functions with the same domain and range as TRP_F. From the PRP/PRF switching lemma [2],

$$\Pr\left[\mathcal{D}^{\mathrm{Gm0}} \Rightarrow 1\right] - \Pr\left[\mathcal{D}^{\mathrm{Gm1}} \Rightarrow 1\right] \leq \frac{\sigma^2}{2^{n+1}} \ .$$

The difference between Gm2 in Fig. 13 and Gm1 is located only on the line 118. In Gm2, t is chosen uniformly at random on the line 118. From Lemma 6,

$$\Pr\left[\mathcal{D}^{\mathrm{Gm1}} \Rightarrow 1\right] - \Pr\left[\mathcal{D}^{\mathrm{Gm2}} \Rightarrow 1\right] \leq \frac{q^2}{2^n} \ .$$

On the other hand, since $\mathcal{D}$ makes no repeated queries,

$$\Pr\left[\mathcal{D}^{\mathrm{Gm2}} \Rightarrow 1\right] = \Pr\left[\mathcal{D}^{\mathcal{R}} \Rightarrow 1\right] \ .$$

This completes the proof. □

Lemma 6. *For Gm1 in Fig. 12 and Gm2 in Fig.13,*

$$\Pr\left[\mathcal{D}^{\mathrm{Gm1}} \Rightarrow 1\right] - \Pr\left[\mathcal{D}^{\mathrm{Gm2}} \Rightarrow 1\right] \leq \frac{q^2}{2^n} \ ,$$

where q is the number of the queries by $\mathcal{D}$.

Proof. Gm1 and Gm2 are equivalent if the values of t obtained on the line 117 are always new. Thus, the probability of a collision of the values of t obtained on the line 117 in Gm1 is evaluated.

Let (N_l, A_l, C_l) be the l-th query by $\mathcal{D}$. In this proof, the variables in $\mathcal{G}_{\mathrm{AE}}$ corresponding to the l-th query is also accompanied with the subscript l. Let $t^{(u)}$ be the value of t just after the line u.

Let (N_l, A_l, C_l) and $(N_{l'}, A_{l'}, C_{l'})$ be distinct queries. If the corresponding tweaks used on the line 118 for ρ are distinct, then the outputs of $\mathcal{G}_{\mathrm{AE}}$ are independent of each other. Thus, it is sufficient to consider the case where $N_l = N_{l'}$ and $m_l = m_{l'}$, which is assumed in the remaining part. Notice that m_l and $m_{l'}$ are the numbers of blocks produced from C_l and $C_{l'}$, respectively.

1. Suppose that $C_l = C_{l'}$. Then, $A_l \neq A_{l'}$ since $\mathcal{D}$ makes no repeated queries.
   (a) Suppose that $a_l \neq a_{l'}$.
      i. Suppose that $A_l \neq \varepsilon$ and $A_{l'} \neq \varepsilon$. Then, $t_l^{(112)}$ and $t_{l'}^{(112)}$ are chosen uniformly at random and independent of each other. Thus, $\Pr\left[t_l^{(117)} = t_{l'}^{(117)}\right] = 1/2^n$.
      ii. Suppose that $A_l \neq \varepsilon$ and $A_{l'} = \varepsilon$. Then, $t_l^{(112)}$ is chosen uniformly at random. Thus, $\Pr\left[t_l^{(117)} = t_{l'}^{(117)}\right] = 1/2^n$.
   (b) Suppose that $a_l = a_{l'} (= a)$.
      i. Suppose that $t_l^{(112)}$ and $t_{l'}^{(112)}$ are produced by ρ with the different tweaks, that is these values are chosen uniformly at random and independent of each other. Thus, $\Pr\left[t_l^{(117)} = t_{l'}^{(117)}\right] = 1/2^n$.

ii. Suppose that $t_l^{(112)}$ and $t_{l'}^{(112)}$ are produced by ρ with the same tweak.

A. Suppose that $A_l[i] = A_{l'}[i]$ for $1 \leq i \leq a-1$ and $A_l[a] \neq A_{l'}[a]$. Then, $t_l^{(106)} \neq t_{l'}^{(106)}$. Thus, since $t_l^{(112)}$ and $t_{l'}^{(112)}$ are chosen uniformly at random and independent of each other, $\Pr\left[t_l^{(117)} = t_{l'}^{(117)}\right] = \Pr\left[t_l^{(112)} = t_{l'}^{(112)}\right] = 1/2^n$.

B. Suppose that $A_l[i] \neq A_{l'}[i]$ for some i such that $1 \leq i \leq a-1$. Then, since $t_l^{(106)}$ and $t_{l'}^{(106)}$ are chosen uniformly at random and independent of each other, $\Pr\left[t_l^{(106)} = t_{l'}^{(106)}\right] = 1/2^n$. Thus,

$$\Pr\left[t_l^{(117)} = t_{l'}^{(117)}\right] = \Pr\left[t_l^{(112)} = t_{l'}^{(112)}\right] = \frac{1}{2^n} + \left(1 - \frac{1}{2^n}\right) \cdot \frac{1}{2^n} \leq \frac{1}{2^{n-1}} \ .$$

2. Suppose that $C_l \neq C_{l'}$. Let $m_l = m_{l'} = m$.

(a) Suppose that $C_l[i] = C_{l'}[i]$ for $1 \leq i \leq m-1$ and $C_l[m] \neq C_{l'}[m]$. If $A_l = A_{l'}$, then $\Pr\left[t_l^{(117)} = t_{l'}^{(117)}\right] = 0$. Otherwise, $\Pr\left[t_l^{(117)} = t_{l'}^{(117)}\right] \leq 1/2^{n-1}$.

(b) Suppose that $C_l[i] \neq C_{l'}[i]$ for some i such that $1 \leq i \leq m-1$. Then, since $t_l^{(117)}$ and $t_{l'}^{(117)}$ are chosen uniformly at random and independent of each other, $\Pr\left[t_l^{(117)} = t_{l'}^{(117)}\right] = 1/2^n$.

The analyses given above shows that the probability of collision of the values of t for a pair of queries is at most $1/2^{n-1}$. Since the number of the queries is at most q, the total probability of the collisions is at most

$$\binom{q}{2} \frac{1}{2^{n-1}} \leq \frac{q^2}{2^n} \ .$$

This completes the proof. □

Let $\mathcal{G}_{\mathrm{MAC}}$ be the tag generation function MAC_Gen instantiated with TRP.

Lemma 7. *For any distinguisher $\mathcal{D}$,*

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathcal{G}_{\mathrm{MAC}}}(\mathcal{D}) \leq \frac{\sigma^2}{2^{n+1}} + \frac{q^2}{2^{n+1}} \ ,$$

where q is the number of the queries made by $\mathcal{D}$ and σ is the number of invocations of TRP *induced by the queries.*

Proof. Omitted since it is similar to the proof of Lemma 5. □

$\underline{\mathcal{G}_{\mathrm{AE}}(N, A, C)}$:
100: $t \leftarrow 0^n$
101: **if** $A \neq \varepsilon$ **then**
102: Pad and Partition A into $(A[1], \ldots, A[a])$
103: **for** $i = 1$ to $a - 1$ **do**
104: $Z^A[i] \leftarrow \mathsf{TRP\_F}(\mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, i, 0, A[i])$
105: **end for**
106: $t \leftarrow Z^A[1] \oplus \cdots \oplus Z^A[a-1] \oplus A[a]$
107: **if** $|A| \bmod n = 0$ **then**
108: $t \leftarrow \mathsf{TRP\_F}(\mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, a-1, 1, t)$
109: **else**
110: $t \leftarrow \mathsf{TRP\_F}(\mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, a-1, 2, t)$
111: **end if**
112: **end if**
113: Partition C into $(C[1], \ldots, C[m])$
114: **for** $i = 1$ to $m - 1$ **do**
115: $Z^C[i] \leftarrow \mathsf{TRP\_F}(\mathsf{flag}_{\mathsf{m}}, N, 2i, 0, C[i])$
116: **end for**
117: $t \leftarrow t \oplus Z^C[1] \oplus \cdots \oplus Z^C[m-1] \oplus C[m]$
118: $t \leftarrow \mathsf{TRP\_F}(\mathsf{flag}_{\mathsf{m}}, N, 2m-1, 1, t)$
119: **return** t

Fig. 11. Game Gm0

$\underline{\mathcal{G}_{\mathrm{AE}}(N, A, C)}$:
100: $t \leftarrow 0^n$
101: **if** $A \neq \varepsilon$ **then**
102: Pad and Partition A into $(A[1], \ldots, A[a])$
103: **for** $i = 1$ to $a - 1$ **do**
104: $Z^A[i] \leftarrow \rho(\mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, i, 0, A[i])$
105: **end for**
106: $t \leftarrow Z^A[1] \oplus \cdots \oplus Z^A[a-1] \oplus A[a]$
107: **if** $|A| \bmod n = 0$ **then**
108: $t \leftarrow \rho(\mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, a-1, 1, t)$
109: **else**
110: $t \leftarrow \rho(\mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, a-1, 2, t)$
111: **end if**
112: **end if**
113: Partition C into $(C[1], \ldots, C[m])$
114: **for** $i = 1$ to $m - 1$ **do**
115: $Z^C[i] \leftarrow \rho(\mathsf{flag}_{\mathsf{m}}, N, 2i, 0, C[i])$
116: **end for**
117: $t \leftarrow t \oplus Z^C[1] \oplus \cdots \oplus Z^C[m-1] \oplus C[m]$
118: $t \leftarrow \rho(\mathsf{flag}_{\mathsf{m}}, N, 2m-1, 1, t)$
119: **return** t

Fig. 12. Game Gm1

$\underline{\mathcal{G}_{\mathrm{AE}}(N, A, C)}$:
100: $t \leftarrow 0^n$
101: **if** $A \neq \varepsilon$ **then**
102: Pad and Partition A into $(A[1], \ldots, A[a])$
103: **for** $i = 1$ to $a - 1$ **do**
104: $Z^A[i] \leftarrow \rho(\mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, i, 0, A[i])$
105: **end for**
106: $t \leftarrow Z^A[1] \oplus \cdots \oplus Z^A[a-1] \oplus A[a]$
107: **if** $|A| \bmod n = 0$ **then**
108: $t \leftarrow \rho(\mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, a-1, 1, t)$
109: **else**
110: $t \leftarrow \rho(\mathsf{flag}_{\mathsf{ad}}, 0^{n/2-s}, a-1, 2, t)$
111: **end if**
112: **end if**
113: Partition C into $(C[1], \ldots, C[m])$
114: **for** $i = 1$ to $m - 1$ **do**
115: $Z^C[i] \leftarrow \rho(\mathsf{flag}_{\mathsf{m}}, N, 2i, 0, C[i])$
116: **end for**
117: $t \leftarrow t \oplus Z^C[1] \oplus \cdots \oplus Z^C[m-1] \oplus C[m]$
118: $t \xleftarrow{R} \{0,1\}^n$
119: **return** t

Fig. 13. Game Gm2

3.1.3 (S)TPRP-Security for Tweakable Even-Mansour in the Ideal Permutation Model

Let P be a public permutation chosen uniformly at random from permutations of n bits. Let P^{-1} be its inverse oracle. $(\mathsf{P}, \mathsf{P}^{-1})$ and TRP are independent of each other.

In this part, we discuss the security of TEM in the ideal permutation model. Let $\mathsf{TEM} = (\mathsf{TEM\_Enc}, \mathsf{TEM\_Dec})$ be TEM instantiated with P instead of P. We show that TEM has $\mathcal{O}(2^{n/2})$ (s)tprp-security.

Since P and P^{-1} are public, in the (S)TPRP game, a distinguisher $\mathcal{D}$ has oracle access to P and P^{-1}. The (s)tprp-advantages of $\mathcal{D}$ are defined as follows:

$$\mathrm{Adv}^{\mathrm{tprp}}_{\mathsf{TEM}}(\mathcal{D}) = \Pr\left[K \xleftarrow{R} \mathcal{K} : \mathcal{D}^{\mathsf{TEM\_Enc}_K, \mathsf{P}, \mathsf{P}^{-1}} \Rightarrow 1\right] - \Pr\left[\mathcal{D}^{\mathsf{TRP\_F}, \mathsf{P}, \mathsf{P}^{-1}} \Rightarrow 1\right] ,$$
$$\mathrm{Adv}^{\mathrm{stprp}}_{\mathsf{TEM}}(\mathcal{D}) = \Pr\left[K \xleftarrow{R} \mathcal{K} : \mathcal{D}^{\mathsf{TEM\_Enc}_K, \mathsf{TEM\_Dec}_K, \mathsf{P}, \mathsf{P}^{-1}} \Rightarrow 1\right] - \Pr\left[\mathcal{D}^{\mathsf{TRP\_F}, \mathsf{TRP\_B}, \mathsf{P}, \mathsf{P}^{-1}} \Rightarrow 1\right] .$$

Theorem 5. *Let n and s be positive integers such that $n/2 - s \geq 1$ and n is even. Let d_1 and d_2 be positive integers. Let (flag, N, i, j) be a tweak of TEM such that $(\mathsf{flag}, N, i, j) \in \{0,1\}^s \times \{0,1\}^{n/2-s} \times \mathbb{Z}_{d_1} \times \mathbb{Z}_{d_2}$ and $(i, j) \neq (0, 0)$. Let $\mathrm{y} \in \mathrm{GF}(2^n)$ such that $\mathrm{y}^i(\mathrm{y}+1)^j \neq 1$ and $\mathrm{y}^i(\mathrm{y}+1)^j \neq \mathrm{y}^{i'}(\mathrm{y}+1)^{j'}$ for any distinct (i, j) and (i', j') in $\mathbb{Z}_{d_1} \times \mathbb{Z}_{d_2} \setminus \{(0, 0)\}$. Then, for any distinguisher $\mathcal{D}$,*

$$\mathrm{Adv}^{\mathrm{stprp}}_{\mathsf{TEM}}(\mathcal{D}) \leq \frac{\sigma^2}{2^{n-1}} + \frac{\sigma}{2^{n/2}} .$$

where σ is the total number of invocations of P and P^{-1} induced by queries made by $\mathcal{D}$.

Proof. The proof also utilizes the game playing technique [2].

Distinguisher $\mathcal{D}$ has four oracles: $\mathcal{E}$, $\mathcal{E}^{-1}$, π and π^{-1}. $\mathcal{E}$ corresponds to $\mathsf{TEM\_Enc}_K$ or $\mathsf{TRP\_F}$. $\mathcal{E}^{-1}$ corresponds to $\mathsf{TEM\_Dec}_K$ or $\mathsf{TRP\_B}$. π corresponds to P. π^{-1} corresponds to P^{-1}. Without loss of generality, we assume that $\mathcal{D}$ is deterministic. We also assume the followings for oracle queries made by $\mathcal{D}$.

- If $\mathcal{D}$ has already obtained the tuple (T_i, M_i, C_i) of a tweak, a plaintext and a ciphertext for $\mathcal{E}$, then $\mathcal{D}$ does not ask (T_i, M_i) to $\mathcal{E}$ nor (T_i, C_i) to $\mathcal{E}^{-1}$.
- If $\mathcal{D}$ has already obtained the tuple (T_i, M_i, C_i) of a tweak, a plaintext and a ciphertext for $\mathcal{E}^{-1}$, then $\mathcal{D}$ does not ask (T_i, M_i) to $\mathcal{E}$ nor (T_i, C_i) to $\mathcal{E}^{-1}$.

– If $\mathcal{D}$ has already obtained a pair (X_i, Y_i) of an input and an output for π, then $\mathcal{D}$ does not ask X_i to π nor Y_i to π^{-1}.
– If $\mathcal{D}$ has already obtained a pair (X_i, Y_i) of an input and an output for π^{-1}, then $\mathcal{D}$ does not ask X_i to π nor Y_i to π^{-1}.

To simplify the notation, $T = (\mathsf{flag}, N, i, j)$ is denoted by $(\bar{N}, (i, j))$, where $\bar{N} = \mathsf{flag} \| N$.
For the game G0 in Fig. 14,

$$\Pr\left[\mathcal{D}^{\mathrm{G0}} \Rightarrow 1\right] = \Pr\left[K \xleftarrow{R} \mathcal{K} : \mathcal{D}^{\mathsf{TRP\_F},\mathsf{TRP\_B},\mathsf{P},\mathsf{P}^{-1}} \Rightarrow 1\right] \ .$$

In the game G1 in Fig. 15, $\mathcal{E}$, $\mathcal{E}^{-1}$, π, and π^{-1} select their outputs uniformly at random. Thus, from the PRP/PRF switching lemma [2],

$$\Pr\left[\mathcal{D}^{\mathrm{G1}} \Rightarrow 1\right] - \Pr\left[\mathcal{D}^{\mathrm{G0}} \Rightarrow 1\right] \leq \frac{\sigma(\sigma-1)}{2^{n+1}} \leq \frac{\sigma^2}{2^{n+1}} \ .$$

The game G2 in Fig. 16 is obtained from G1 by adding some parts for $\mathsf{TEM\_Enc}_K$ and $\mathsf{TEM\_Dec}_K$, which do not affect the outputs of oracles. Thus,

$$\Pr\left[\mathcal{D}^{\mathrm{G2}} \Rightarrow 1\right] = \Pr\left[\mathcal{D}^{\mathrm{G1}} \Rightarrow 1\right] \ .$$

In the game G3 in Fig. 17, Y^M, X^C and Y^N are chosen uniformly at random instead of C, M and L. This change does not affect the behavior of oracles. Thus,

$$\Pr\left[\mathcal{D}^{\mathrm{G3}} \Rightarrow 1\right] = \Pr\left[\mathcal{D}^{\mathrm{G2}} \Rightarrow 1\right] \ .$$

In the game G4 in Fig. 18, ϖ and ϖ^{-1} are introduced, and uniform and random choices in $\mathcal{E}$, $\mathcal{E}^{-1}$ and *compL* are implemented by calls to them. There is no other difference between G3 and G4. Thus,

$$\Pr\left[\mathcal{D}^{\mathrm{G4}} \Rightarrow 1\right] = \Pr\left[\mathcal{D}^{\mathrm{G3}} \Rightarrow 1\right] \ .$$

In the game G5 in Fig. 19, ϖ, and ϖ^{-1} implements P, and P^{-1}, respectively. In G4, outputs of ϖ and ϖ^{-1} are chosen uniformly at random and these are not consistent, while in G5, ϖ is a random permutation and ϖ^{-1} is its inverse. Thus, in G4, the following queries yield the difference between G4 and G5.

– Repeated queries:
  • $Y \leftarrow \varpi(X)$ then $Y^* \leftarrow \varpi(X)$ where Y^* is independent of Y.
  • $X \leftarrow \varpi^{-1}(Y)$ then $X^* \leftarrow \varpi^{-1}(Y)$ where X^* is independent of X.
  • $Y \leftarrow \varpi(X)$ then $X^* \leftarrow \varpi^{-1}(Y)$ where X^* is independent of X.
  • $X \leftarrow \varpi^{-1}(Y)$ then $Y^* \leftarrow \varpi(X)$ where Y^* is independent of Y.
– Queries yielding collisions:
  • $Y \leftarrow \varpi(X)$ then $Y \leftarrow \varpi(X^*)$ where $X \neq X^*$.
  • $X \leftarrow \varpi^{-1}(Y)$ then $X \leftarrow \varpi^{-1}(Y^*)$ where $Y \neq Y^*$.
  • $Y \leftarrow \varpi(X)$ then $X \leftarrow \varpi^{-1}(Y^*)$ where $Y \neq Y^*$.
  • $X \leftarrow \varpi^{-1}(Y)$ then $Y \leftarrow \varpi(X^*)$ where $X \neq X^*$.

Thus, G4 and G5 are equivalent unless collisions occur

– among X^M, X^C, X^N, X^F and X^B, and
– among Y^M, Y^C, Y^N, Y^F and Y^B

in both games. Let Bad_4 be the collision event in G4. From Lemma 8 presented after this proof,

$$\begin{aligned}\Pr\left[\mathcal{D}^{\mathrm{G5}} \Rightarrow 1\right] - \Pr\left[\mathcal{D}^{\mathrm{G4}} \Rightarrow 1\right] &\leq \Pr[\mathsf{Bad}_4] \\ &\leq \frac{3\sigma^2}{2^{n+1}} + \frac{\sigma}{2^{n/2}} \ .\end{aligned}$$

In the game G6 in Fig. 20, ϖ and ϖ^{-1} are removed, and π and π^{-1} take their roles. *compL* does not check the reuse of $\bar{N}$. These changes do not affect the behavior of oracles. Thus,

$$\Pr\left[\mathcal{D}^{\mathrm{G6}} \Rightarrow 1\right] = \Pr\left[\mathcal{D}^{\mathrm{G5}} \Rightarrow 1\right] \ .$$

In G6, $\mathcal{E}$, $\mathcal{E}^{-1}$, π, and π^{-1} implement $\mathsf{TEM\_Enc}_K$, $\mathsf{TEM\_Dec}_K$, P, and P^{-1}, respectively. Namely,

$$\Pr\left[\mathcal{D}^{\mathrm{G6}} \Rightarrow 1\right] = \Pr\left[K \xleftarrow{R} \mathcal{K} : \mathcal{D}^{\mathsf{TEM\_Enc}_K, \mathsf{TEM\_Dec}_K, \mathsf{P}, \mathsf{P}^{-1}} \Rightarrow 1\right] \ .$$

From the discussions above,

$$\begin{aligned}\Pr\left[\mathcal{D}^{\mathrm{G6}} \Rightarrow 1\right] - \Pr\left[\mathcal{D}^{\mathrm{G0}} \Rightarrow 1\right] &\leq \left(\Pr\left[\mathcal{D}^{\mathrm{G5}} \Rightarrow 1\right] - \Pr\left[\mathcal{D}^{\mathrm{G4}} \Rightarrow 1\right]\right) + \left(\Pr\left[\mathcal{D}^{\mathrm{G1}} \Rightarrow 1\right] - \Pr\left[\mathcal{D}^{\mathrm{G0}} \Rightarrow 1\right]\right) \\ &\leq \frac{\sigma^2}{2^{n-1}} + \frac{\sigma}{2^{n/2}} \ .\end{aligned}$$

This completes the proof. □

Lemma 8.

$$\Pr[\mathsf{Bad}_4] \leq \frac{3\sigma^2}{2^{n+1}} + \frac{\sigma}{2^{n/2}} \ .$$

Proof. Let Bad_2 be the collision event in G2. Since G4 is equivalent to G2, $\Pr[\mathsf{Bad}_2] = \Pr[\mathsf{Bad}_4]$. Thus in the following, we evaluate the probability of $\Pr[\mathsf{Bad}_2]$.

In this proof, we consider the following event Perm.

- $\mathcal{E}$ works as a tweakable permutation in accordance with $\mathcal{E}^{-1}$, and
- π works as a permutation in accordance with π^{-1}.

We thus have

$$\begin{aligned}\Pr[\mathsf{Bad}_2] &= \Pr[\mathsf{Bad}_2 \wedge \mathsf{Perm}] + \Pr[\mathsf{Bad}_2 \wedge \neg\mathsf{Perm}] \\ &\leq \Pr[\mathsf{Bad}_2 | \mathsf{Perm}] + \Pr[\neg\mathsf{Perm}] \ .\end{aligned}$$

First we evaluate the probability of $\Pr[\neg\mathsf{Perm}]$. From the PRP/PRF switching lemma [2], we have

$$\Pr[\neg\mathsf{Perm}] \leq \frac{\sigma^2}{2^{n+1}} \ .$$

Next, we evaluate the probability of $\Pr[\mathsf{Bad}|\mathsf{Perm}]$. In the following, we assume that Perm occurs.

In this evaluation, the l-th query of $\mathcal{D}$ is accompanied with the subscript l. Notice that it is a consecutive number for all the oracles. Let $\mathsf{idx}(\mathcal{O})$ be the set of the numbers assigned to queries to the oracle $\mathcal{O}$ made by $\mathcal{D}$. If $\mathcal{O} \neq \mathcal{O}'$, then $\mathsf{idx}(\mathcal{O}) \cap \mathsf{idx}(\mathcal{O}')$ is an empty set.

The variables related to the l-th query of $\mathcal{D}$ is also accompanied with the subscript l. For example, variables related to the query (T_l, M_l) of $\mathcal{D}$ to $\mathcal{E}$ is denoted by X_l^M, Y_l^M, X_l^N, Y_l^N, and so on.

The probability of collisions is evaluated in the following part. Since $\mathcal{D}$ is assumed to be deterministic, the probability of collisions is evaluated in the setting where the oracles perform all the computation except for choices of replies (C, M, Y^F, X^B) to $\mathcal{D}$ after the termination of $\mathcal{D}$. Thus, secret key K and values L are chosen uniformly at random after the termination of $\mathcal{D}$. Then, all the remaining computation is performed for each query and the corresponding reply generated during the interaction with $\mathcal{D}$.

Probabilities that X_l^M collides with $X_{l'}^M$, $X_{l'}^C$, $X_{l'}^N$, $X_{l'}^F$ and $X_{l'}^B$.

1. $\Pr\left[X_l^M = X_{l'}^M\right] \leq 2^{-n}$ for $l \neq l'$ is confirmed as follows. Notice that

$$\begin{aligned}X_l^M &= X_{l'}^M \\ M_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= M_{l'} \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} &= M_l \oplus M_{l'} \ .\end{aligned}$$

   (a) Suppose that $\bar{N}_l = \bar{N}_{l'}$. Then, $L_l = L_{l'}$ due to the list NL.
   - Suppose that $(i_l, j_l) = (i_{l'}, j_{l'})$. Then $\mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l = \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'}$. On the other hand, $M_l \neq M_{l'}$ from the assumption given at the beginning of the proof of Theorem 5 ($\mathcal{D}$ makes no repeated queries). Thus, in this case,

$$\Pr\left[X_l^M = X_{l'}^M\right] = 0 \ .$$

– Suppose that $(i_l, j_l) \neq (i_{l'}, j_{l'})$. Then, since L_l is chosen uniformly at random,

$$\Pr\left[X_l^M = X_{l'}^M\right] = 2^{-n} \ .$$

(b) Suppose that $\bar{N}_l \neq \bar{N}_{l'}$. Then, L_l and $L_{l'}$ are chosen uniformly at random and independent of each other. Thus,

$$\Pr\left[X_l^M = X_{l'}^M\right] = 2^{-n} \ .$$

2. $\Pr\left[X_l^M = X_{l'}^C\right] \leq 2^{-n}$ is confirmed as follows. Notice that

$$\begin{aligned} X_l^M &= X_{l'}^C \\ M_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= M_{l'} \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} &= M_l \oplus M_{l'} \ . \end{aligned}$$

(a) Suppose that $\bar{N}_l = \bar{N}_{l'}$. Then, $L_l = L_{l'}$ due to the list NL.
– Suppose that $(i_l, j_l) = (i_{l'}, j_{l'})$. Then, $\mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l = \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'}$. On the other hand, $M_l \neq M_{l'}$ from the assumptions for Perm and for the proof of Theorem 5 (given at the beginning). Thus,

$$\Pr\left[X_l^M = X_{l'}^C\right] = 0 \ .$$

– Suppose that $(i_l, j_l) \neq (i_{l'}, j_{l'})$. Then, since L_l is chosen uniformly at random,

$$\Pr\left[X_l^M = X_{l'}^C\right] = 2^{-n} \ .$$

(b) Suppose that $\bar{N}_l \neq \bar{N}_{l'}$. Then L_l and $L_{l'}$ are chosen uniformly at random and independent of each other. Thus,

$$\Pr\left[X_l^M = X_{l'}^C\right] = 2^{-n} \ .$$

3. $\Pr\left[X_l^M = X_{l'}^N\right] = 2^{-n}$ since

$$\begin{aligned} X_l^M &= X_{l'}^N \\ M_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= K \| \bar{N}_{l'} \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= K \| \bar{N}_{l'} \oplus M_l \end{aligned}$$

and L_l is chosen uniformly at random and independent of K.

4. $\Pr\left[X_l^M = X_{l'}^F\right] = 2^{-n}$ since

$$\begin{aligned} X_l^M &= X_{l'}^F \\ M_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= X_{l'}^F \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= X_{l'}^F \oplus M_l \end{aligned}$$

and L_l is chosen uniformly at random.

5. $\Pr\left[X_l^M = X_{l'}^B\right] = 2^{-n}$ since

$$\begin{aligned} X_l^M &= X_{l'}^B \\ M_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= X_{l'}^B \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= X_{l'}^B \oplus M_l \end{aligned}$$

and L_l is chosen uniformly at random.

Probabilities that X_l^C collides with $X_{l'}^C$, $X_{l'}^N$, $X_{l'}^F$ and $X_{l'}^B$.

1. $\Pr\left[X_l^C = X_{l'}^C\right] \leq 2^{-n}$ for $l \neq l'$ is confirmed as follows. Notice that

$$\begin{aligned} X_l^C &= X_{l'}^C \\ M_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= M_{l'} \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} &= M_l \oplus M_{l'} \ . \end{aligned}$$

(a) Suppose that $\bar{N}_l = \bar{N}_{l'}$. Then, $L_l = L_{l'}$ due to the list NL.
- Suppose that $(i_l, j_l) = (i_{l'}, j_{l'})$. Then $\mathsf{y}^{i_l}(\mathsf{y}+1)^{j_l} L_l = \mathsf{y}^{i_{l'}}(\mathsf{y}+1)^{j_{l'}} L_{l'}$. On the other hand, $M_l \neq M_{l'}$ from the assumptions for Perm and for the proof of Theorem 5. Thus,

$$\Pr\left[X_l^C = X_{l'}^C\right] = 0 \ .$$

- Suppose that $(i_l, j_l) \neq (i_{l'}, j_{l'})$. Then, since L_l is chosen uniformly at random,

$$\Pr\left[X_l^C = X_{l'}^C\right] = 2^{-n} \ .$$

(b) Suppose that $\bar{N}_l \neq \bar{N}_{l'}$. Then, L_l and $L_{l'}$ are chosen uniformly at random and independent of each other. Thus,

$$\Pr\left[X_l^C = X_{l'}^C\right] = 2^{-n} \ .$$

2. $\Pr\left[X_l^C = X_{l'}^N\right] = 2^{-n}$ since

$$\begin{aligned} X_l^C &= X_{l'}^N \\ M_l \oplus \mathsf{y}^{i_l}(\mathsf{y}+1)^{j_l} L_l &= K \| \bar{N}_{l'} \\ \mathsf{y}^{i_l}(\mathsf{y}+1)^{j_l} L_l &= K \| \bar{N}_{l'} \oplus M_l \end{aligned}$$

and L_l is chosen uniformly at random and independent of K.
3. $\Pr\left[X_l^C = X_{l'}^F\right] = 2^{-n}$ since

$$\begin{aligned} X_l^C &= X_{l'}^F \\ M_l \oplus \mathsf{y}^{i_l}(\mathsf{y}+1)^{j_l} L_l &= X_{l'}^F \\ \mathsf{y}^{i_l}(\mathsf{y}+1)^{j_l} L_l &= X_{l'}^F \oplus M_l \end{aligned}$$

and L_l is chosen uniformly at random.
4. $\Pr\left[X_l^C = X_{l'}^B\right] = 2^{-n}$ since

$$\begin{aligned} X_l^C &= X_{l'}^B \\ M_l \oplus \mathsf{y}^{i_l}(\mathsf{y}+1)^{j_l} L_l &= X_{l'}^B \\ \mathsf{y}^{i_l}(\mathsf{y}+1)^{j_l} L_l &= X_{l'}^B \oplus M_l \end{aligned}$$

and L_l is chosen uniformly at random.

Probabilities that X_l^N collides with $X_{l'}^N$, $X_{l'}^F$ and $X_{l'}^B$.

1. $\Pr\left[X_l^N = X_{l'}^N\right] = 0$ if $\bar{N}_l \neq \bar{N}_{l'}$. If $\bar{N}_l = \bar{N}_{l'}$, then the repeated queries are avoided due to the list NL.
2. $\Pr[X_l^N = X_{l'}^F] \leq 2^{-n/2}$ where $X_l^N = K \| \bar{N}_l$, since $K \in \{0,1\}^{n/2}$ is chosen uniformly at random, and

$$\Pr\left[X_l^N = X_{l'}^F\right] = \begin{cases} 2^{-n/2} & ((\text{The lower } n/2 \text{ bits of } X_{l'}^F) = \bar{N}_l) \\ 0 & ((\text{The lower } n/2 \text{ bits of } X_{l'}^F) \neq \bar{N}_l) \ . \end{cases}$$

3. $\Pr[X_l^N = X_{l'}^B] \leq 2^{-n/2}$ where $X_l^N = K \| \bar{N}_l$, since $K \in \{0,1\}^{n/2}$ is chosen uniformly at random, and

$$\Pr\left[X_l^N = X_{l'}^B\right] = \begin{cases} 2^{-n/2} & ((\text{The lower } n/2 \text{ bits of } X_{l'}^B) = \bar{N}_l) \\ 0 & ((\text{The lower } n/2 \text{ bits of } X_{l'}^B) \neq \bar{N}_l) \ . \end{cases}$$

Probabilities of collisions among X_l^F and $X_{l'}^B$.

1. $\Pr[X_l^F = X_{l'}^F] = 0$, $\Pr[X_l^F = X_{l'}^B] = 0$, and $\Pr[X_l^B = X_{l'}^B] = 0$ from the assumption given at the beginning of the proof of Theorem 5.

Probabilities that Y_l^M collides with $Y_{l'}^M$, $Y_{l'}^C$, $Y_{l'}^N$, $Y_{l'}^F$, and $Y_{l'}^B$.

1. $\Pr\left[Y_l^M = Y_{l'}^M\right] \leq 2^{-n}$ for $l \neq l'$ is confirmed as follows. Notice that

$$\begin{aligned} Y_l^M &= Y_{l'}^M \\ C_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= C_{l'} \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} &= C_l \oplus C_{l'} \ . \end{aligned}$$

   (a) Suppose that $\bar{N}_l = \bar{N}_{l'}$. Then, $L_l = L_{l'}$ due to the list NL.
   - Suppose that $(i_l, j_l) = (i_{l'}, j_{l'})$. Then, $\mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l = \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'}$. On the other hand, $C_l \neq C_{l'}$ from the assumptions for Perm and for the proof of Theorem 5. Thus,

$$\Pr\left[Y_l^M = Y_{l'}^M\right] = 0 \ .$$

   - Suppose that $(i_l, j_l) \neq (i_{l'}, j_{l'})$. Then since L_l is chosen uniformly at random,

$$\Pr\left[Y_l^M = Y_{l'}^M\right] = 2^{-n} \ .$$

   (b) Suppose that $\bar{N}_l \neq \bar{N}_{l'}$. L_l and $L_{l'}$ are chosen uniformly at random and independent of each other. Thus,

$$\Pr\left[Y_l^M = Y_{l'}^M\right] = 2^{-n} \ .$$

2. $\Pr\left[Y_l^M = Y_{l'}^C\right] \leq 2^{-n}$ is confirmed as follows. Notice that

$$\begin{aligned} Y_l^M &= Y_{l'}^C \\ C_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= C_{l'} \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} &= C_l \oplus C_{l'} \ . \end{aligned}$$

   (a) Suppose that $\bar{N}_l = \bar{N}_{l'}$. Then, $L_l = L_{l'}$ due to the list NL.
   - Suppose that $(i_l, j_l) = (i_{l'}, j_{l'})$. Then, $\mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l = \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'}$. On the other hand, $C_l \neq C_{l'}$ from the assumptions for Perm and for the proof of Theorem 5. Thus,

$$\Pr\left[Y_l^M = Y_{l'}^C\right] = 0 \ .$$

   - Suppose that $(i_l, j_l) \neq (i_{l'}, j_{l'})$. Then, since L_l is chosen uniformly at random,

$$\Pr\left[Y_l^M = Y_{l'}^C\right] = 2^{-n} \ .$$

   (b) Suppose that $\bar{N}_l \neq \bar{N}_{l'}$. Since L_l and $L_{l'}$ are chosen uniformly at random and independent of each other,

$$\Pr\left[Y_l^M = Y_{l'}^C\right] = 2^{-n} \ .$$

3. $\Pr\left[Y_l^M = Y_{l'}^N\right] \leq 2^{-n}$ is confirmed as follows. Notice that

$$\begin{aligned} Y_l^M &= Y_{l'}^N \\ C_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= L_{l'} \oplus (K \| \bar{N}_{l'}) \\ L_{l'} \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= C_l \oplus (K \| \bar{N}_{l'}) \ . \end{aligned}$$

   (a) Suppose that $\bar{N}_l = \bar{N}_{l'}$. Then, $L_l = L_{l'}$ due to the list NL. Since $\mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} \neq 1$ and L_l is chosen uniformly at random,

$$\Pr\left[Y_l^M = Y_{l'}^N\right] = 2^{-n} \ .$$

   (b) Suppose that $\bar{N}_l \neq \bar{N}_{l'}$. Since L_l and $L_{l'}$ are chosen uniformly at random and independent of each other,

$$\Pr\left[Y_l^M = Y_{l'}^N\right] = 2^{-n} \ .$$

4. $\Pr\left[Y_l^M = Y_{l'}^F\right] = 2^{-n}$ since

$$\begin{aligned} Y_l^M &= Y_{l'}^F \\ C_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= Y_{l'}^F \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= Y_{l'}^F \oplus C_l \end{aligned}$$

   and L_l is chosen uniformly at random.

5. $\Pr\left[Y_l^M = Y_{l'}^B\right] = 2^{-n}$ since

$$\begin{aligned} Y_l^M &= Y_{l'}^B \\ C_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= Y_{l'}^B \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= Y_{l'}^B \oplus C_l \end{aligned}$$

and L_l is chosen uniformly at random.

Probabilities that Y_l^C collides with $Y_{l'}^C$, $Y_{l'}^N$, $Y_{l'}^F$, and $Y_{l'}^B$.

1. $\Pr\left[Y_l^C = Y_{l'}^C\right] \leq 2^{-n}$ for $l \neq l'$ is confirmed as follows. Notice that

$$\begin{aligned} Y_l^C &= Y_{l'}^C \\ C_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= C_{l'} \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l \oplus \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'} &= C_l \oplus C_{l'} \ . \end{aligned}$$

   (a) Suppose that $\bar{N}_l = \bar{N}_{l'}$. Then, $L_l = L_{l'}$ due to the list NL.
   - Suppose that $(i_l, j_l) = (i_{l'}, j_{l'})$. Then, $\mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l = \mathrm{y}^{i_{l'}}(\mathrm{y}+1)^{j_{l'}} L_{l'}$. On the other hand, $C_l \neq C_{l'}$ from the assumption of the proof of Theorem 5. Thus,

$$\Pr\left[Y_l^C = Y_{l'}^C\right] = 0 \ .$$

   - Suppose that $(i_l, j_l) \neq (i_{l'}, j_{l'})$. Then since L_l is chosen uniformly at random,

$$\Pr\left[Y_l^C = Y_{l'}^C\right] = 2^{-n} \ .$$

   (b) Suppose that $\bar{N}_l \neq \bar{N}_{l'}$. Then, L_l and $L_{l'}$ are chosen uniformly at random and independent of each other. Thus,

$$\Pr\left[Y_l^C = Y_{l'}^C\right] = 2^{-n} \ .$$

2. $\Pr\left[Y_l^C = Y_{l'}^N\right] \leq 2^{-n}$ is confirmed as follows. Notice that

$$\begin{aligned} Y_l^C &= Y_{l'}^N \\ C_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= L_{l'} \oplus (K \| \bar{N}_{l'}) \\ L_{l'} \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= C_l \oplus (K \| \bar{N}_{l'}) \ . \end{aligned}$$

   (a) Suppose that $\bar{N}_l = \bar{N}_{l'}$. Then, $L_l = L_{l'}$ due to the list NL. Since $\mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} \neq 1$ and L_l is chosen uniformly at random, and independent of K,

$$\Pr\left[Y_l^C = Y_{l'}^N\right] = 2^{-n} \ .$$

   (b) Suppose that $\bar{N}_l \neq \bar{N}_{l'}$. Since L_l and $L_{l'}$ are chosen uniformly at random and independent of each other, and independent of K,

$$\Pr\left[Y_l^C = Y_{l'}^N\right] = 2^{-n} \ .$$

3. $\Pr\left[Y_l^C = Y_{l'}^F\right] = 2^{-n}$ since

$$\begin{aligned} Y_l^C &= Y_{l'}^F \\ C_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= Y_{l'}^F \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= Y_{l'}^F \oplus C_l \end{aligned}$$

   and L_l is chosen uniformly at random.
4. $\Pr\left[Y_l^C = Y_{l'}^B\right] = 2^{-n}$ since

$$\begin{aligned} Y_l^C &= Y_{l'}^B \\ C_l \oplus \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= Y_{l'}^B \\ \mathrm{y}^{i_l}(\mathrm{y}+1)^{j_l} L_l &= Y_{l'}^B \oplus C_l \end{aligned}$$

   and L_l is chosen uniformly at random.

Probabilities that Y_l^N collides with $Y_{l'}^N$, $Y_{l'}^F$, and $Y_{l'}^B$.

1. $\Pr\left[Y_l^N = Y_{l'}^N\right] \leq 2^{-n}$ for $l \neq l'$ is confirmed as follows. Notice that

$$\begin{aligned} Y_l^N &= Y_{l'}^N \\ L_l \oplus (K\|\bar{N}_l) &= L_{l'} \oplus (K\|\bar{N}_{l'}) \\ L_l \oplus L_{l'} &= (K\|\bar{N}_l) \oplus (K\|\bar{N}_{l'}) \ . \end{aligned}$$

   $N_l \neq N_{l'}$ due to the list NL, and thereby L_l and $L_{l'}$ are chosen uniformly at random and independent of each other. Thus,

$$\Pr\left[Y_l^N = Y_{l'}^N\right] = 2^{-n} \ .$$

2. $\Pr\left[Y_l^N = Y_{l'}^F\right] = 2^{-n}$ since

$$\begin{aligned} Y_l^N &= Y_{l'}^F \\ L_l \oplus (K\|\bar{N}_l) &= Y_{l'}^F \\ L_l &= Y_{l'}^F \oplus (K\|\bar{N}_l), \end{aligned}$$

   L_l is chosen uniformly at random and independent of K.
3. $\Pr\left[Y_l^N = Y_{l'}^B\right] = 2^{-n}$ since

$$\begin{aligned} Y_l^N &= Y_{l'}^B \\ L_l \oplus (K\|\bar{N}_l) &= Y_{l'}^B \\ L_l &= Y_{l'}^B \oplus (K\|\bar{N}_l) \end{aligned}$$

   and L_l is chosen uniformly at random and independent of K.

Probabilities of collision among Y_l^F and $Y_{l'}^B$.

1. $\Pr[Y_l^F = Y_{l'}^F] = 0$, $\Pr[Y_l^F = Y_{l'}^B] = 0$, and $\Pr[Y_l^B = Y_{l'}^B] = 0$ from the assumption given at the beginning of the proof of Theorem 5.

We thus have

$$\Pr[\mathsf{Bad}_2|\mathsf{Perm}] \leq 2\binom{\sigma}{2}\frac{1}{2^n} + \sigma \cdot \frac{1}{2^{n/2}} \leq \frac{\sigma^2}{2^n} + \frac{\sigma}{2^{n/2}} \ .$$

The first term of the bound is from the collision probabilities which are not from the analyses of randomness of K. The last term of the bound is from the collision probabilities which are from the analyses of randomness of K.

This completes the proof.

□

$\underline{\mathcal{E}(T,M)}$: 100: **return** $\mathsf{TRP\_F}(T,M)$ $\underline{\mathcal{E}^{-1}(T,C)}$: 150: **return** $\mathsf{TRP\_B}(T,C)$	$\underline{\pi(X)}$: 500: **return** $\mathsf{P}(X)$ $\underline{\pi^{-1}(Y)}$: 550: **return** $\mathsf{P}^{-1}(Y)$

Fig. 14. Game G0

$\mathcal{E}(T, M)$:

100: $C \xleftarrow{R} \{0,1\}^n$
101: **return** C

$\mathcal{E}^{-1}(T, C)$:

150: $M \xleftarrow{R} \{0,1\}^n$
151: **return** M

$\pi(X)$:

500: $Y \xleftarrow{R} \{0,1\}^n$
501: **return** Y

$\pi^{-1}(Y)$:

550: $X \xleftarrow{R} \{0,1\}^n$
551: **return** X

Fig. 15. Game G1

Initialization:

0: NL is an empty list.

$\mathcal{E}(T, M)$:

100: Parse T into $(\bar{N}, (i, j))$
101: $L \leftarrow compL(\bar{N})$
102: $X^M \leftarrow M \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
103: $C \xleftarrow{R} \{0,1\}^n$
104: $Y^M \leftarrow C \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
105: **return** C

$\mathcal{E}^{-1}(T, C)$:

150: Parse T into $(\bar{N}, (i, j))$
151: $L \leftarrow compL(\bar{N})$
152: $Y^C \leftarrow C \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
153: $M \xleftarrow{R} \{0,1\}^n$
154: $X^C \leftarrow M \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
155: **return** M

function $compL(\bar{N})$:

200: **if** $\exists(\bar{N}, L') \in NL$ **then**
201: $\quad L \leftarrow L'$
202: **else**
203: $\quad X^N \leftarrow K \| \bar{N}$
204: $\quad L \xleftarrow{R} \{0,1\}^n$
205: $\quad Y^N \leftarrow L \oplus X^N$
206: $\quad NL \leftarrow NL \cup \{(\bar{N}, L)\}$
207: **end if**
208: **return** L

$\pi(X^F)$:

500: $Y^F \xleftarrow{R} \{0,1\}^n$
501: **return** Y^F

$\pi^{-1}(Y^B)$:

550: $X^B \xleftarrow{R} \{0,1\}^n$
551: **return** X^B

Fig. 16. Game G2. The superscripts ^{M}, ^{C}, ^{N}, ^{F}, and ^{B} to X and Y are attached for ease of reference in Lemma 8.

Initialization:

0: NL is an empty list.

$\underline{\mathcal{E}(T, M)}$:

100: Parse T into $(\bar{N}, (i, j))$
101: $L \leftarrow compL(\bar{N})$
102: $X^M \leftarrow M \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
103: $Y^M \xleftarrow{R} \{0,1\}^n$
104: $C \leftarrow Y^M \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
105: **return** C

$\underline{\mathcal{E}^{-1}(T, C)}$:

150: Parse T into $(\bar{N}, (i, j))$
151: $L \leftarrow compL(\bar{N})$
152: $Y^C \leftarrow C \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
153: $X^C \xleftarrow{R} \{0,1\}^n$
154: $M \leftarrow X^C \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
155: **return** M

function $\underline{compL(\bar{N})}$:

200: **if** $\exists(\bar{N}, L') \in NL$ **then**
201: $\quad L \leftarrow L'$
202: **else**
203: $\quad X^N \leftarrow K \| \bar{N}$
204: $\quad Y^N \xleftarrow{R} \{0,1\}^n$
205: $\quad L \leftarrow X^N \oplus Y^N$
206: $\quad NL \leftarrow NL \cup \{(\bar{N}, L)\}$
207: **end if**
208: **return** L

$\underline{\pi(X^F)}$:

500: $Y^F \xleftarrow{R} \{0,1\}^n$
501: **return** Y^F

$\underline{\pi^{-1}(Y^B)}$:

550: $X^B \xleftarrow{R} \{0,1\}^n$
551: **return** X^B

Fig. 17. Game G3

Initialization:

0: NL is an empty list.

$\underline{\mathcal{E}(T, M)}$:

100: Parse T into $(\bar{N}, (i, j))$
101: $L \leftarrow compL(\bar{N})$
102: $X^M \leftarrow M \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
103: $Y^M \leftarrow \varpi(X^M)$
104: $C \leftarrow Y^M \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
105: **return** C

$\underline{\mathcal{E}^{-1}(T, C)}$:

150: Parse T into $(\bar{N}, (i, j))$
151: $L \leftarrow compL(\bar{N})$
152: $Y^C \leftarrow C \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
153: $X^C \leftarrow \varpi^{-1}(Y^C)$
154: $M \leftarrow X^C \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
155: **return** M

function $\underline{compL(\bar{N})}$:

200: **if** $\exists(\bar{N}, L') \in NL$ **then**
201: $\quad L \leftarrow L'$
202: **else**
203: $\quad X^N \leftarrow K \| \bar{N}$
204: $\quad Y^N \leftarrow \varpi(X^N)$
205: $\quad L \leftarrow X^N \oplus Y^N$
206: $\quad NL \leftarrow NL \cup \{(\bar{N}, L)\}$
207: **end if**
208: **return** L

$\underline{\pi(X^F)}$:

500: $Y^F \leftarrow \varpi(X^F)$
501: **return** Y^F

$\underline{\pi^{-1}(Y^B)}$:

550: $X^B \leftarrow \varpi^{-1}(Y^B)$
551: **return** X^B

$\underline{\varpi(X)}$:

700: $Y \xleftarrow{R} \{0,1\}^n$
701: **return** Y

$\underline{\varpi^{-1}(Y)}$:

750: $X \xleftarrow{R} \{0,1\}^n$
751: **return** X

Fig. 18. Game G4

Initialization:

0: NL is an empty list.

$\underline{\mathcal{E}(T, M)}$:

100: Parse T into $(\bar{N}, (i, j))$
101: $L \leftarrow compL(\bar{N})$
102: $X^M \leftarrow M \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
103: $Y^M \leftarrow \varpi(X^M)$
104: $C \leftarrow Y^M \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
105: **return** C

$\underline{\mathcal{E}^{-1}(T, C)}$:

150: Parse T into $(\bar{N}, (i, j))$
151: $L \leftarrow compL(\bar{N})$
152: $Y^C \leftarrow C \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
153: $X^C \leftarrow \varpi^{-1}(Y^C)$
154: $M \leftarrow X^C \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
155: **return** M

function $\underline{compL(\bar{N})}$:

200: **if** $\exists(\bar{N}, L') \in NL$ **then**
201: $\quad L \leftarrow L'$
202: **else**
203: $\quad X^N \leftarrow K \| \bar{N}$
204: $\quad Y^N \leftarrow \varpi(X^N)$
205: $\quad L \leftarrow X^N \oplus Y^N$
206: $\quad NL \leftarrow NL \cup \{(\bar{N}, L)\}$
207: **end if**
208: **return** L

$\underline{\pi(X^F)}$:

500: $Y^F \leftarrow \varpi(X^F)$
501: **return** Y^F

$\underline{\pi^{-1}(Y^B)}$:

550: $X^B \leftarrow \varpi^{-1}(Y^B)$
551: **return** X^B

$\underline{\varpi(X)}$:

700: **return** $\mathsf{P}(X)$

$\underline{\varpi^{-1}(Y)}$:

750: **return** $\mathsf{P}^{-1}(Y)$

Fig. 19. Game G5

$\underline{\mathcal{E}(T, M)}$:

100: Parse T into $(\bar{N}, (i, j))$
101: $L \leftarrow compL(\bar{N})$
102: $X^M \leftarrow M \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
103: $Y^M \leftarrow \pi(X^M)$
104: $C \leftarrow Y^M \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
105: **return** C

$\underline{\mathcal{E}^{-1}(T, C)}$:

150: Parse T into $(\bar{N}, (i, j))$
151: $L \leftarrow compL(\bar{N})$
152: $Y^C \leftarrow C \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
153: $X^C \leftarrow \pi^{-1}(Y^C)$
154: $M \leftarrow X^C \oplus \mathrm{y}^i(\mathrm{y}+1)^j L$
155: **return** M

function $\underline{compL(\bar{N})}$:

200: $X^N \leftarrow K \| \bar{N}$
201: $Y^N \leftarrow \pi(X^N)$
202: $L \leftarrow X^N \oplus Y^N$
203: **return** L

$\underline{\pi(X)}$:

500: **return** $\mathsf{P}(X)$

$\underline{\pi^{-1}(Y)}$:

550: **return** $\mathsf{P}^{-1}(Y)$

Fig. 20. Game G6

Theorem 6. *Let n and s be positive integers such that $n/2-s \geq 1$ and n is even. Let d_1 and d_2 be positive integers. Let (flag, N, i, j) be a tweak of TEM such that $(\mathsf{flag}, N, i, j) \in \{0,1\}^s \times \{0,1\}^{n/2-s} \times \mathbb{Z}_{d_1} \times \mathbb{Z}_{d_2}$ and $(i, j) \neq (0, 0)$. Let $\mathrm{y} \in \mathrm{GF}(2^n)$ such that $\mathrm{y}^i(\mathrm{y}+1)^j \neq 1$ and $\mathrm{y}^i(\mathrm{y}+1)^j \neq \mathrm{y}^{i'}(\mathrm{y}+1)^{j'}$ for any distinct (i, j) and (i', j') in $\mathbb{Z}_{d_1} \times \mathbb{Z}_{d_2} \setminus \{(0, 0)\}$. Then, for any distinguisher $\mathcal{D}$,*

$$\mathrm{Adv}^{\mathrm{tprp}}_{\mathsf{TEM}}(\mathcal{D}) \leq \frac{\sigma^2}{2^{n-1}} + \frac{\sigma}{2^{n/2}} \ ,$$

where σ is the total number of invocations of P and P^{-1} induced by queries made by $\mathcal{D}$.

Proof. Clearly, for any $\mathcal{D}$, there exists a $\mathcal{D}^*$ such that

$$\mathrm{Adv}^{\mathrm{tprp}}_{\mathsf{TEM}}(\mathcal{D}) \leq \mathrm{Adv}^{\mathrm{stprp}}_{\mathsf{TEM}}(\mathcal{D}^*) \ .$$

Thus, from Theorem 5, this theorem holds. □

3.2 Minimum Number of Active S-boxes of 4-round Minalpher-P in Differential Cryptanalysis

In this section, we prove that the minimum number of active S-boxes of 4-round Minalpher-P is 22.

3.2.1 Preliminaries

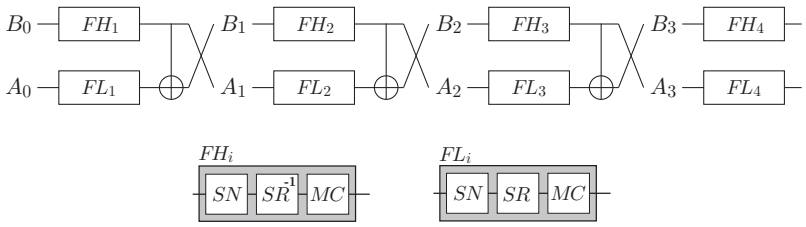


Fig. 21. 4-round Minalpher-P

3.2.1.1 Equivalent Transformation

For simplicity of analysis, we use a different representation of the round function of Minalpher-P. The order of operations is SubNibbles, ShuffleRows, MixColumns, XorMatrix, and then SwapMatrices. The XORing with the round constant is omitted because it does not affect the number of active S-boxes. We can express the 4-round computation of Minalpher-P as Fig. 21, where $FH_i = MC \circ SR^{-1} \circ SN$ and $FL_i = MC \circ SR \circ SN$.

3.2.1.2 Notations

Let X and X^* be values on $\left\{\{0,1\}^4\right\}^{4\times 8}$. Let ΔX be the difference between X and X^* as $\Delta X = X \oplus X^*$. Moreover, $\delta X \in \{0,1\}^{4\times 8}$ is defined as follows:

$$\delta X[i][j] = 1 \ \text{ if } \Delta X[i][j] \neq 0 \ \text{ for any } i = 0,1,2,3 \text{ and } j = 0,1,\ldots,7$$
$$\delta X[i][j] = 0 \ \text{ if } \Delta X[i][j] = 0 \ \text{ for any } i = 0,1,2,3 \text{ and } j = 0,1,\ldots,7$$

For $\delta X \in \{0,1\}^{4\times 8}$, the number of active nibbles ($\mathrm{An}(\delta X)$), the number of active columns ($\mathrm{Ac}(\delta X)$), the number of maximum active nibbles for any row ($\mathrm{Mn}(\delta X)$) and the number of maximum active nibbles for any column ($\mathrm{Mc}(\delta X)$) are defined as follows:

$$\mathrm{An}(\delta X) = \sum_{i=0}^{3}\sum_{j=0}^{7} \delta X[i][j], \qquad \mathrm{Ac}(\delta X) = \#\left\{0 \leq j \leq 7 \middle| \left(\sum_{i=0}^{3} \delta X[i][j]\right) \neq 0\right\},$$
$$\mathrm{Mn}(\delta X) = \max\left\{\sum_{j=0}^{7} \delta X[i][j] \middle| 0 \leq i \leq 3\right\}, \qquad \mathrm{Mc}(\delta X) = \max\left\{\sum_{i=0}^{3} \delta X[i][j] \middle| 0 \leq j \leq 7\right\}.$$

3.2.2 Basic Properties Used in Proof

We show the minimum number of active S-boxes for 4-round Minalpher-P (see Fig. 21). Each number of active S-boxes in the FH_i-function and FL_i-function is equal to $\mathrm{An}(\delta B_i)$ and $\mathrm{An}(\delta A_i)$, respectively. Then the number of active S-boxes in 4-round Minalpher-P is $\sum_{i=0}^{3}(\mathrm{An}(\delta A_i) + \mathrm{An}(\delta B_i))$.

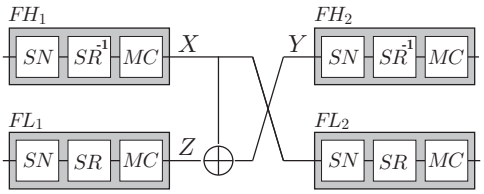


Fig. 22. Properties of 2-round Minalpher-P

3.2.2.1 Properties of 2-round Minalpher-P

First we show several properties of 2-round Minalpher-P, and the FH_i-function and FL_i-function have a MixColumns (MC) whose branch number is 4. Let X and Z be the outputs of FH_1 and FL_1, respectively. Let Y be the input of FH_2. Then, 2-round Minalpher-P satisfies the following properties (see Fig. 22).

1. The sum of the number of active S-boxes in FH_1 and FL_2 is at least $4 \times \mathrm{Ac}(\delta X)$.
2. The sum of the number of active S-boxes in FH_1, FL_1 and FH_2 is at least $4 \times \mathrm{Ac}(\delta Y)$.
3. The sum of the number of active S-boxes in FL_1, FH_2 and FL_2 is at least $4 \times \mathrm{Ac}(\delta Z)$.

3.2.2.2 Properties from Involutive Functionality

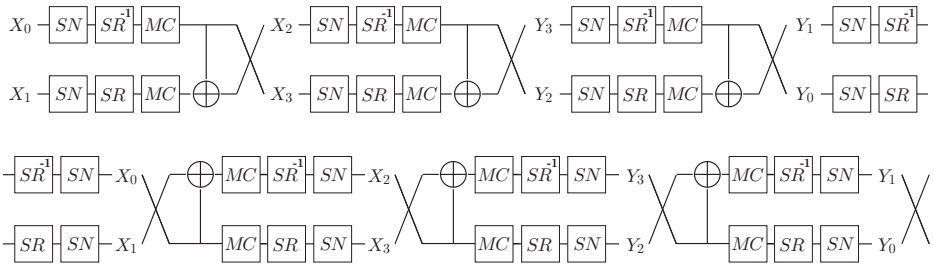


Fig. 23. Properties from Involutive Functionality

We can simplify the proof of the minimum number of active S-boxes by using the fact that Minalpher-P is involutive. Let n_{H1}, n_{H2}, n_{H3} and n_{H4} be numbers of active S-boxes of FH_1, FH_2, FH_3 and FH_4, respectively. Let n_{L1}, n_{L2}, n_{L3} and n_{L4} be numbers of active S-boxes of FL_1, FL_2, FL_3 and FL_4, respectively. We consider a fixed value n. Then if the minimum number of active S-boxes of 4-round Minalpher-P with $n_{H1} = n$ is at least 22, the minimum number of active S-boxes of 4-round Minalpher-P with $n_{L4} = n$ is at least 22. In this section, we prove this property. In Fig. 23, the bottom figure can be obtained by applying an equivalent transformation to the upper one. The forward operation of the upper function and the backward operation of the bottom function are equivalent. Then if we prove the minimum number of active S-boxes against one side, we do not need to prove that against the other side. Similarly, if minimum numbers of active S-boxes of 4-round Minalpher-P with $n_{H2} = n$, $n_{H3} = n$ and $n_{H4} = n$ are at least 22, minimum numbers of active S-boxes of 4-round Minalpher-P with $n_{L3} = n$, $n_{L2} = n$ and $n_{L1} = n$ are at least 22, respectively.

3.2.2.3 Properties from SR and MC

Let A_1 and B_1 be the input of FH_1 and FL_1, respectively. Let A_2 and B_2 be the output of FH_2 and FL_2, respectively. X denotes the output of FH_1 and Y denotes the input of FH_2. Fig. 24 shows 2-round Minalpher-P. Minalpher-P is designed to fulfill the following properties.

Property 1

- *When* A_1 *is nonactive,* $\mathrm{Ac}(\delta B_1) + \mathrm{Ac}(\delta A_2)$ *is at least* 4.
- *When* B_1 *is nonactive,* $\mathrm{Ac}(\delta A_1) + \mathrm{Ac}(\delta A_2)$ *and* $\mathrm{Ac}(\delta A_1) + \mathrm{Ac}(\delta B_2)$ *are at least* 4.

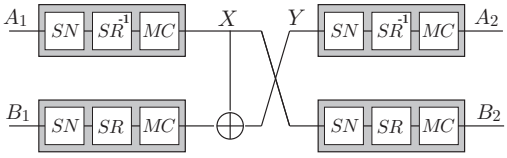


Fig. 24. Properties from SR and MC

– *Otherwise,* $\mathrm{Ac}(\delta A_1) + \mathrm{Ac}(\delta B_2)$ *is at least* 4.

Property 2 *For any* A_1 *satisfying* $\mathrm{Mn}(\delta A_1) = 1$ *and* B_1 *satisfying* $\delta B_1 = \delta A_1$, $\mathrm{An}(\delta B_1) + \mathrm{An}(\delta Y) \geq 6$ *is always satisfied.*

Property 3 *For any* X *and* Y *satisfying* $\mathrm{Ac}(\delta X) = \mathrm{Ac}(\delta Y) = 1$. *The positions of active column of* X *and* Y *are identical. Then* $\mathrm{Ac}(\delta(A_1 \oplus B_1)) \geq \mathrm{Mc}(\delta MC(X))$ *and* $\mathrm{Ac}(\delta(A_2 \oplus B_2)) \geq \mathrm{Mc}(\delta X)$ *are always satisfied, namely,* $\mathrm{Ac}(\delta(A_1 \oplus B_1)) + \mathrm{Ac}(\delta(A_2 \oplus B_2)) \geq 4$ *is always satisfied.*

Property 4

– *When* A_1 *is nonactive and* $\mathrm{Ac}(Y) \geq 2$, $\mathrm{Ac}(\delta B_1) + \mathrm{Ac}(\delta A_2)$ *is at least* 5.
– *When* A_1 *is active and* $\mathrm{Ac}(X) \geq 2$, $\mathrm{Ac}(\delta A_1) + \mathrm{Ac}(\delta B_2)$ *is at least* 5.

3.2.3 Proof Details

We prove the minimum number of active S-boxes by using the case analysis. First we consider 4-round Minalpher-P where any FH_i-function or FL_i-function is nonactive. In Minalpher-P, FL_{i+1} is always nonactive if FH_i is nonactive. Moreover we consider 4-round Minalpher-P where all functions are active. Then all possibilities are covered with the following cases.

1. FL_1 is nonactive.
2. FL_2 is nonactive.
3. FL_3 is nonactive.
4. FL_4 is nonactive.
5. FH_4 is nonactive.
6. All functions are active.

We do not need to care the case 4, because the case 4 is equivalent to the case 2 due to the properties from involutive functionality. Similarly we do not need to care the case 5, because the case 5 is equivalent to the case 1.

3.2.3.1 Case of Nonactive FL_1

Let X be the output of FH_2. Let Y be the input of FH_2. Let Z be the output of FL_3. Fig. 25 shows 4-round Minalpher-P. We pay attention to $\mathrm{Ac}(\delta X)$ and prove the minimum number of active S-boxes by using a case analysis.

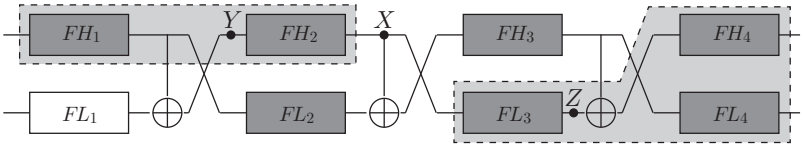


Fig. 25. Case of Nonactive FL_1

– When $\mathrm{Ac}(\delta X) = 1$, $\mathrm{Ac}(\delta Y) + \mathrm{Ac}(\delta Z) \geq 4$ is always satisfied because of Property 1. Then the total number of active S-boxes of FH_1, FH_2, FL_3, FH_4 and FL_4 is at least $4 \times 4 = 16$. Moreover X satisfying $\mathrm{Ac}(\delta X) = 1$ always satisfies $\mathrm{Mn}(\delta Y) = 1$, then the total number of active S-boxes of FL_2 and FH_3 is at least 6 because of Property 2. Therefore the total number of active S-boxes of 4-round Minalpher-P is at least $16 + 6 = 22$.
– When $\mathrm{Ac}(\delta X) \geq 2$, $\mathrm{Ac}(\delta Y) + \mathrm{Ac}(\delta Z) \geq 5$ is always satisfied because of Property 4. Then the total number of active S-boxes of FH_1, FH_2, FL_3, FH_4 and FL_4 is at least $4 \times 5 = 20$. Moreover both FL_2 and FH_3 are active, thus the total number of active S-boxes of 4-round Minalpher-P is at least $20 + 2 = 22$.

3.2.3.2 Case of Nonactive FL_2

Let X be the output of FH_2. Let Y be the input of FH_2. Let Z_1 and Z_2 be the output of FH_3 and FL_3, respectively. Fig. 26 shows 4-round Minalpher-P. We pay attention to $\mathrm{Ac}(\delta X)$ and prove the minimum number of active S-boxes by using a case analysis.

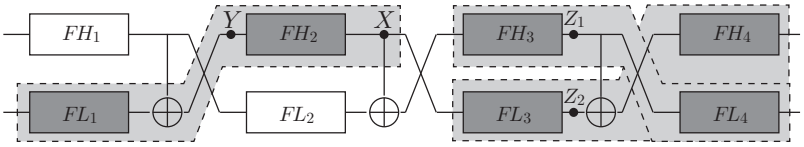


Fig. 26. Case of Nonactive FL_2

– When $\mathrm{Ac}(\delta X) = 1$, $\mathrm{Ac}(\delta Y) + \mathrm{Ac}(\delta Z1) \geq 4$ is always satisfied because of Property 1. Then the total number of active S-boxes of FL_1, FH_2, FH_3 and FL_4 is at least $4 \times 4 = 16$. Moreover X satisfying $\mathrm{Ac}(\delta X) = 1$ always satisfies $\mathrm{Mn}(\delta X) = 1$, then the total number of active S-boxes of FL_3 and FH_4 is at least 6 because of Property 2. Therefore the total number of active S-boxes of 4-round Minalpher-P is at least $16 + 6 = 22$.
– When $\mathrm{Ac}(\delta X) \geq 2$, $\mathrm{Ac}(\delta Y) + \mathrm{Ac}(\delta Z1) \geq 5$ is always satisfied because of Property 4. Then the total number of active S-boxes of FL_1, FH_2, FL_3, FH_4 and FL_4 is at least $4 \times 5 = 20$. Moreover the number of active S-boxes of FH_3 is at least 2 when X satisfies $\mathrm{Ac}(\delta X) \geq 2$. Therefore the total number of active S-boxes of 4-round Minalpher-P is at least $20 + 2 = 22$.

3.2.3.3 Case of Nonactive FL_3

Let X be the input of FH_3. Let Y_1 and Y_2 be the output of FH_1 and FL_1, respectively. Let Z_1 and Z_2 be the input of FH_3 and FL_3, respectively. Fig. 27 shows 4-round Minalpher-P. We pay attention to $\mathrm{Ac}(\delta X)$ and prove the minimum number of active S-boxes by using a case analysis.

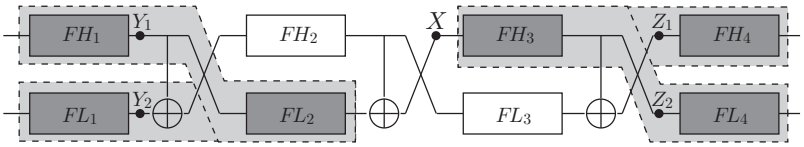


Fig. 27. Case of Nonactive FL_3

– When $\mathrm{Ac}(\delta X) = 1$, $\mathrm{Ac}(\delta Y1) + \mathrm{Ac}(\delta Z2) \geq 4$ and $\mathrm{Ac}(\delta Y2) + \mathrm{Ac}(\delta Z1) \geq 4$ are always satisfied because of Property 1. Then the total number of active S-boxes of FH_1, FL_2, FH_3 and FL_4 is at least $4 \times 4 = 16$. Moreover the total number of active S-boxes of FL_1, FL_2, FH_3 and FH_4 is at least $4 \times 4 = 16$. Now the maximum number of active S-boxes of FL_2 and FH_3 is 8 because of $\mathrm{Ac}(\delta X) = 1$. Then the total number of active S-boxes of 4-round Minalpher-P is at least $16 + 16 - 8 = 24$.

– When $\mathrm{Ac}(\delta X) \geq 2$, $\mathrm{Ac}(\delta Y1) + \mathrm{Ac}(\delta Z2) \geq 5$ is always satisfied because of Property 4. Then the total number of active S-boxes of FH_1, FL_2, FH_3 and FL_4 is at least $4 \times 5 = 20$. Now both FH_3 and FH_4 are active, then the total number of active S-boxes of 4-round Minalpher-P is at least $20 + 2 = 22$.

3.2.3.4 Case of All Active F-functions

Finally we show the minimum number of active S-boxes of 4-round Minalpher-P is 22 when all FH_i/FL_i-functions are active. Let X be the output of FH_2. Let Y be the input of FH_2. Let Z be the output of FH_3. Fig. 28 shows 4-round Minalpher-P. We pay attention to $\mathrm{Ac}(\delta X)$ and prove the minimum number of active S-boxes by using a case analysis.

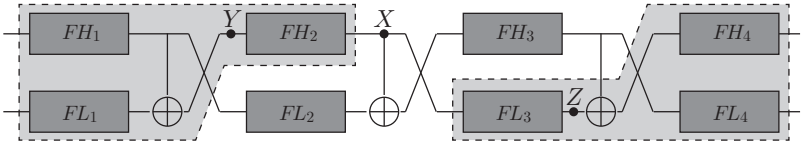


Fig. 28. Case of All Active F-functions

– When $\mathrm{Ac}(\delta X)$ is 1, $\mathrm{Ac}(\delta Y) + \mathrm{Ac}(\delta Z) \geq 4$ is always satisfied because of Property 1. Then the total number of active S-boxes of FH_1, FL_1, FH_2, FL_3, FH_4 and FL_4 is at least $4 \times 4 = 16$. We need to show that the minimum number of active S-boxes of 4-round Minalpher-P is 22 when the sum of active S-boxes of FL_2 and FH_3 is less than 6. Then we consider the following cases,
  1. The number of active S-boxes of FL_2 is 1,
  2. The number of active S-boxes of FL_2 is 2,
  3. The number of active S-boxes of FL_2 is 3,
  4. The number of active S-boxes of FL_2 is 4.

  We prove case 1 and case 2 in the next paragraph, while we do not need to care the other cases. If the number of active S-boxes of FL_2 is more than 2, the necessary condition that the minimum number of active S-boxes of 4-round Minalpher-P is less than 22 is that the number of active S-boxes of FH_3 is less than 3. If minimum numbers of active S-boxes of 4-round Minalpher-P in case 1 and case 2 are at least 22, we do not need to prove case 3 and case 4 because of properties from involutive functionality.
– When $\mathrm{Ac}(\delta X)$ is more than 1, $\mathrm{Ac}(\delta Y) + \mathrm{Ac}(\delta Z) \geq 5$ is always satisfied because of Property 4. Then the total number of active S-boxes of FH_1, FL_1, FH_2, FL_3, FH_4 and FL_4 is at least $4 \times 5 = 20$. Moreover, the total number of active S-boxes of 4-round Minalpher-P is at least $20 + 2 = 22$ because both FL_2 and FH_3 are active.

Case of 1 Active S-box in FL_2

X denotes the output of FH_2, Y denotes the input of FH_3 and Z denotes the output of FL_2. We pay attention to the active columns of X and Z.

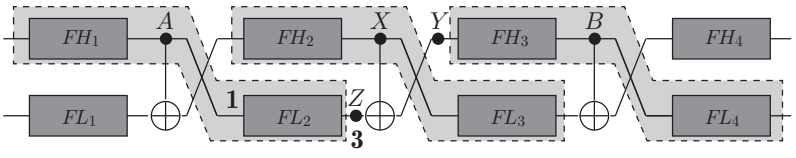


Fig. 29. Case that the Active Column Positions of X and Z Are Different

First we pay attention to 4-round Minalpher-P where the active column position of X and that of Z are different. Let A and B be the output of FH_1 and FH_3, respectively. Fig. 29 shows 4-round Minalpher-P. Now $\mathrm{Mc}(\delta Z) \geq 3$ is always satisfied, and the active column of X and that of Z is different. Then

$\mathrm{Mc}(\delta Y) \geq 3$ is also satisfied, and it derives $\mathrm{Ac}(\delta B) \geq 3$. Therefore $\mathrm{Ac}(\delta A) + \mathrm{Ac}(\delta X) + \mathrm{Ac}(\delta B) \geq 5$ is always satisfied. As a result the total number of active S-boxes of 4-round Minalpher-P is at least $20 + 2 = 22$ because both FL_1 and FH_4 are active.

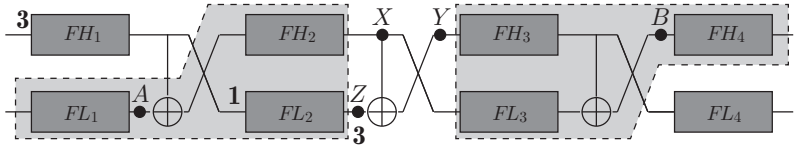


Fig. 30. Case that the Active Column Positions of X and Z Are Identical

Next, we pay attention to 4-round Minalpher-P where the active column position of X and that of Z are identical. Let A and B be the output of FL_1 and the input of FH_4, respectively. Fig. 30 shows 4-round Minalpher-P. X and Y satisfy $\mathrm{Ac}(\delta X) = \mathrm{Ac}(\delta Y) = 1$, and the position of active column of X and Y are the same. In this case, $\mathrm{Ac}(\delta A) + \mathrm{Ac}(\delta B) \geq 4$ is always satisfied because of Property 3. Then the total number of active S-boxes of FL_1, FH_2, FL_2, FH_3, FL_3 and FH_4 is at least $4 \times 4 = 16$. Moreover the number of active S-boxes of FL_4 is at least 3 because of $\mathrm{Ac}(\delta Y) = 1$. Therefore the total number of active S-boxes of 4-round Minalpher-P is at least $16 + 3 + 3 = 22$.

Case of 2 Active S-boxes in FL_2

X denotes the output of FH_2, Y denotes the input of FH_3 and Z denotes the output of FL_2. We pay attention to $\mathrm{Ac}(\delta Z)$.

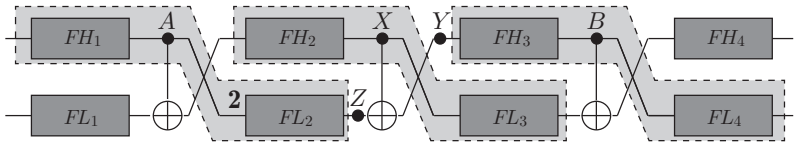


Fig. 31. Case of 2 Active S-boxes in FL_2

First, we pay attention to 4-round Minalpher-P satisfying $\mathrm{Ac}(\delta Z) = 2$. Let A and B be the output of FH_1 and FH_3, respectively. Fig. 31 shows 4-round Minalpher-P. When Z has two active columns and 3 nibbles of each active column are active. Now Y is calculated as $X \oplus Z$ and $\mathrm{Ac}(\delta X) = 1$ is satisfied. Then $\mathrm{Mc}(\delta Y) \geq 3$ is always satisfied, and it derives $\mathrm{Ac}(\delta B) \geq 3$. Therefore $\mathrm{Ac}(\delta A) + \mathrm{Ac}(\delta X) + \mathrm{Ac}(\delta B) \geq 5$ is always satisfied. As a result, the total number of active S-boxes of 4-round Minalpher-P is at least $20 + 2 = 22$ because both FL_1 and FH_4 are active.

Next, we pay attention to 4-round Minalpher-P satisfying $\mathrm{Ac}(\delta Z) = 1$. Moreover we assume that active column positions of X and Z are different. When $\mathrm{Ac}(\delta Z)$ is 1, the number of active nibbles in an active column is 2. Now Y is calculated as $X \oplus Z$ and, and the active column of X and that of Z are different. Then $\mathrm{Mc}(\delta Y) \geq 2$ is always satisfied, and it derives $\mathrm{Ac}(\delta B) \geq 2$. Moreover $\mathrm{Ac}(\delta A) \geq 2$ is always satisfied because $\mathrm{Ac}(\delta Z) = 1$. Therefore $\mathrm{Ac}(\delta A) + \mathrm{Ac}(\delta X) + \mathrm{Ac}(\delta B) \geq 5$ is always satisfied. As a result, the total number of active S-boxes of 4-round Minalpher-P is at least $20 + 2 = 22$ because both FL_1 and FH_4 are active.

Next, we pay attention to 4-round Minalpher-P satisfying $\mathrm{Ac}(\delta Z) = 1$. Moreover we assume that active column positions of X and Z are identical. Let A and B be the output of FL_1 and the input of FH_4, respectively. Fig. 32 shows 4-round Minalpher-P. X and Y satisfy $\mathrm{Ac}(\delta X) = \mathrm{Ac}(\delta Y) = 1$, and active column positions of X and Y are identical. In this case, $\mathrm{Ac}(\delta A) + \mathrm{Ac}(\delta B) \geq 4$ is always satisfied because of Property 3. Then the total number of active S-boxes of FL_1, FH_2, FL_2, FH_3, FL_3 and FH_4 is at least $4 \times 4 = 16$. Moreover the number of active S-boxes of FH_1 is at least 6 because of $\mathrm{Ac}(\delta Z) = 1$. As a result, the total number of active S-boxes of 4-round Minalpher-P is at least $16 + 6 + 1 = 23$ because FL_4 is active.

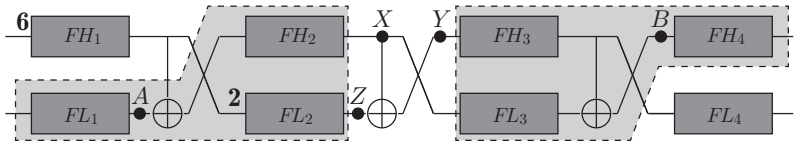


Fig. 32. Case that the Active Column Positions of X and Z Are Identical

3.2.4 Remark for 5-, 6- and 7-round Minalpher-P

The minimum number of active S-boxes for 4-round Minalpher-P is 22. To obtain the minimum number of active S-boxes for 5-, 6- and 7-round Minalpher-P, we use the mixed integer linear programming [39]. As experimental results, minimum numbers of active S-boxes for 5-, 6- and 7-round Minalpher-P are proven to be 41, 58 and 72, respectively.

3.3 Cryptanalysis

In this section, we show initial security evaluation of Minalpher. Note that the key size is 128 bits, and thus Minalpher loses all security when the attacker has a computation power of 128 bits to perform the brute force key recovery attack. In this section, the attacker's main goal is showing forgery attacks and tweak recovery attacks for reduced-round variants. The security proof of Minalpher is based on the 128-bit security for indistinguishability of the tweakable Even-Mansour from an ideal tweakable permutation. Hence, distinguishing attacks up to 2^{128} complexity are also discussed.

3.3.1 Differential Cryptanalysis

The security against classical differential cryptanalysis [6] can be roughly evaluated from the number of active S-boxes and the probability of the maximum differential characteristic for each S-box. The probability of the maximum differential characteristic for each S-box is 2^{-2}. As discussed in Sect. 3.2, the number of minimum active S-boxes is proved for the reduced-round variant of Minalpher-P, which is summarized in Table 2.

Table 2. The Number of Minimum Active S-boxes

Number of Rounds	1	2	3	4	5	6	7	≥ 8
Theoretical Proof	1	4	10	22	N/A	N/A	N/A	N/A
Experimental Proof	1	4	10	22	41	58	72	N/A

From Table 2, the number of active S-boxes is greater than 64 after 6 rounds. The probability of the maximum differential characteristic for 7 rounds is at most 2^{-144}. It seems quite difficult to mount a differential cryptanalysis against full rounds, *i.e.* 17.5 rounds.

On the contrary, the 6-round differential characteristic with probability $2^{-2\times 58} = 2^{-116}$ immediately leads to forgery attacks. In the following, we show two types of forgery attacks for 5.5 rounds.[1]

3.3.1.1 Existential Ciphertext Forgery Attack

The adversary observes valid tuples of 1-block ciphertext C, nonce N and corresponding tag tag. The goal of the adversary is producing a pair of a ciphertext C' and a tag tag' such that $C \neq C'$ and (C', N, tag') will pass the verification by the decryption oracle.

[1] 6 rounds can be attacked, but we prefer to consider the reduced round variant with a half round in the end. Otherwise, the permutation is not involutive.

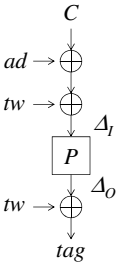


Fig. 33. Computation Structure for the Existential Forgery Attack

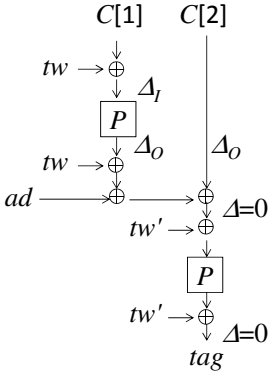


Fig. 34. Computation Structure for the Second Preimage Forgery Attack

For a one-block ciphertext, the computation of the tag can be written as follows, which is also illustrated in Fig. 33.

$$tag \leftarrow P(C \oplus tw \oplus ad) \oplus tw,$$

where, tw is a tweak value and ad is an output from the associated data part. The attack strategy is using a pair of differences (Δ_I, Δ_O) such that the input difference Δ_I is transferred to Δ_O with probability higher than 2^{-128}. Then, by setting $C' \leftarrow C \oplus \Delta_I$ and $tag' \leftarrow tag \oplus \Delta_O$, (C', N, tag') can be valid with probability higher than 2^{-128}.

The propagation of the number of active nibbles in the 5.5-round differential characteristic with 58 active S-boxes is as follows:

$$\begin{array}{ll}
\text{Round 1} & (11,5) \overset{SN}{\to} (11,5) \overset{SR}{\to} (11,5) \overset{SM}{\to} (5,11) \overset{XM}{\to} (5,10) \overset{MC}{\to} (3,8) \overset{RC}{\to} (3,8), \\
\text{Round 2} & (3,8) \overset{SN}{\to} (3,8) \overset{SR}{\to} (3,8) \overset{SM}{\to} (8,3) \overset{XM}{\to} (8,7) \overset{MC}{\to} (4,5) \overset{RC}{\to} (4,5), \\
\text{Round 3} & (4,5) \overset{SN}{\to} (4,5) \overset{SR}{\to} (4,5) \overset{SM}{\to} (5,4) \overset{XM}{\to} (5,3) \overset{MC}{\to} (3,1) \overset{RC}{\to} (3,1), \\
\text{Round 4} & (3,1) \overset{SN}{\to} (3,1) \overset{SR}{\to} (3,1) \overset{SM}{\to} (1,3) \overset{XM}{\to} (1,4) \overset{MC}{\to} (3,4) \overset{RC}{\to} (3,4), \\
\text{Round 5} & (3,4) \overset{SN}{\to} (3,4) \overset{SR}{\to} (3,4) \overset{SM}{\to} (4,3) \overset{XM}{\to} (4,5) \overset{MC}{\to} (8,3) \overset{RC}{\to} (8,3), \\
\text{Round 5.5} & (8,3) \overset{SN}{\to} (8,3) \overset{SR}{\to} (8,3) \overset{SM}{\to} (3,8),
\end{array}$$

where the left and right numbers in a parenthesis represent the number of active nibbles in the half state A and B, respectively. The characteristic is also depicted in Fig. 35.

We fix the input difference of all the 16 active nibbles to be identical. According to the property of the S-box, the maximum probability of the differential characteristic for any active S-box is 2^{-2}. If all active S-box satisfy the differential transition with a maximum probability, the following linear computation part, *i.e.*, XorMatrix and MixColumns, can satisfy the above differential propagation. The number of active S-boxes is 16 for the first round, 11 for the second round, 9 for the third round, 4 for the fourth round, 7 for the fifth round, and 11 for the last 0.5 round, in total 58. Therefore, with a probability of $2^{-2\times 58} = 2^{-116}$, the above differential propagation is satisfied. Finally, with 2^{116} different choices of (C, tag) and (C', tag'), the adversary succeeds in the forgery attack.

Note that, a classical existential forgery attack on MAC schemes finds two distinct input messages M, M' producing the same tag value. While, the attack in this section is different in the sense that the adversary's goal is finding two distinct ciphertexts C, C' producing the same tag value. Hence, we call the attack existential ciphertext forgery attack. Also note that the classical existential forgery attack in the AEAD mode is hard in the nonce respect model, because the same nonce cannot be reused under the same key. However, because the MAC mode does not use the nonce, the above differential cryptanalysis can be applied to the classical existential forgery attack on the MAC mode directly.

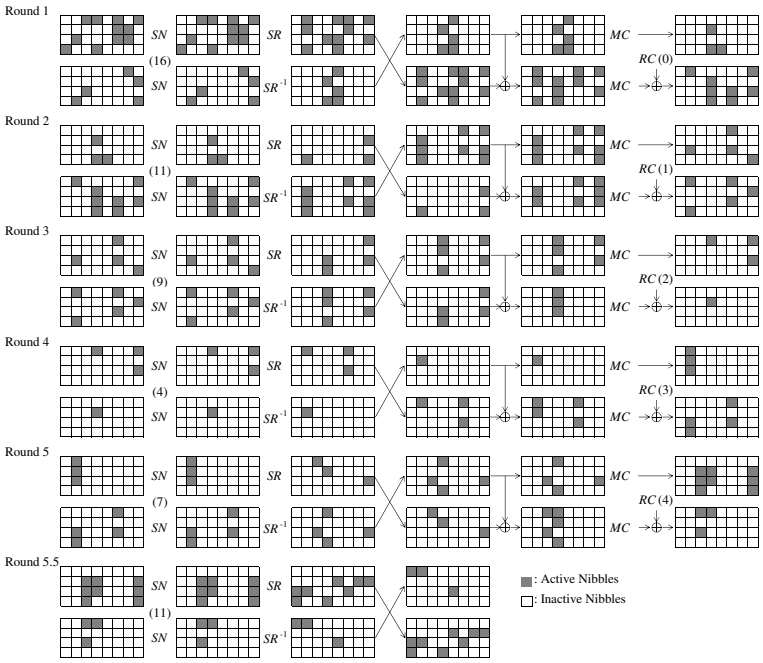


Fig. 35. 5.5-round Differential Characteristic with 58 Active S-boxes

3.3.1.2 Second Preimage Ciphertext Forgery Attack

The above existential ciphertext forgery attack can be converted into the 5.5-round second preimage forgery attack, in which the goal of the adversary is, with a given tuple of (C, N, tag), finding another ciphertext C' which results in the given tag tag. To achieve this goal, the size of the given ciphertext must be two blocks or longer. In this section, the size of the given ciphertext is assumed to be two blocks. (Even if it is longer than two blocks, the adversary can do the same by focusing on the last two-block computation). For a two-block ciphertext, the computation of the tag can be written as follows, which is also illustrated in Fig. 34.

$$tag \leftarrow P(P(C[1] \oplus tw) \oplus ad \oplus C[2] \oplus tw') \oplus tw'.$$

The adversary sets $C' \leftarrow (C[1] \oplus \Delta_I \| C[2] \oplus \Delta_O)$. Δ_I on the input to the first block will return Δ_O after 5.5 rounds with probability 2^{-116}, and this will be canceled out by the difference of $C[2]$. Therefore, with probability 2^{-116}, (C', N, tag) can be valid.

3.3.2 Linear Cryptanalysis

Similarly to the differential cryptanalysis, the resistance against linear cryptanalysis can be explained with Table 2. Because the probability of the maximum linear characteristic is 2^{-2} for each S-box, the probability of the maximum linear characteristic for 7 rounds is at most 2^{-144}. It seems quite difficult to mount a linear cryptanalysis against full rounds.

3.3.3 Truncated Differential Cryptanalysis

3.3.3.1 6-round Iterative Differential Characteristic

We first introduce the 6-round iterative differential characteristic, which is later used for the truncated differential cryptanalysis. The iterative characteristic is given in Fig. 36. The number of active S-boxes transfers as $8 \rightarrow 8 \rightarrow 16 \rightarrow 8 \rightarrow 8 \rightarrow 16 \rightarrow 8$ in each round. Moreover, with appending one more round in the end, the 7-round characteristic involves 72 S-boxes. This matches the number in Table 2, thus the characteristic has the minimum number of active S-boxes for 7 rounds.

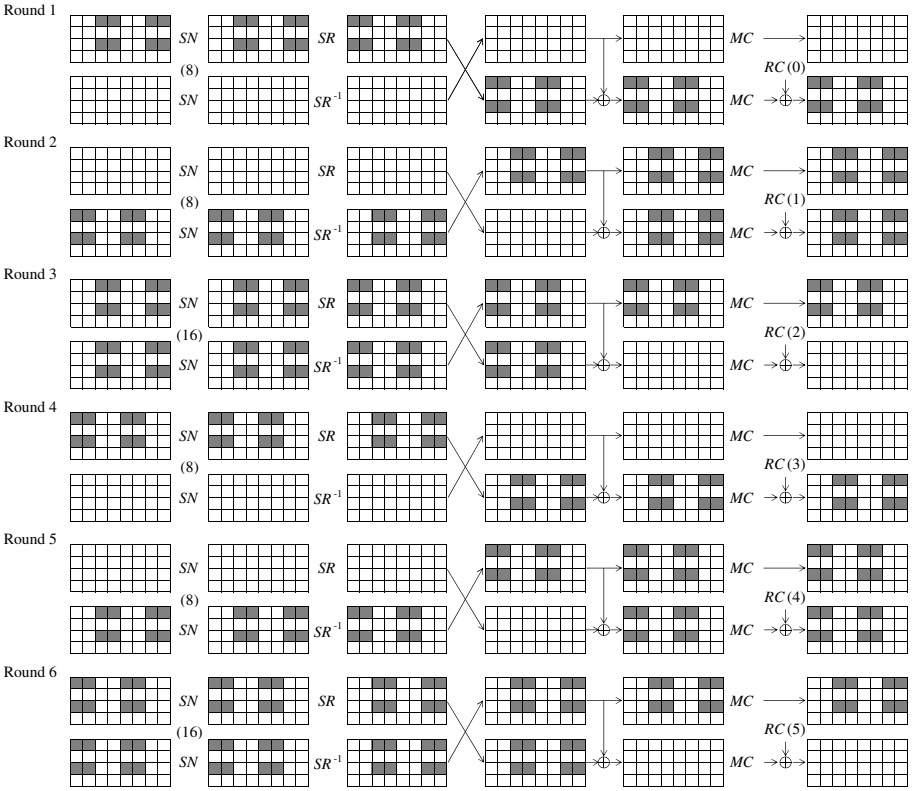


Fig. 36. 6-round Iterative Differential Characteristic

3.3.3.2 7.5-round Partial Tweak Recovery Attack

With the above characteristic, a partial tweak recovery attack can be performed for 7.5 rounds. In this attack, we use the first 5 rounds of the iterative differential characteristic and extend it by 1 round in backward and by 1.5 rounds in forward. The entire characteristic is shown in Fig. 37.

In the truncated differential cryptanalysis, the attacker focuses on whether each nibble is active or inactive. Hence, SN and AC operations do not give any impact to the truncated differential characteristic, and only the linear operations can affect it. We start with summarizing the property of the MC operation.

1. 4 active nibbles can shrink to 2 active nibbles with probability 2^{-8}.
2. 3 active nibbles can shrink to 1 active nibble with probability 2^{-8}.
3. 3 active nibbles can shrink to 2 active nibbles with probability 2^{-4}.
4. 2 active nibbles can be 2 active nibbles again with probability 2^{-4}.

However, if the impact of the XM operation is considered, the analysis can be complicated. We carefully evaluate the probability of the truncated differential transition in each round. Remember that each state is denoted by the notations defined in Sect. 1.4.

Round 1: For each column with active nibbles in A_0^{MC}, *i.e.* $A_0^{MC}[\cdot][2]$, $A_0^{MC}[\cdot][3]$, $A_0^{MC}[\cdot][6]$, and $A_0^{MC}[\cdot][7]$, the truncated differential characteristic is satisfied with probability 2^{-8}. The total probability in this round is $(2^{-8})^4 = 2^{-32}$.

Round 2: For each column with active nibbles in A_1^{MC}, the truncated differential characteristic is satisfied with probability 2^{-4}. The total probability in this round is $(2^{-4})^4 = 2^{-16}$.

Round 3: For each column with active nibbles in A_2^{MC}, the truncated differential characteristic is satisfied with probability 2^{-4}. The difference in B_2^{XM} is the same as the one in A_2^{XM}. Thus, when the differential transition from A_2^{XM} to A_3^{MC} is satisfied, the differential transition from B_2^{XM} to B_3^{MC} is also satisfied. The total probability in this round is $(2^{-4})^4 = 2^{-16}$.

Round 4: The XM operation is probabilistic in this round. For column position i, where $i = 0, 1, 4, 5$, the condition that 4 active nibbles shrink to 2 active nibbles after the XM and MC operations is

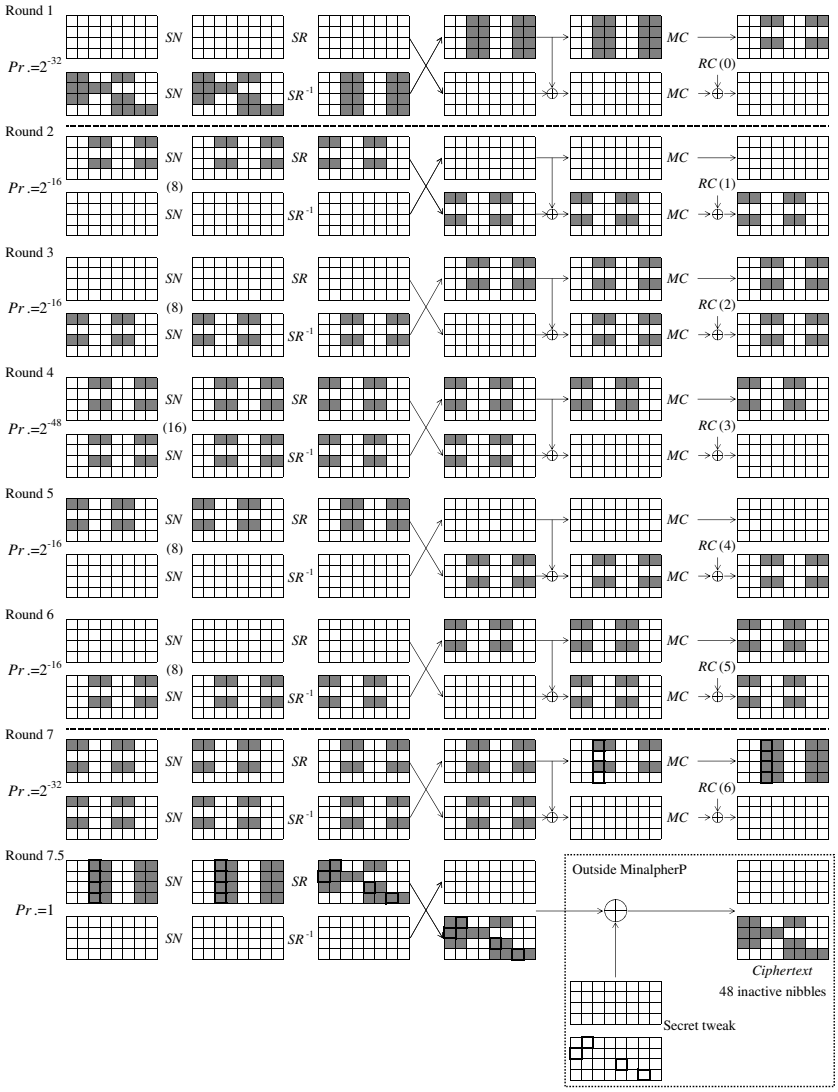


Fig. 37. 7.5-round Partial Tweak Recovery with Truncated Differential Attack

$\Delta A_3^{XM}[0][i] = \Delta A_3^{XM}[2][i] = \Delta B_3^{XM}[0][i] = \Delta B_3^{XM}[2][i]$. This is satisfied with probability 2^{-12} for each column i. The total probability in this round is $(2^{-12})^4 = 2^{-48}$.

Round 5: With the similar evaluation as Round 2, the total probability in this round is 2^{-16}.

Round 6: With the similar evaluation as Round 3, the total probability in this round is 2^{-16}.

Round 7: The only probabilistic event in this round is the difference cancellation in the XM operation. Hence, the condition is $A_6^{XM}[j][i] = B_6^{XM}[j][i]$ for each pair of (i, j), where $i \in \{2, 3, 6, 7\}, j \in \{0, 2\}$. There are 8 choices of (i, j), thus the total probability in this round is $(2^{-4})^8 = 2^{-32}$.

In the end, the probability for 7.5 rounds is $2^{-32-16-16-48-16-16-32} = 2^{-176}$. After 7.5 rounds, the ciphertext is generated by taking the XOR with a secret tweak value. Note that as long as the above characteristic is satisfied, 48 nibbles of the ciphertext are inactive.

The attack is performed in the nonce-misuse model. The attack procedure is as follows.

1. Fix inactive nibbles of the plaintext to a randomly chosen value. Prepare 2^{64} all different values for the 16 active nibbles in the plaintext, which produces 2^{127} pairs in one structure. By changing the inactive nibble values 2^{50} times, prepare 2^{50} structures, which produces 2^{177} pairs in total.
2. For each structure, query 2^{64} plaintexts under the same nonce and key to obtain the corresponding ciphertexts.

3. Sort those 2^{64} ciphertexts by using 48 inactive nibble values as index. If a pair colliding in 48 inactive nibbles of the ciphertext is found, pick up the pair. The pair is expected to follow the truncated differential characteristic in Fig. 37.
4. Repeat Step 3 for 2^{50} structures. Two pairs are expected to satisfy the characteristic.
5. Guess the tweak value partially. This can be done in column by column. For example, guess 2^{16} values of tweak in nibble positions $(0,1)$, $(1,0)$, $(2,4)$ and $(3,6)$, which are stressed by bold lines in Fig. 37. With the partial decryption up to state A_6^{XM}, only the correct guess lead to 2 active nibbles at $A_6^{XM}[\cdot][2]$ for both pairs.
6. Many tweak nibbles can be recovered in the same method. We omit the remaining procedure.

In the above procedure, 2^{177} pairs of plaintexts are generated. Because the probability of the truncated differential characteristic is 2^{-176}, two pairs are expected to satisfy the characteristic. On the contrary, the probability that 48 nibbles become inactive with 2^{177} pairs is $2^{177-(48\times4)} = 2^{-15}$, which is unlikely to occur. Thus, the detected two pairs are right pairs. Each guessed tweak generates 2 active nibbles at the fixed positions with probability 2^{-8}. Thus, by testing two pairs, the number of correct tweak guess becomes $2^{16-8-8} = 1$. The success probability increases by using more data complexity.

In the end, the data complexity of the attack is 2^{114} chosen plaintexts. It also requires 2^{114} memory accesses to deal with those plaintexts and obtained ciphertexts. The analysis can be done for each structure, thus the memory requirement is for storing 2^{64} plaintexts. Note that the computational cost for the tweak guess is negligible compared to processing 2^{64} texts.

3.3.4 Boomerang Attack and Amplified Boomerang Attack

3.3.4.1 Attack Scenario (Nonce Misuse Setting)

The boomerang attack [51] divides the target cipher E into two subsequent subciphers $E_1 \circ E_0$. Suppose that there exists a differential characteristic for E_0 which propagates the input difference Δ_{IN} to an output difference represented by α with probability p. Also suppose that there exists a differential characteristic for E_1 which propagates the input difference β to an output difference represented by Δ_{OUT} with probability q. The framework of the boomerang attack for authenticated encryption schemes is as follows.

1. The adversary chooses a pair of plaintexts (P_0, P_1) such that their difference $P_0 \oplus P_1 = \Delta_{IN}$. (P_0, P_1) are passed to the encryption oracle with the same nonce value N, and the adversary obtains the corresponding ciphertexts (C_0, C_1).
2. The adversary generates (C_2, C_3) by $C_2 \leftarrow C_0 \oplus \Delta_{OUT}$ and $C_3 \leftarrow C_1 \oplus \Delta_{OUT}$. (C_2, C_3) are passed to the decryption oracle with the same nonce value N, and the adversary obtains the corresponding plaintexts (P_2, P_3).
3. The equation $P_2 \oplus P_3 = \Delta_{IN}$ is satisfied with probability p^2q^2.

Note that the attack requires the misuse of the nonce, *i.e.*, the same nonce value needs to be used when P_0 and P_1 are passed to the encryption oracle. One may think that the queries to the encryption oracle can be avoided in different attack scenarios, *e.g.*, starting from choosing the ciphertext or attacking the MAC mode. However, considering that the boomerang needs to return (Enc-then-Dec or Dec-then-Enc), we have not found any attack scenario that can work without assuming the nonce misuse.[2]

Our attack is a tweak recovery attack for 6.5 rounds. The first 3 rounds are set to E_0 and the last 3.5 rounds are set to E_1. With the similar notation as the previous section, the two differential characteristics in our boomerang attack can be represented as follows, which are also depicted in Fig. 38.

Differential characteristic for E_0

$$\begin{aligned}
&\text{Round 1} && (3,3) \overset{SN}{\to} (3,3) \overset{SR}{\to} (3,3) \overset{SM}{\to} (3,3) \overset{XM}{\to} (3,0) \overset{MC}{\to} (1,0) \overset{RC}{\to} (1,0),\\
&\text{Round 2} && (1,0) \overset{SN}{\to} (1,0) \overset{SR}{\to} (1,0) \overset{SM}{\to} (0,1) \overset{XM}{\to} (0,1) \overset{MC}{\to} (0,3) \overset{RC}{\to} (0,3),\\
&\text{Round 3} && (0,3) \overset{SN}{\to} (0,3) \overset{SR}{\to} (0,3) \overset{SM}{\to} (3,0) \overset{XM}{\to} (3,3) \overset{MC}{\to} (9,9) \overset{RC}{\to} (9,9),
\end{aligned}$$

[2] Impossibility of the rectangle attack proposed by Biham *et al.* [5], which uses only one of Enc or Dec, on Minalpher is later discussed in the end of this section.

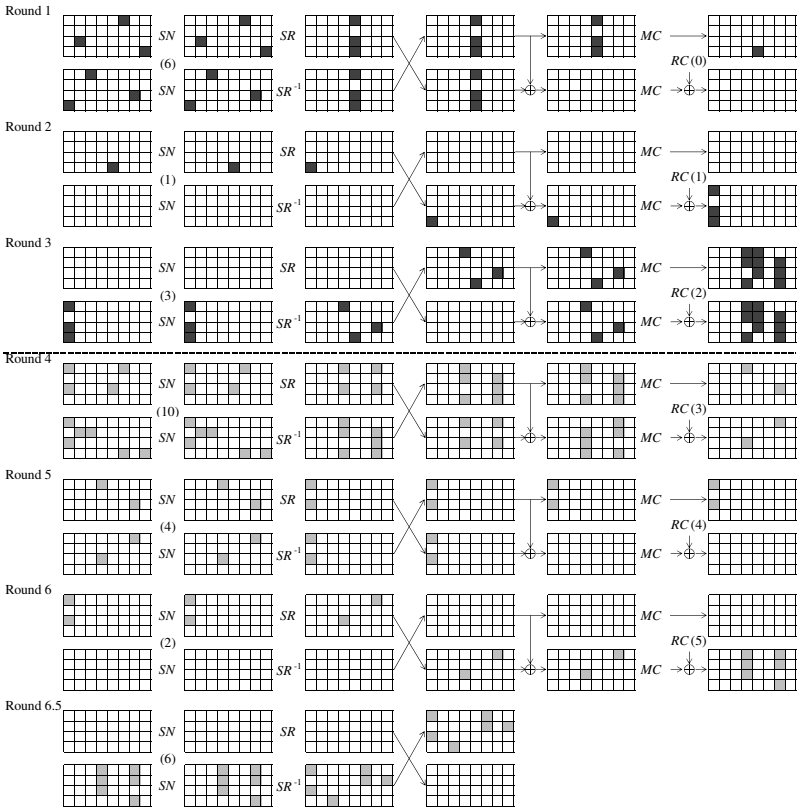


Fig. 38. The Differential Characteristics for Amplified Boomerang Attack. Dark gray cells and light gray cells represent active nibbles for the E_0 characteristic and E_1 characteristic, respectively.

Differential characteristic for E_1

$$
\begin{aligned}
&\text{Round 4} && (4,6) \overset{SN}{\to} (4,6) \overset{SR}{\to} (4,6) \overset{SM}{\to} (6,4) \overset{XM}{\to} (6,6) \overset{MC}{\to} (2,2) \overset{RC}{\to} (2,2),\\
&\text{Round 5} && (2,2) \overset{SN}{\to} (2,2) \overset{SR}{\to} (2,2) \overset{SM}{\to} (2,2) \overset{XM}{\to} (2,0) \overset{MC}{\to} (2,0) \overset{RC}{\to} (2,0),\\
&\text{Round 6} && (2,0) \overset{SN}{\to} (2,0) \overset{SR}{\to} (2,0) \overset{SM}{\to} (0,2) \overset{XM}{\to} (0,2) \overset{MC}{\to} (0,6) \overset{RC}{\to} (0,6),\\
&\text{Round 6.5} && (0,6) \overset{SN}{\to} (0,6) \overset{SR}{\to} (0,6) \overset{SM}{\to} (6,0).
\end{aligned}
$$

Note that the active nibble positions of the output from E_0 and input to E_1 overlap in one nibble. We experimentally verified that for any active nibble byte positions, the characteristic for E_0 and E_1 will overlap at least in one nibble. Here we assume that the inconsistency of connecting two independent characteristics [40] do not occur in this attack.

Similarly to the differential cryptanalysis, we fix the input difference of all the $3 + 3 = 6$ active nibbles to be identical. According the property of the S-box, the maximum probability of the differential characteristic for any active S-box is 2^{-2}. If all active S-boxes satisfy the differential transition with a maximum probability, the following linear computation part, *i.e.*, XorMatrix and MixColumns, can satisfy the above differential propagation. The number of active nibbles for each SN operation is 6 for the first round, 1 for the second round and 3 for the third round. Therefore, the differential characteristic for P_0 has the probability $p = 2^{-2(6+1+3)} = 2^{-20}$, which leads to $p^2 = 2^{-40}$. Similarly, the number of active nibbles for each SN operation is 10 for the fourth round, 4 for the fifth round, 2 for the sixth round and 6 for the last half round. Therefore, the differential characteristic for P_1 has the probability $q = 2^{-2(10+4+2+6)} = 2^{-44}$, which leads to $q^2 = 2^{-88}$. Finally, by examining the boomerang attack framework $(p^2q^2)^{-1} = 2^{128}$ times, the adversary can find a boomerang quartet, or (P_1, P_2, P_3, P_4) and (C_1, C_2, C_3, C_4) that satisfy the two characteristics.

Unfortunately, the above simple boomerang attack requires 2×2^{128} queries for both of the encryption and decryption oracles. Thus, the data complexity is 2^{130}, which is more than the claimed security level of 2^{128}. However, the attack can be valid by considering multiple differential characteristics as an amplified boomerang attack [23], which is explained in the next paragraph.

3.3.4.2 Reducing Data Complexity with Amplified Boomerang Attack

The simple boomerang attack uses only a single differential characteristic, while the amplified boomerang attack considers the multiple differential characteristics. Let $Pr[\Delta_{IN} \rightarrow \alpha]$ be the probability of the differential characteristic for E_0, *i.e.*, p. The amplified boomerang attack considers all possible differences for α, and its probability is represented as

$$\hat{p} = \sqrt{\sum_{\alpha} (Pr[\Delta_{IN} \rightarrow \alpha])^2}.$$

Similarly, the probability for E_1 is represented by

$$\hat{q} = \sqrt{\sum_{\beta} (Pr[\beta \rightarrow \Delta_{OUT}])^2}.$$

With the same framework as the simple boomerang attack, the probability that $P_2 \oplus P_3 = \Delta_{IN}$ is satisfied increases to $\hat{p}^2\hat{q}^2$.

Let us consider the probability of the differential transition for the first round. The input difference is fixed to be identical for all nibbles, and if the output differences from all the S-boxes are identical, we obtain the desired differential propagation. Due to the design policy of our S-box, for any input difference, there is one output difference provided with probability 2^{-2}, and there are six other output differences provided with a probability of 2^{-3}. The probability that all S-boxes produce an identical difference is

$$(2^{-2})^6 + 6 \cdot (2^{-3})^6.$$

Its squared value is $(2^{-4})^6 + 6 \cdot (2^{-6})^6 \approx 2^{-24}$. From the same reason, the probability for the second round is $(2^{-2})^1 + 6 \cdot (2^{-3})^1$, and its squared value is $(2^{-4})^1 + 6 \cdot (2^{-6})^1 \approx 2^{-2.68}$. The evaluation for the third round is a bit different because the cancellation after the SN operation is not necessary. Therefore, we can have different output difference for each S-box, and only the condition is that the same active nibbles produce the same output difference between two pairs. The squared probability is calculated by

$$(2^{-4})^3 + \binom{3}{2} \cdot 6 \cdot (2^{-4})^2 \cdot (2^{-6}) + \binom{3}{1} \cdot 6^2 \cdot (2^{-4}) \cdot (2^{-6})^2 + 6^3 \cdot (2^{-6})^3 \approx 2^{-8.03}.$$

Finally, $\hat{p}^2$ is calculated as $2^{-24} \cdot 2^{-2.68} \cdot 2^{-8.03} = 2^{-34.71}$.

Regarding $\hat{q}$, we evaluate it in the reverse order, *i.e.*, from round 6.5 to round 4.

$$\begin{aligned}
&\text{Round 6.5 :} && (2^{-4})^6 + 6 \times (2^{-6})^6 \approx 2^{-24},\\
&\text{Round 6 :} && (2^{-4})^4 + 6 \times (2^{-6})^4 \approx 2^{-15.97},\\
&\text{Round 5 :} && (2^{-4})^2 + 6 \times (2^{-6})^2 \approx 2^{-7.54},\\
&\text{Round 4 :} && \sum_{i=0}^{10} \binom{10}{i} \cdot 6^{10-i} \cdot (2^{-4})^i \cdot (2^{-6})^{10-i} \approx 2^{-26.78}.
\end{aligned}$$

In total, $\hat{q}^2$ is calculated as $2^{-24} \cdot 2^{-15.97} \cdot 2^{-7.54} \cdot 2^{-26.78} = 2^{-74.29}$.

The data complexity of the entire attack is $\hat{p}^{-2}\hat{q}^{-2}$ quartets, which is $2^{34.71} \cdot 2^{74.29} \cdot 4 = 2^{111.00}$. If such a quartet is generated, the adversary can eventually recover several nibbles of the tweak value.

3.3.4.3 Impossibility of Rectangle Attack

In many cases, the adaptively chosen-ciphertext or the adaptively chosen-plaintext setting of the (amplified) boomerang attack can be avoided by using the rectangle attack framework [5]. This also allows the adversary to attack the MAC mode, in which the nonce is never used and the decryption oracle never exists. We show that applying the rectangle attack to Minalpher is impossible within the claimed security. The idea of the rectangle attack for a b-bit block-cipher is as follows.

1. Generate $(\hat{p}\hat{q})^{-1}2^{b/2}$ plaintext pairs satisfying the difference Δ_{IN}.
2. Making all possible quartets. Then, about $(\hat{p}^2\hat{q}^2)^{-1}2^b$ quartets are generated.
3. One quartet is expected to satisfy $\hat{p}$ in E_0 for two pairs, to have the b-bit distance β between two pairs in the middle of E_0 and E_1, and to satisfy $\hat{q}$ in E_1 for two pairs.
4. One ciphertext quartet is expected to satisfy Δ_{OUT} in two pairs.

In the end, by satisfying $(\hat{p}\hat{q})^{-1} < 2^{b/2}$ or $(\hat{p}^2\hat{q}^2)^{-1} < 2^b$, which is the same condition as the (amplified) boomerang attack, the adversary can perform the rectangle attack. In many block-ciphers in practice, the key size is bigger than or equal to the block-size, and thus the claimed security is at least b bits. Hence, the rectangle attack is a valid attack. However, in Minalpher, the claimed security is 128 bits while the permutation size is 256 bits. The rectangle attack principally requires the data complexity which is more than $2^{b/2}$. Therefore, the rectangle attack is impossible for Minalpher.

3.3.5 Integral Attack

Integral attack was firstly proposed by Daemen *et al.* [15] and was later unified as integral attack by Knudsen and Wagner [25]. It first constructs an integral distinguisher. In more details, an attacker prepares a set of chosen plaintexts such that several bits take all possibilities among the texts and the other bits are constant for all texts. For these plaintexts, the corresponding states after a few encryption rounds have a certain property, *e.g.* the XOR sum of all states in the set is 0 with a probability 1, which is often called a balanced property. Based on the integral distinguisher, a partial tweak recovery attack on the AEAD mode or a forgery attack on the MAC can be constructed.

3.3.5.1 Universal Forgery Attack on 4.5 Rounds

The attack basically exploits the low algebraic degrees of 4.5-round Minalpher. The idea of the forgery attack exploiting the low algebraic degrees was shown in [1]. In Fig. 39, we describe the 5-round integral distinguisher. (Only the first 5 rounds out of 6 rounds are used for the universal forgery.) 'A' represents that all the 16 elements in the nibble appear exactly the same number, *i.e.*, if the data complexity is D, each element appears exactly $D/16$ times. 'B' represents the balanced nibbles defined in the above. The blank nibbles represents that the nibble takes only a fixed value for the entire set of texts. The initial state contains 23 'A' nibbles. Therefore, by querying $2^{23\times4} = 2^{92}$ messages whose 'A' nibbles take all possibilities and other nibbles are fixed to some constant, the XOR of all the corresponding 2^{92} MAC values becomes 0 in all nibbles.

To achieve the universal forgery attack, let M be the target message to be forged. The attacker generates $2^{92}-1$ new messages by computing $M_i \leftarrow M \oplus \Delta_i$ for $\Delta_i = 1, 2, \ldots, 2^{92}-1$, where Δ_i is the difference for the 23 'A' nibbles in the initial state. The target M can be represented by M_0 because $M = M \oplus 0$. The attacker then queries M_i for $i = 1, 2, \ldots, 2^{92}-1$ to the MAC oracle MAC$(\cdot)$, and computes the XOR of the corresponding MAC values. Let σ be the computed XOR sum of the $2^{92}-1$ MAC values, *i.e.*,

$$\sigma \leftarrow \bigoplus_{i=1}^{2^{92-1}} \mathrm{MAC}(M_i).$$

From the 4.5-round integral distinguisher, we know that

$$\bigoplus_{i=0}^{2^{92-1}} \mathrm{MAC}(M_i) = \mathrm{MAC}(M_0) \oplus \bigoplus_{i=1}^{2^{92-1}} \mathrm{MAC}(M_i) = 0.$$

Hence, MAC$(M_0) = \sigma$. The 4.5-round universal forgery attack requires 2^{92} queries, 2^{92} XOR computations and negligible memory.

3.3.5.2 Partial Tweak Recovery Attack on 8.5 Rounds

For the tweak recovery attack, the condition of the output of the distinguisher can be loosed, *i.e.* it is usually enough to have only 1 balanced nibble. Then, the integral distinguisher can be extended to 6 rounds, which is also shown in Fig. 39. Here, gray nibbles represent that no structural property exists in the nibble.

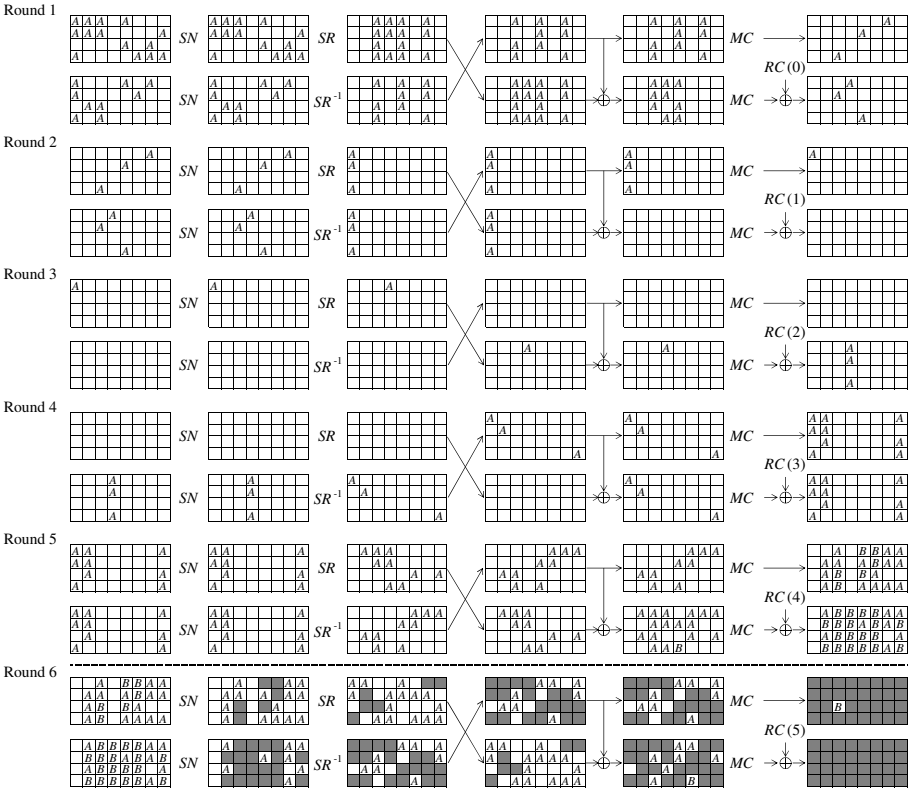


Fig. 39. Integral Distinguisher for 5 Rounds and 6 Rounds

We append 2.5 rounds after the 6-round integral distinguisher, which is shown in Fig. 40. The nibbles with crossed lines are the ones that relate to the 2.5-round partial decryption from the ciphertext to the output of the distinguisher. Note that the attack requires oracle accesses under the same pair of the key and the nonce 2^{92} times. Hence, the nonce-misuse setting for the encryption or unverified plaintext release for the decryption. A 256-bit tweak value which is secret to the attacker is XORed to the output of the 8.5-round Minalpher-P, and then output as a ciphertext.

As shown in Fig. 40, 23 nibbles (92 bits) are related to the partial decryption up to the balanced nibble in the end of the distinguisher. Hence, the attack starts from storing 2^{92} 92-bit values of the ciphertext. From the structure of the AEAD mode, it is obvious that 92 bits of the tweak value is related to the partial decryption. With a straight-forward method, the partial decryption requires a complexity of $2^{92+92} = 2^{184}$. However, several techniques are known to improve the attack complexity.

The partial decryption can be described by $F(C \oplus K)$, where C is a ciphertext, K is a secret-key and F is a public function. In [50], Todo showed that the XOR-sum of the function output in this form can be computed efficiently by using the Fast Fourier Transform, which was originally observed by Collard *et al.* to improve the complexity of the linear cryptanalysis [14]. Let the size of the related ciphertext bits be N, and the size of the key bits be N. Also let the size of the balanced word be b bits. Then, the complexity is roughly evaluated as $N \times 2^N \times b$ [50]. In our case, $N = 92$ and $b = 4$. Thus the complexity of the 8.5-round partial tweak recovery attack is $92 \times 2^{92} \times 4 = 2^{100.5}$. The attack also requires 2^{92} chosen plaintexts and a memory to store 2^{92} ciphertexts.

3.3.6 Impossible Differential Attack

As well as other permutations with an SPN structure, a relatively long impossible differential characteristic can be constructed from 1-active nibble state to 1-active nibble state. A 6.5-round impossible differential characteristic, which is one of the longest ones discovered in our experiment, is shown in Fig. 41. Grey nibbles have non-zero difference with probability 1, empty nibbles do not have difference with probability 1, and nibbles with '*?*' do not have any property with probability 1.

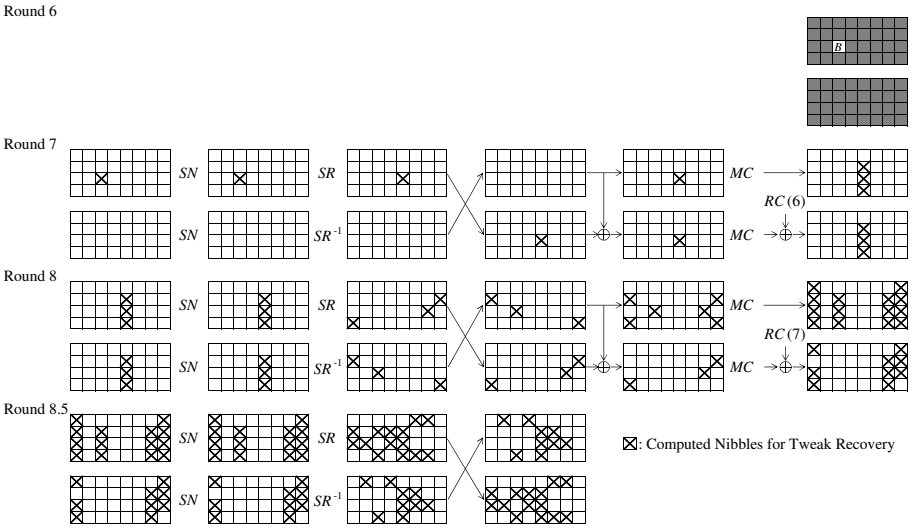


Fig. 40. 8.5-round Integral Attack for Tweak Recovery

However, we currently do not have any idea to exploit the above impossible differential characteristic with a complexity below 2^{128}. For example, finding a pair with 1-active nibble input difference and 1-active nibble output difference requires 2^{249} queries. We show the impossible differential characteristic only for the future research progress.

3.3.7 Evaluation of Algebraic Degrees

Algebraic degrees are useful for evaluating the nonlinearity of functions. The algebraic degree of the round function of Minalpher-P is 3 because S-box is the only non-linear component and the algebraic degree of each S-box is 3. By the trivial evaluation, the algebraic degree of r-round Minalpher-P is at most 3^r. Boura *et al.* proposed how to evaluate a tighter algebraic degree [11].

Theorem 7 ([11]). *Let F be a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^n$ corresponding to the concatenation of m smaller S-boxes, $S_1, S_2, \ldots, S_m$, defined over $\mathbb{F}_2^{n_0}$. Then, for any function G from $\mathbb{F}_2^n$ into $\mathbb{F}_2^\ell$, we have*

$$\deg(G \circ F) \leq n - \frac{n - \deg(G)}{n_0 - 1}.$$

By using the trivial evaluation and this theorem, the upper bound of the algebraic degree of the output of r rounds is computed as follows:

Number of Rounds	1	2	3	4	5	6	7	8	9
Bound on degree	3	9	27	81	197	236	249	253	255

By using 2^{82} chosen plaintexts, we can construct a 4-round integral distinguisher. However, the distinguisher is not efficient than the one described in Subsection 3.3.5.

3.3.8 Meet-in-the-Middle Attack

A meet-in-the-middle (MitM) attack [12, 16] divides the target function F into two subsequent independent subfunctions $F_1 \circ F_0$. Because the analysis of F_0 and F_1 are done independently, the attack cost is the sum of two analyses instead of their product. The results of the independent analyses will match in the middle of the two subfunctions F_0 and F_1. If the independently simulated results are consistent, the results can match in the middle of F_0 and F_1 with probability 1. Otherwise, they match only probabilistically. When the function size is n bits, the MitM attack principally requires the attack complexity of at least $2^{n/2}$.

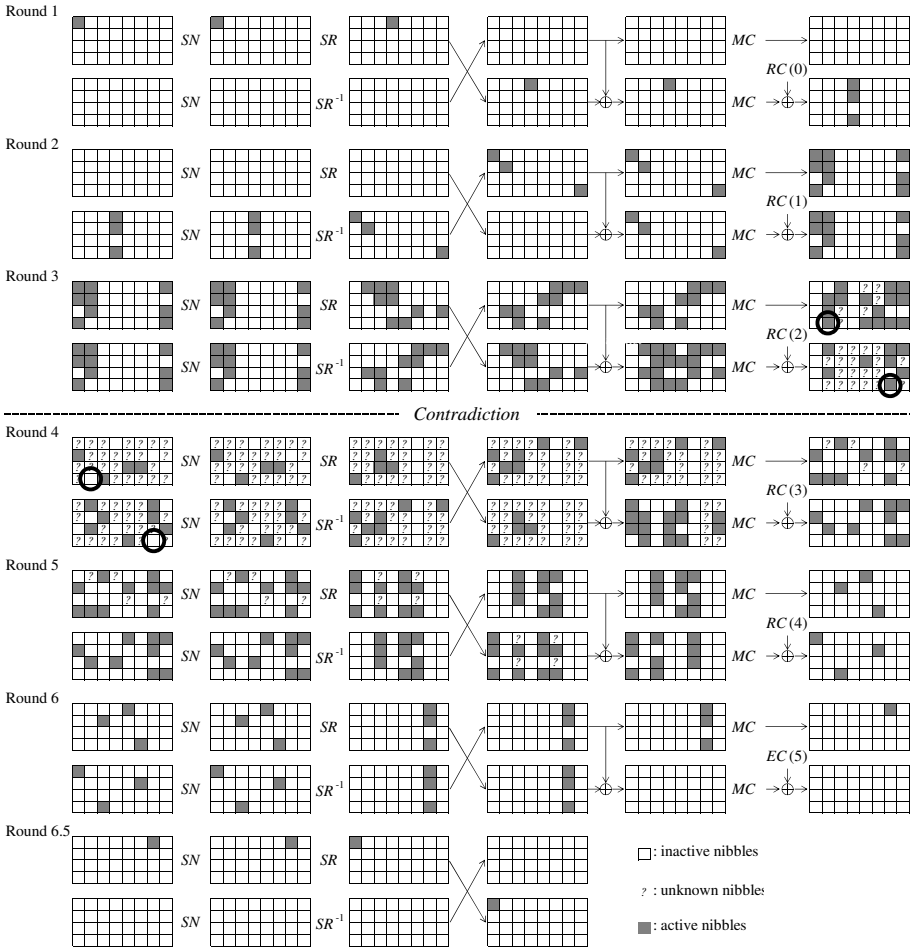


Fig. 41. 6.5-round Impossible Differential Characteristic for Minalpher-P

The designers have not found an efficient method to mount the MitM attack on Minalpher-P. The primary difficulty is that the permutation size, 256 bits, is a double of the number of bits of the claimed security, 128 bits. By following the principle, the complexity of the MitM attack is at least $2^{256/2} = 2^{128}$, and it cannot be a valid attack.

3.3.9 Related-key Cryptanalysis

A generic related-key attack is known for any block-cipher with a k-bit key and a relatively big block size. Let $\#R$ be the number of distinct key oracles to which the adversary can access, and the key relations are known to the adversary. Let T be the time complexity. The adversary can recover all $\#R$ keys with a trade-off $T \cdot \#R = 2^k$.

For simplicity, this section describes the attack on the AEAD mode in the nonce misuse setting.

1. Fix the message to a randomly chosen value, say 0.
2. The adversary queries the fixed message to $\#R$ distinct related-key oracles. The obtained ciphertexts are stored in a list L_1.
3. The adversary guesses T distinct key values and computes the ciphertexts of the fixed message. The results are stored in a list L_2.
4. A match between L_1 and L_2 suggests the key value with a good probability. Note that false positives can be eliminated with two additional queries.

3.3.10 Distinguisher on Minalpher-P with Rebound Attack

The rebound attack presented by Mendel *et al.* [36] is a strong distinguishing attack for key-less primitives especially for AES-like hash functions and permutations. Then the SuperSbox technique was later combined to improve the attack [19, 31]. Jean *et al.* further extended the attack by presenting a method called improved rebound attack [22].

The rebound attack is well-applied to the substitution-permutation network (SPN), in which the computation in one round consists of a non-linear layer (NL) and a linear layer (L) and key addition for block-ciphers. In short, the rebound attack is a technique to efficiently find a pair of values satisfying the differential transition over L–NL–L. Combination with the SuperSbox technique can efficiently satisfy the differential transition over L–NL–L–NL–L, and the improved rebound attack can efficiently satisfy the differential transition over L–NL–L–NL–L–NL–L. Minalpher also adopts the SPN structure, and the improved rebound attack can be applied.

3.3.10.1 Attack Scenario (Distinguisher for the Internal Permutation)

The rebound attack aims to detect a non-ideal behavior of the target key-less primitive from the ideally designed one. Hence, the analysis can be applied only for the internal permutation. In this section, we never consider the impact of the key K, nonce N and tweak values. Note that security of Minalpher is proven by assuming that the tweakable Even-Mansour construction is a 2^{128} tweakable pseudorandom permutation. Hence, distinguishing a public permutation Minalpher-P from a random permutation with complexity less than 2^{128} do not give any impact to the security of Minalpher.

The goal of the adversary is to detect an input difference Δ_{IN} and the output difference Δ_{OUT} that can be satisfied in Minalpher faster than the cost that are necessary for a random permutation.

Note that some trivial distinguisher can be constructed easily such that the adversary chooses two random inputs I_0 and I_1, computes the corresponding outputs O_0 and O_1, and declares that $\Delta_{IN} = I_0 \oplus I_1$ and $\Delta_{OUT} = O_0 \oplus O_1$ can be satisfied at the same time with two computations, while satisfying such a property requires 2^{256} queries for a random permutation. This kind of distinguisher is obviously invalid because a pair of input and output differences that can be satisfied easily can also be constructed to a ideally designed random permutation only with two queries. We regard that the attack presented in this section is valid because it can work even if the distinguisher is challenged with several factors *i.e.* constant values or S-box specifications are randomly determined after the attack is described.

3.3.10.2 Truncated Differential Characteristic

We show that 7.5 rounds of Minalpher-P can be distinguished from a random permutation with less than 2^{128} complexity. The truncated differential characteristic is described in Fig. 42.

The attack divides the characteristic into the inbound part and the outbound part. The borders between the inbound and outbound are drawn by dotted lines in Fig. 42. Namely, from state (A_2^{SN}, B_2^{SN}) to (A_6, B_6) is covered by the inbound part, which includes the differential transition over L–NL–L–NL–L–NL–L. The attacker's first goal is finding a pair of values satisfying the inbound part as fast as possible. The obtained pair is then tested if they also satisfy the truncated differential characteristic for the backward outbound part and the forward outbound part. The probability in the left-hand side in Fig. 42 shows the probability to satisfy the characteristic for each round. Note that probabilistic events are only caused by the linear operations, XM and MC.

3.3.10.3 Inbound Phase

The detailed analysis of the inbound phase is shown in Fig. 43. We first fix the input difference to the inbound phase $(\Delta A_2^{SN}, \Delta B_2^{SN})$ and the output difference from the inbound phase $(\Delta A_6, \Delta B_6)$. Then, the corresponding differences $(\Delta A_3, \Delta B_3)$ and $(\Delta A_5^{SN}, \Delta B_5^{SN})$ can be computed uniquely because all the related operations are linear.

Next, we proceed the computation by using the SuperSbox technique. The SuperSbox for Minalpher-P is somehow different from the one for AES-like ciphers. Due to the XM operation, the attacker needs to guess the values of the same column in two half states A and B. Therefore, the size of each SuperSbox for Minalpher-P is 8 nibbles, or 32 bits. In the forward computation, we guess the values of 8-nibbles (2^{32} possibilities) in (A_3, B_3) which will be located in the same column after the SR and SR^{-1} operations. With those 8-nibble values, we can compute the corresponding 8-nibble values at (A_4^{SM}, B_4^{SM}). The results are stored as the forward SuperSbox for that column. As a result, with $8 \times 2^{32} = 2^{35}$ computations and

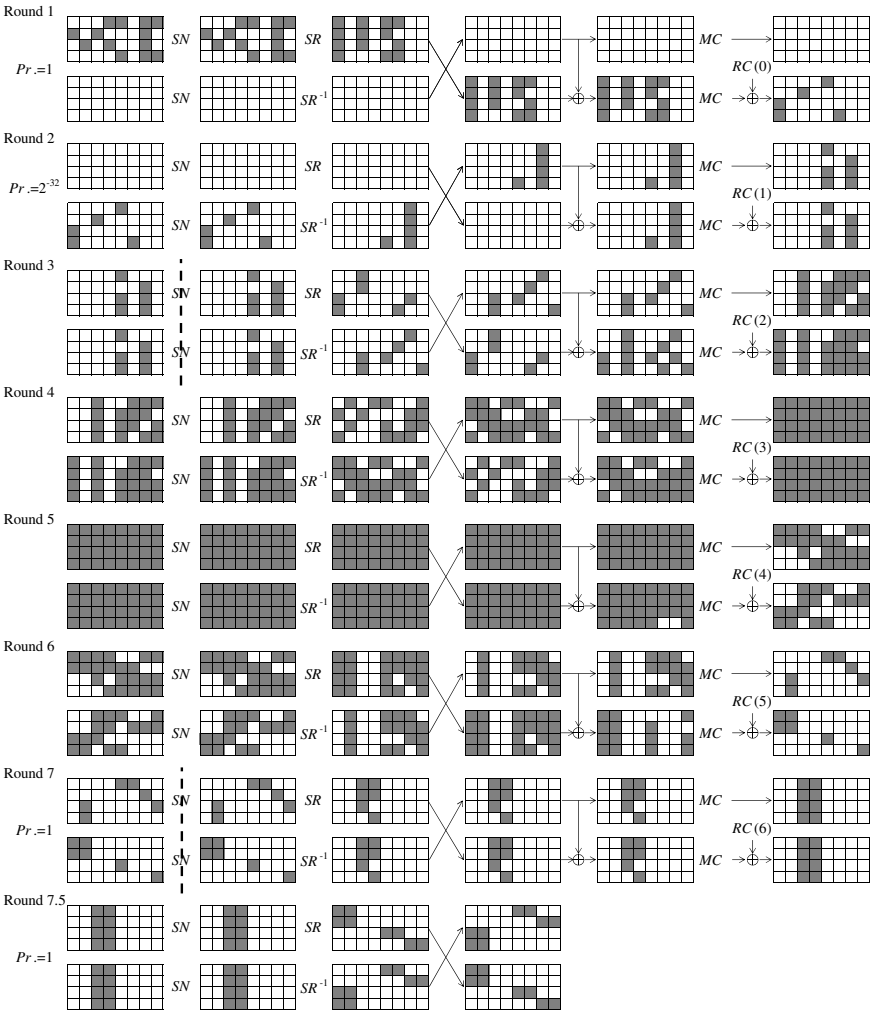


Fig. 42. 7.5-round Truncated Differential Characteristic for Rebound Attack

$8 \times 2^{32} = 2^{35}$ memory, 8 forward SuperSboxes with 2^{32} entries are generated. In Fig. 43, the computation for the forward SuperSbox for column 0 is stressed by bold line with a character F. Similarly, we prepare 8 backward SuperSboxes with $8 \times 2^{32} = 2^{35}$ computations and $8 \times 2^{32} = 2^{35}$ memory. We omit the details. In Fig. 43, the computation for the backward SuperSbox for column 0 is stressed by bold line with a character B.

SuperSboxes from two directions overlap at state (A_4^{SM}, B_4^{SM}). The goal of the attacker is to find a value for each SuperSbox that are consistent each other. Note that, for each fixed difference $(\Delta A_2^{SN}, \Delta B_2^{SN})$ and $(\Delta A_6, \Delta B_6)$, the number of values satisfying the inbound characteristic is 1. On one hand, for the input state, we have 2^{256} choices of paired values. On the one hand, for the output state, two texts in the pair must satisfy the 256-bit relation $(\Delta A_6, \Delta B_6)$.

We search for the solution by using the guess-and-determine procedure. The algorithm is summarized in [47]. However, due to the unique structure of Minalpher-P, several optimizations are necessary to apply the improved rebound attack. In the improved rebound attack, the attacker first prepares the intersection table to check which forward SuperSboxes and backward SuperSboxes overlap each other. As shown in Fig. 43, the forward SuperSbox for column 0 and the backward SuperSbox for column 0 do not overlap. It is obvious from Fig. 43 that the forward SuperSbox for column 0 overlap with the backward SuperSboxes for columns 2, 3, 4, and 6. Moreover, they overlap in 2 nibbles. In the intersection table, the intersection of two overlapping SuperSboxes has a blank cell. The intersection of two unrelated SuperSboxes are filled with black. The generated intersection table is given in Fig. 44. The backward SuperSboxes cannot take all possible differences due to the fixed active nibble positions in the characteristic. For example, column 0

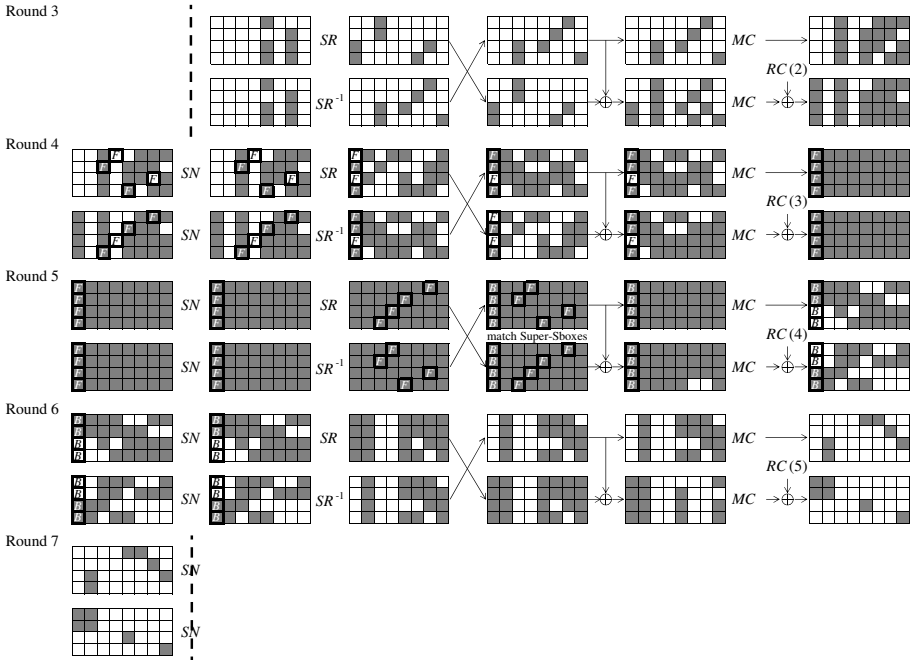


Fig. 43. Detailed Analysis of the Inbound Phase

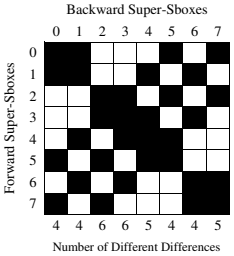


Fig. 44. Intersection Table

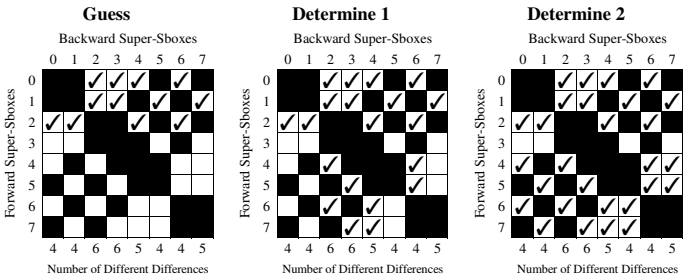


Fig. 45. Guess and Determine Procedure for Improved Rebound Attack

starts with 4-nibble differences, thus it takes only 2^{16} differences. This information is listed in the bottom of Fig 44 as a number of different differences.

Finally, we start to fix the values of SuperSboxes. The attack procedure is as follows, which is also depicted in Fig. 45.

Guess. Fix paired values for forward SuperSboxes for columns 0, 1, and 2 from the 2^{96} possible choices in three columns. This fixes the value and difference for those columns. '$\surd$' denotes that both the value and difference of that nibble are fixed. The fixed value and difference will be constrains to the other SuperSboxes.

Determine 1. Remember that each intersection of SuperSboxes denotes the overlap in 2 nibbles. Backward SuperSboxes for columns 2, 3, and 4 already have 4-nibble constraints in the value and 4-nibble constraints in the difference. Therefore, only 1 solution is expected for these three SuperSboxes. Those columns are fixed with respect to value and difference in all nibbles.

Determine 2. Then, forward SuperSboxes for columns 4, 5, 6, and 7 obtain 4-nibble constraints in both of the value and difference. Hence, those columns are fixed with respect to value and difference in all nibbles.

Determine 3. Similarly, the consistency of all SuperSboxes can be checked, and finally the attack obtains a solution of the inbound phase.

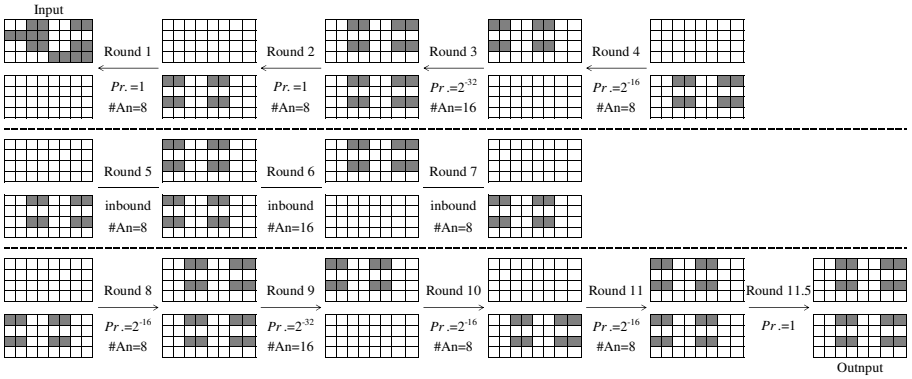


Fig. 46. Summary of the Entire Differential Characteristic.

The above procedure must be repeated for all the 2^{96} possible choices in the guess phase. Thus, the complexity to obtain the solution of the inbound phase is $2^{96} \times 4/7.5 \approx 2^{95}$ 7.5-round Minalpher-P computations.

Note that the analysis of the inbound phase can be iterated by changing the 12-nibble starting difference $(\Delta A_2^{SN}, \Delta B_2^{SN})$ and $(\Delta A_6, \Delta B_6)$. Hence, 2^{48} solutions can be generated in maximum. However, because the attacker is only allowed to use up to 2^{128} computation power, generating all 2^{48} solutions is impossible. In this attack, we generate 2^{32} solutions with spending $2^{95+32} = 2^{127}$ complexity.

3.3.10.4 Outbound Phase and Comparison with a Random Permutation

With the generated 2^{32} solutions of the inbound phase, we find the one that also satisfy the outbound differential characteristic. The backward computation for round 2 is the only probabilistic transition. The condition to satisfy this transition is

$$\Delta A_1^{MC}[0][4] = \Delta A_1^{MC}[2][4] = \Delta A_1^{MC}[3][4] = \Delta B_1^{MC}[0][4] = \Delta B_1^{MC}[2][4] = \Delta B_1^{MC}[3][4]$$
$$\Delta A_1^{MC}[1][6] = \Delta B_1^{MC}[1][6],\ \Delta A_1^{MC}[2][6] = \Delta B_1^{MC}[2][6],\ \Delta A_1^{MC}[3][6] = \Delta B_1^{MC}[3][6].$$

The total probability for round 2 is 2^{-32}, and thus one of the obtained solutions will satisfy those conditions. As a result, the attacker will obtain one solution to satisfy all the characteristic.

The characteristic in Fig. 42 has 13 active nibbles in the input and 16 active nibbles in the output. According to Iwamoto *et al.* [21], the minimum number of queries to satisfy such input and output differential forms with access to a random function is $2^{256-(4\times13)-(4\times16)+1} = 2^{141}$. Because our attack finds such a pair with 2^{127} computations, the attack successfully distinguishes 7.5-round Minalpher-P from a random permutation.

We stress again that distinguishing Minalpher-P from a random permutation with complexity less than 2^{128} do not give any impact to the security of Minalpher.

3.3.11 Distinguisher on Minalpher-P with Rebound Attack and Iterative Characteristic

We present another type of rebound attack against 11.5 rounds of Minalpher-P. The approach is based on the one by Matusiewicz *et al.* [34], which chooses the internal state value round by round to satisfy as much differential propagation through S-box as possible.

For the middle 9 rounds (from round 3 to round 11), we use the 6-round iterative differential characteristic shown in Fig. 36. We then extend it by two rounds in backward, and by a half round in forward. For the middle 9 rounds, we consider a particular difference, which propagates with probability 2^{-2} for each S-box. For the extended first two rounds, and the last half round, we consider the truncated difference, *i.e.* we only care if each nibble is active or inactive. The summary of the differential characteristic is given in Fig. 46. '#An' represents the number of active nibbles in that round.

The middle 3 rounds (rounds 5, 6, and 7) are the inbound part. We generate 2^{128} solutions of the inbound part with a complexity of 2^{126} 11.5-round Minalpher-P computations. Then, each solution is tested if it also satisfies the outbound part.

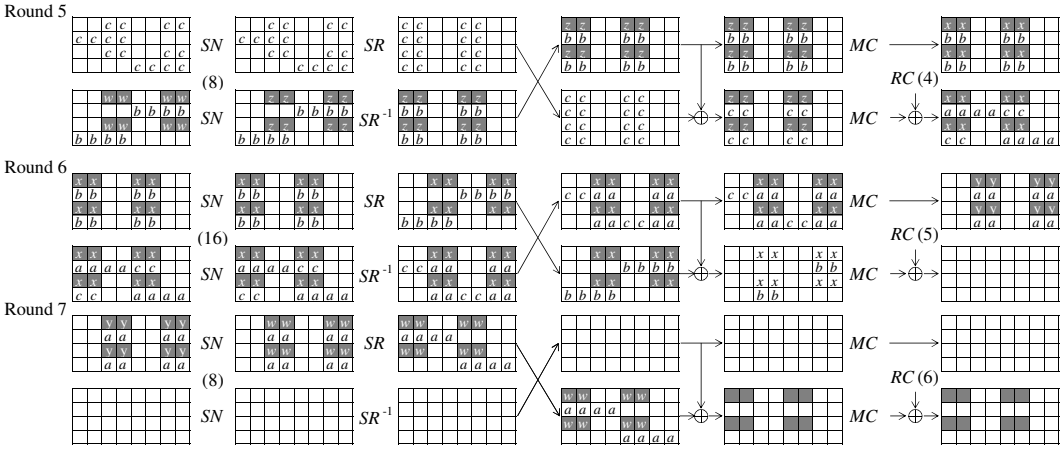


Fig. 47. Detailed Analysis of Inbound Phase.

3.3.11.1 Inbound Phase

We explain the details of the 3 inbound rounds in Fig. 47. The attack procedure is as follows.

1. Choose the active nibble difference at the beginning of round 6 ($\Delta A_5, \Delta B_5$) and at the state after the SN operation ($\Delta A_5^{SN}, \Delta B_5^{SN}$), so that the S-box transition can have solutions. Set the value of each nibble to the solution. Here, all the active nibbles are set to the same value and the same difference. With this operation, the nibbles with 'x' in Fig. 47 are fixed with respect to both the value and difference.
2. From the property of the linear operation, we can compute the difference of several other nibbles though the value cannot be fixed yet. The difference of the nibbles with 'y' and 'z' in Fig. 47 are fixed. Note that the difference of the x nibbles, y nibbles, and z nibbles are identical. However, the value is not fixed for y and z, and they can have different values.
3. Choose the active nibble difference at the state B_4 and A_6^{SN} denoted by 'w' so that the S-box transition can have solutions. Set the value of each nibble to the solution. Here, all the active nibbles are set to the same value and the same difference. With this operation, all the nibbles with y, z, and w are fixed with respect to both the value and difference but for 8 nibbles with z in B_4^{XM}. So far, the difference has been consistent in all nibbles.
4. We then connect the values of x in A_5^{XM} and y in A_5^{MC}. We first pay our attention to the computation in column 2. $A_5^{MC}[0][2]$ and $A_5^{MC}[2][2]$ are computed as follows.

$$A_5^{MC}[0][2] = A_5^{XM}[0][2] \oplus A_5^{XM}[1][2] \oplus A_5^{XM}[3][2],$$
$$A_5^{MC}[2][2] = A_5^{XM}[1][2] \oplus A_5^{XM}[2][2] \oplus A_5^{XM}[3][2].$$

   Because we set that $A_5^{MC}[0][2] = A_5^{MC}[2][2]$ and $A_5^{XM}[0][2] = A_5^{XM}[2][2]$, unfixed values of $A_5^{XM}[1][2]$ and $A_5^{XM}[3][2]$ can be chosen to any of the one that satisfies $A_5^{MC}[0][2] \oplus A_5^{XM}[0][2] = A_5^{XM}[1][2] \oplus A_5^{XM}[3][2]$. Similarly, the other active columns can be fixed. With this operation, the values of the nibbles with 'a' are fixed, and all the fixed values for round 6 and 7 become consistent.
5. Similarly, we connect the values of z in A_4^{XM} and x in A_4^{MC}. The detailed explanation is omitted. With this operation, the values of the nibbles with 'b' are fixed, and all the fixed values of z and x become consistent but for 8 nibbles in B_4^{XM}.
6. Finally, we choose 4-nibble values at B_4^{MC} denoted by 'c', and then compute back to B_4^{XM}. For any choice of 4-nibble values at B_4^{MC}, we obtain the valid value for B_4^{XM}. With this operation, the values of the nibbles with 'c' are fixed and all the differential transitions in the three inbound rounds become consistent. Note that by trying all choices of 4-nibble values at B_4^{MC}, up to 2^{16} solutions can be generated with 2^{16} operations.
7. There are still 28 unfixed nibbles (112 unfixed bits) at state (A_4^{MC}, B_4^{MC}), and for any value for these nibbles, one solution of the three inbound rounds is obtained. Together with 2^{16} solutions in the previous step, up to $2^{16} \times 2^{112} = 2^{128}$ solutions can be generated with a complexity of $2^{128} \times 3/11.5 \approx 2^{126}$ 11.5-round Minalpher-P computations.

3.3.11.2 Outbound Phase and Comparison with a Random Permutation

With the generated 2^{128} solutions of the inbound phase, we find the one that also satisfies the outbound differential characteristic. As long as all S-boxes follow the differential transition with probability 2^{-2}, the iterative characteristic continues. The probability to satisfy each round is given in Fig. 46. With 2^{128} solutions, we can satisfy 2-round backward computation and 4-round forward computation. For the remaining part, we only check if each nibble is active or inactive, thus the propagation is deterministic. In the end, we obtain 1 pair that has only 16 active nibbles in the input and 16 active nibbles in the output.

The complexity of the outbound phase can be smaller than 2^{128} 11.5-round Minalpher-P computations by using the early aborting technique. Let the outbound phase start from checking round 4. If a candidate does not satisfy the differential transition in round 4, we immediately stop the check procedure. Then, only $2^{128-16} = 2^{112}$ candidates will be tested for the other rounds. The complexity of the outbound phase is $2^{128} \times 1/11.5$ 11.5-round Minalpher-P computations, which is smaller than the inbound phase.

According to Iwamoto *et al.* [21], the minimum number of queries to satisfy such input and output differential forms with access to a random function is $2^{256-(4\times16)-(4\times16)+1} = 2^{129}$. Because our attack finds such a pair with 2^{126} computations, the attack successfully distinguishes 11.5-round Minalpher-P from a random permutation.

We stress again that distinguishing Minalpher-P from a random permutation with complexity less than 2^{128} do not give any impact to the security of Minalpher.

3.3.12 Known-key Distinguisher/Chosen-key Distinguisher

A known-key distinguisher was originally proposed by Knudsen and Rijmen against AES and block-ciphers with Feistel network [24]. The goal is finding non-ideal properties of a randomly instantiated block-cipher's permutation from a random permutation. In the known-key distinguisher, the adversary analyzes the target block-cipher under the assumption that the key value will be known to the adversary but the adversary needs to detect non-ideal properties without knowing the exact key value. The known-key analysis is often useful to understand the security of a block-cipher when it is used as a built-in primitive of a hash function. In the chosen-key distinguisher [8, 43], the key input is controlled by the adversary. The goal is detecting a particular key input and non-ideal behaviors of the instantiated block-cipher's permutation. It is notable that no one succeeds in formalizing the notions of the known-key and chosen-key distinguishers so far.

The known-key and the chosen-key distinguishers are security notions for block-ciphers, while the core part of Minalpher is the Even-Mansour construction that consists of a public permutation and two key additions. Hence, applying those distinguishers to Minalpher does not make sense.

Indeed, constructing a known-key distinguisher is trivial for Minalpher. Let x and y be two input values to the internal permutation P. The adversary computes their output values $P(x)$ and $P(y)$ and claims that finding two input values with the input difference $x \oplus y$ and the output difference $P(x) \oplus P(y)$ is trivial for any key for Minalpher. Actually, after the key K is randomly given to the adversary, two message $x \oplus K$ and $y \oplus K$ satisfy the claimed property, while detecting such a pair for a random permutation requires much more complexity than two queries.

In summary, the designers claim that Minalpher does not provide any protection against known-key and chosen-key distinguishers. In other words, any known-key and chosen-key distinguishers never impact to the security of Minalpher. The designers also never recommend building a cryptographic hash function by applying any mode of operation for block-ciphers to Minalpher.

4 Features

4.1 Advantages of Minalpher

Minalpher is designed so that it can be easily used in practice.

The first goal is providing 128-bit security for both of confidentiality and integrity. This is a significant advantage compared to many of AES based authenticated encryption schemes including AES-GCM [35,

42] and AES-OCB [26], where the integrity is only proven to be secure up to a half of the block size of the underlying block-cipher, *i.e.*, only up to 64 bits.[3]

The second goal is reducing the security risk when Minalpher is used with some unintentional implementation. For this purpose, Minalpher provides some level of the robustness against nonce misuse and unverified plaintext release.

The third goal is making Minalpher to be efficiently implemented in various platforms, especially in embedded systems. The 8-bit S-box adopted in AES is not suitable for embedded systems due to its large memory requirement. Minalpher is designed to use a 4-bit S-box. Moreover, the internal permutation is involution so that the encryption circuit and decryption circuit are identical. These properties make Minalpher suitable for embedded systems. When sufficient resource is available, the fully parallelizable computation structure makes the computational speed very fast.

The fourth goal is providing a various optional functionalities such as the MAC mode, associate data reuse, and incremental computation.

Finally, the fifth goal is a uniqueness of our design. Minalpher is designed based on the tweakable Even-Mansour construction with an involution permutation. This makes Minalpher different from other block-cipher based designs and sponge based designs. In this section, we explain each property of Minalpher in details.

4.1.1 Proved Security against Nonce Misuse

Minalpher is designed to provide some level of the robustness against the nonce misuse *i.e.*, it provides 128-bit security for integrity and confidentiality against nonce misuse.

The security claim for integrity is simple. The tag generation function of the AEAD mode is proven to be a PRF up to $\mathcal{O}(2^{n/2})$ queries in the nonce misuse setting. The tag generation function of the MAC mode, which does not use a nonce, is also proven to be a PRF up to $\mathcal{O}(2^{n/2})$ queries.

The security claim for confidentiality is a little bit complicated. The confidentiality of current authenticated encryption schemes against nonce misuse can be classified into the following three classes.

Full PRF: A cipher is a PRF even if the same nonce is reused for the same key. The full PRF cipher is called "offline cipher".

Suffix PRF: A cipher is a PRF after a different message block, *i.e.*, for two distinct messages sharing the same prefix, $M = M_{\text{fix}} \| M$ and $M' = M_{\text{fix}} \| M'$, the ciphertexts of the prefix parts become identical but those of the suffix parts are independent. The suffix PRF cipher is called "online cipher".

Insecure: The message is recovered from the ciphertext.

The first two notions are usually regarded to be nonce misuse resistant in the crypto community.

Here, we would like to introduce a new class of misuse resistant security called block-wise PRF which leaks information of "positions" more than a suffix PRF but leaks no other information such as message values.

Block-wise PRF: PRF is preserved for all message blocks unless exactly the same message value has been queried before at exactly the same block position.

The encryption function of the AEAD mode is proven to be a block-wise PRF up to $\mathcal{O}(2^{n/2})$ queries in the nonce misuse setting. This is a strong advantage compared to AES-GCM and AES-OCB, which are insecure in the nonce misuse setting.

4.1.2 Proved Security against Unverified Plaintext Release

Most of the current authenticated encryption schemes, during the decryption, do not allow to output the decrypted plaintext until the verification of the tag is confirmed. It indicates that all the message blocks must be stored in a memory to wait for the result of the tag verification. This is very hard or infeasible to implement, especially if the input message length is long. Minalpher takes a ciphertext of the size up to $2^{104} - 1$ bits as input, and thus the designers do not think storing all the message blocks is feasible.

[3] Moreover, Niwa *et al.* recently has shown that the proved security of GCM is much small than a half of the block size, due to a huge constant factor in the security proof [44].

To avoid this self-contradictory situation, Minalpher is designed so that the decryption results can be released block by block before the tag verification is finished with keeping some level of the robustness. The integrity of Minalpher mode of operation is proven to provide $\mathcal{O}(2^{n/2})$ security in the unverified plaintext release setting.

In the crypto community, this property is sometimes called "online decryption" or "ciphertext misuse". However, the designers prefer to call it unverified plaintext release. This is because 1) the ability of processing the ciphertext online is an independent issue of whether the processed plaintext can be released immediately or must be stored until the tag is verified. 2) for a long message, cipher's designers should allow implementations not to store all the message blocks, in other words, the unverified plaintext release should not be regarded as a misuse by implementors.

4.1.3 Fully Parallelizable

The scalability is an important factor for both of the software and hardware implementations. Minalpher is designed to be fully parallelizable to achieve the scalability. By generating tweak values (or an intermediate variable $L' \leftarrow (K\|\mathsf{flag_m}\|0^{n/2-s}) \oplus P(K\|\mathsf{flag_m}\|0^{n/2-s})$ and $L \leftarrow (K\|\mathsf{flag_m}\|N) \oplus P(K\|\mathsf{flag_m}\|N)$, the encryption, decryption, and MAC mode can process each associated-data block and message block in parallel.

4.1.4 MAC Mode

Some application does not require the confidentiality and only requires the integrity. Minalpher provides the MAC mode to meet both of such demand and the demand of authenticated encryption only with one design. At the same time, the computation time is optimized so that it can be faster than simply computing the AEAD mode then discarding ciphertexts. It is notable that the nonce does not need to be implemented if Minalpher is used only for the MAC mode.

4.1.5 Fixed Associated Data Reuse

Some application never changes the associated data value. In such a case, by reusing the result of processed associated data, the computation time can be optimized. We stress that the fixed associated-data reuse is an independent concept for the nonce reuse. Because the nonce is not used to process the associate data, the fixed associated-data reuse is available even if the nonce changes every time.

4.1.6 Incremental Authenticated Encryption/Incremental MAC

Suppose that, for a multi-blocks message $M = M_0\|M_1\|\cdots\|M_{N-1}$, the corresponding ciphertext $C = C_0\|C_1\|\cdots\|C_{N-1}$ and the tag T are computed under a key K and a nonce N. Also suppose that we compute a ciphertext and a tag for a new message M' in which only a small fraction, *e.g.* one block, of M are modified under the same key K and the nonce N. The scheme is called incremental authenticated encryption if new C and T can be computed significantly faster than simply recomputing M'. According to Yasuda [52], several classes of the incremental authenticated encryption exist depending on the assumption when new C and T are computed.

In Minalpher, new C'_i and T' can be efficiently computed from M_i, C_i, M'_i, T by using an additional n-bit memory, where n is the permutation size ($n = 256$ for Minalpher). In details, every time a tag is computed, we store a 256-bit intermediate value just before computing the last P for generating the tag. It is obvious that C'_i can be computed only from M'_i (and K and N). To update the tag, we replace the impact to the stored intermediate value from C_i with the one from C'_i, and then compute the last permutation to obtain the new tag T'.

Incremental authenticated encryption can be performed only when the same pair of (K, N) is reused. Hence, the security decreases up to the nonce misuse level. Note that the nonce reuse may not be a problem depending on the application. For example, for storage, the ciphertext, tag, and nonce may not be sent immediately, and thus the incremental authenticated encryption is useful to modify only a small fraction of the message. Also note that the MAC mode does not require the nonce. Hence, Minalpher is an incremental MAC without any security loss.

4.1.7 Availability in Embedded Systems

Minalpher is designed so that it can be easily implemented even in 4-bit micro-controllers. Compared to the 8-bit S-box used in AES, the 4-bit S-box in Minalpher achieves a much smaller implementation. This is an advantage of Minalpher over AES-based authenticated encryption schemes.

4.1.8 Simplicity with Permutation Design

A permutation is a simpler primitive than a block-cipher. This feature makes the computations of Minalpher easy to understand and easy to analyze. As an example, the related-key analysis does not have to be considered for the tweakable Even-Mansour construction part. It also simplifies the status of Minalpher against the known-key and chosen-key distinguishers.

4.1.9 Involutive Permutation for Small Implementation

Minalpher requires to call P^{-1} for the decryption. Implementing P and P^{-1} independently requires a large amount of memory. To avoid this problem, Minalpher was designed so that the permutation is involutive *i.e.*, P and P^{-1} are almost identical (except for round constants).

4.1.10 Byte-oriented Tweak Generations

Tweaks for each message block is often generated with a multiplication over a finite field or the Gray code. Those operations require a bitwise implementation, and without a careful analysis, those operations give a significantly bad impact to the performance. In Minalpher, the tweak generation is optimized so that all operations are byte-wise, which can be computed efficiently in most of the environment.

4.1.11 Design Uniqueness

Both the mode of operation and the primitive of Minalpher are very different from ones adopted in existing authenticated encryption schemes, such as the OCB mode, the GCM mode, the duplex sponge [3] or AES-based block-cipher/permutation, Keccak-based permutation, stream-cipher based permutation, and so on.

4.2 Performance in Software on High-end CPU

Minalpher can efficiently be performed on high-end CPU using so-called `vperm` instruction implementation. Intel's SSSE3 instruction set includes `pshufb` to realize `vperm`, and similar instructions are available on other architectures. `vperm` instruction can parallelly look-up 4-bit S-boxes. Utilizing `vperm` instruction, we first need to convert the data structure from 2 nibbles to 1 nibble in each byte.

As the example, we explain the operation on Intel x86(_64). Since the bit size of the XMM register is 128-bit, we can store at most 16 nibbles in one XMM register. In Minalpher-P, the state is expressed by two 4×8 matrices $A \in \left\{\{0,1\}^4\right\}^{4\times 8}$ and $B \in \left\{\{0,1\}^4\right\}^{4\times 8}$, namely, each row of two matrices has 8 nibbles. MixColumns is a linear function within each column, and the operation for A is the same so that for B. Then we store both i-th row of A and B in one XMM register. When we have this data structure, the state in Minalpher-P is managed using 4 XMM registers. SubNibbles, ShuffleRows, SwapMatrices and MixColumns are simply performed on 4 XMM registers. Only XorMatrix diffuses the inside of each XMM register, but it can efficiently perform using `shift` and `xor`.

To avoid the performance degradation from XorMatrix, we use 2-block parallel implementation which performs 512 bits in the same time. When we use 2-block parallel implementation, the state in Minalpher-P is managed using 8 XMM registers. Then all operations are simply performed on 8 XMM registers, namely, we do not need to diffuse the inside of each XMM register. When YMM registers whose bit length is 512-bit are available, we implement 1-, 2-, and 4-block parallel implementation of Minalpher-Pusing similar idea as one for XMM registers.

Table 3 shows the performance of our implementations. We can confirm that Minalpher can perform better than several AES-based authenticated encryption modes of operation if AES instructions are not used.

Note that the following conditions are used to get the numbers in Table 3.

Table 3. High-end CPU Software Performance in Cycles per Byte

	Implementation Type	Data Length				
		31B	63B	1KB	8KB	64KB
Intel Core i7-3770 (Ivy Bridge)	1-block	25.21	19.58	14.40	14.05	14.05
Intel Core i7-3770 (Ivy Bridge)	2-block	26.97	17.97	9.63	8.94	8.85
Intel Core i7-4770 (Haswell)	1-block	25.47	19.66	14.00	13.69	13.65
Intel Core i7-4770 (Haswell)	2-block	26.12	16.97	8.80	8.30	8.23
Intel Core i7-4770 (Haswell)	4-block	26.85	17.27	6.33	5.76	5.69

- Agree with SUPERCOP API.
- Only one core is used.
- Speedstep, Turboboost, and Hyperthreading technologies are disabled.
- Choose the best numbers from several trials to avoid lags caused by cache status, interrupts, and other reasons.

4.3 Performance in Software on Low-end Microcontroller

4.3.1 Overview

In embedded environments, reducing memory size often has priority over achieving faster speed. Taking this real-world practice in consideration, Minalpher is carefully designed so that extremely small implementation is possible on low-end microcontrollers, at the same time, without losing possibilities of high speed implementation.

Minalpher contains a single 4-bit S-box as its non-linear component, which contributes to reducing data memory size in low-memory embedded environments, but adjacent two S-boxes can be merged into an 8-bit look-up table, which also gives a programmer an option for creating a fast speed code at the cost of an increase in 240 data bytes.

Minalpher-P is a permutation with the involution property. This means that, unlike block-cipher-based authenticated encryption algorithms, a programmer does not have to struggle with an overhead of key scheduling part, and that a cost for treating its inversion can be negligible. We were able to implement Minalpher-P in only 220 ROM bytes on the RL78 microcontroller.

The overall structure of Minalpher is very simple. It is comprised of a repetition of the Even-Mansour component of the form $\varphi \oplus P(M \oplus \varphi)$. An update of the tweak φ is computed as a multiplication of a polynomial over $GF(2^8)$ with a small constant polynomial, which is much simpler and faster than the general multiplication over $GF(2^{128})$ in AES-GCM, particularly on a low-end microcontroller.

Minalpher has a byte-oriented architecture. We avoided introducing bitwise shift operations in designing Minalpher due to implementation reasons. Efficiency of shift instructions greatly depends on processor hardware. Some microcontrollers do not have a shift instruction with a multiple shift count. Another microcontroller does not support a rotate shift instruction with carry. We did not take a risk that Minalpher might lose performance in a particular type of microcontrollers.

4.3.2 Interface and Metrics

A simple program interface is critically important in embedded software on a low-end microcontroller. When minimizing code size is required, handling a complex interface of a target program can be a significant overhead rather than processing its main logic. Also in practice of embedded systems, it is common that a message must be processed separately. This is not only because its length is often unknown or undetermined when encryption begins, but also because an interrupt that has occurred during encryption process must be handled as quickly as possible.

Keeping a practical and usable code on embedded microcontrollers in mind, we designed our benchmark programs and measured their size and speed on the basis of the following policies, balancing minimalism, usability and security:

1. A code should be described as a subroutine callable from a high-level language.
2. A code should be able to process an intermediate one-block data and return to a caller program.

3. A code should run in constant time, independent of secret information.
4. A code should process message padding.
5. A code should run as an application program, not using system-purpose instructions and system memory area.
6. ROM size count should include instruction code and constant data.
7. RAM size count should include all parameters given by a caller program, internal stack and temporary memory used in the code.

In our benchmark codes, all parameters given by a caller program and internal temporary memory except stack are allocated in consecutive area in a callee program, and its head address is passed to a caller program as an external variable.

4.3.3 Implementation Results

We chose the RL78 microcontroller as the evaluation platform, since it is a typical CISC processor suitable to code size reduction. RL78 has an accumulator-based architecture consisting of eight 8-bit general registers `a,x,b,c,d,e,h,l`, of which many instructions accept only `a` or `ax` as a destination. A limited number of instructions have a 16-bit form with register pairs `ax,bc,de,hl`.

We hence often suffer from 'register starvation' on this microcontroller, which requires extra instructions and memory for saving and restoring data on an accumulator, but a big advantage of this CISC processor is that an average code size is short due to its support of read-modify instructions. For instance, the instruction `xor a,[hl]`, – read from an address pointed by `hl` and xor to `a` –, is a one-cycle instruction with one-byte length. This significantly contributes to code size reduction.

Our codes of Minalpher, which are a subroutine callable from C language, are written in an assembly language and receive fixed-length parameters passed from a caller program. The parameters consist of data to be processed, key, nonce, tag, data length and mode. The mode indicates which part of Minalpher should be processed as follows:

Mode	Function
`[1]`	Initialization (Computation of L and L')
`[2]`	Associate Data Block (except last block)
`[3]`	Associate Data Block (last block)
`[4]`	Encryption Block (except last block)
`[5]`	Encryption Block (last block) and Tag Generation
`[6]`	Decryption Block (except last block)
`[7]`	Decryption Block (last block) and Tag Verification

Table 4 shows the result of our embedded software implementation. In this table, speed of mode 3 contains two numbers, of which the first one shows that the last block is 32-byte long, and the second one indicates otherwise. These codes process message padding inside and run in constant time for any fixed-length associated data and plaintext/ciphertext. The small size code requires only 510-byte ROM, and the fast speed code with less than 1300 ROM bytes runs more than five times faster, at the speed of 514 cycles/byte for long data.

Design Goal	ROM (bytes)	RAM (bytes)	Speed (mode) (cycles)
Small Size	510	214	`[1]` 90,235, `[2]` 45,302, `[3]` 45,563/45,932
			`[4]` 90,992, `[5]` 90,859, `[6]` 91,081, `[7]` 91,017
Fast Speed	1275	470	`[1]` 16,805, `[2]` 8,166, `[3]` 8,459/8,772
			`[4]` 16,447, `[5]` 16,416, `[6]` 16,669, `[7]` 16,637

Table 4. Software implementation on the RL78 microcontroller.

4.4 Performance in Hardware

4.4.1 Overview

Minalpher is designed with emphasize on performance in hardware implementations. It is characterized by the (i) Even-Mansour like construction, (ii) nibble-oriented operations, and (iii) involutive property.

Firstly, key-related circuits can be separated from the one for permutation, thanks to the Even-Mansour like construction. As a result, the Minalpher-P circuit can be small comparable to 128-bit block-ciphers that usually require at least two 128-bit registers for message and key.

Secondly, the nibble-oriented operations of Minalpher-P enables efficient circuits. Notably, the 4-bit S-boxes are advantageous to larger S-boxes in terms of both delay and circuit area [10]. In addition, the nibble-wise permutation (*i.e.*, SR/SR^{-1}) enables an efficient shift-register-based architecture which is advantageous in compact implementations.

Finally, both encryption and decryption are calculated with the same circuit because of the involutive property. The number of additional selectors required for switching between encryption/decryption is very small.

Another important advantage of Minalpher is its high scalability between the speed-area trade-offs. In the latter sections, implementations in three corner cases are shown. They are the (i) high-speed core, (ii) low-area coprocessor, and (iii) mid-range core.

4.4.2 High-speed Core

The high-speed core aims to achieve high throughput. The core works in a stand-alone manner; all the required operations namely encryption/decryption, tweak generation/update, and tag accumulation are involved within the core.

Firstly, a circuit for Minalpher-P is discussed. A natural way to achieve high throughput is to use a 256-bit datapath where 1-round function is implemented as a combinatorial circuit. Fig. 48 shows a datapath diagram of such a circuit. In the circuit, the 1-round function is calculated in 1 cycle, thus a single 256-bit input is permuted with 18 cycles.

The high-speed core is built on top of the above 256-bit Minalpher-P circuit. The core provides command-based interface. Both the AEAD and MAC modes are implemented by combining the commands. Some registers are added for intermediate values. They are registers to store (i) 128-bit key (K), (ii) 256-bit tweak (L), and (iii) 256-bit tag (T). Selectors and XOR gates are added to support appropriate data flow between the registers.

It is worth noting that higher throughput can be achieved by pipelining the Minalpher-P circuit. With pipelining, more than two data blocks can be permuted in parallel. Thanks to the parallelizable property of Minalpher, the pipeline can be saturated with data (*i.e.*, operated without bubbles). For example, 2-stage pipelining is achieved by inserting a register in the middle of the datapath. Ideally, the throughput is doubled at the cost of the 256-bit additional register. However, a concrete evaluation of such an architecture is not covered in this article.

4.4.3 Low-area Coprocessor

In a chip with restricted resources, it is common to implement only a fraction of the whole algorithm as a coprocessor. Remaining parts of the algorithm are calculated by an external CPU. The low-area coprocessor is designed with the above situation in mind.

Since the most time consuming part in Minalpher is Minalpher-P, it is natural to implement it as a coprocessor. A compact implementation of Minalpher-P is discussed. We can employ various datapath widths thanks to the regular structure of Minalpher-P. A good candidate for a compact implementation is the 16-bit datapath. Fig. 49 shows the 16-bit datapath where four S-boxes and one matrix multiplication are implemented as combinatorial logic. For an efficient data management, the order of operations are changed from the original specification. Nibble-wise shuffling in SR/SR^{-1} can be implemented with special shift registers labeled "Data manager". With the data manager, the nibble-wise shuffling SR/SR^{-1} are implemented with 108 2-input selectors, that is advantageous to a straight-forward implementation with at least 256 2-input selectors. In the 16-bit Minalpher-P circuit, a 256-bit input data is permuted with $288 = 18 \times 16$ cycles (*i.e.*, 1-round/16-cycle) at maximum.

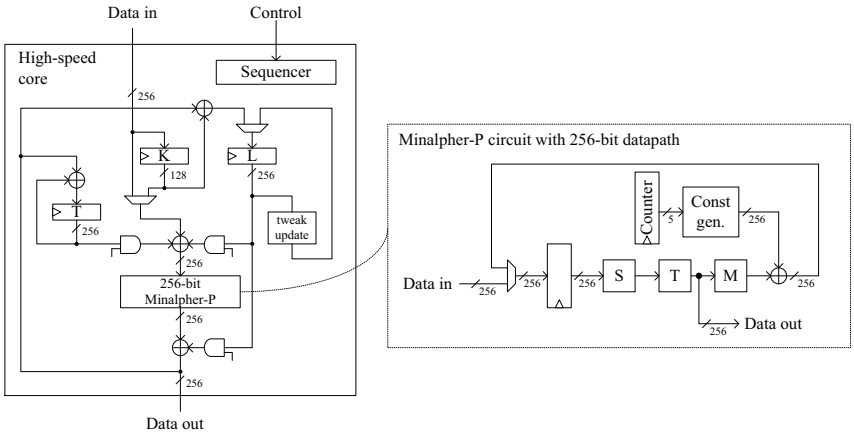


Fig. 48. High-speed core

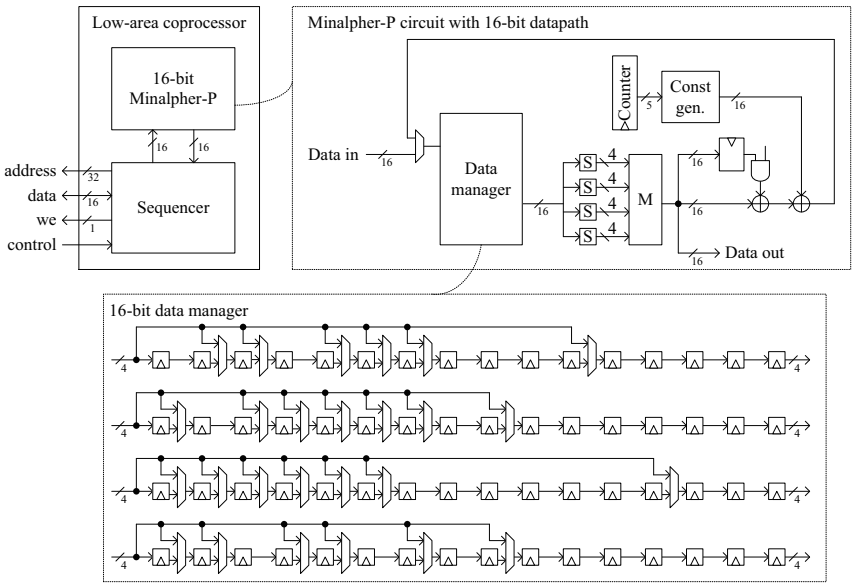


Fig. 49. Low-area coprocessor

An overall architecture of the low-area coprocessor is depicted in Fig. 49. The low-area coprocessor can be thought as the 16-bit Minalpher-P circuit with a thin wrapper for communicating with an external memory through a 16-bit bus. The operation latency is 304: 288 cycles for Minalpher-P and 16 additional cycles for memory transmission.

4.4.4 Mid-range Core

The last implementation is the one for yet another requirement. There is a case where compact circuit area is desired but an external CPU cannot provide much support. In that case, AEAD/MAC-only modes should be supported within the core. The mid-range core is an implementation for such a case.

An overall architecture is shown in Fig. 50. The 16-bit Minalpher-P circuit is employed again. Interconnections between function blocks are similar to the high-speed core, however, all the datapath are shrinked to 16-bit width for the sake of smaller circuit area. Tweak updating is conducted in a function unit labeled "the 16-bit tweak update circuit". That is designed to support both byte-oriented tweak update and 16-bit-word-oriented data feeding. The function unit feeds (i) 16-bit chunks of a tweak sequentially when the Minalpher-P circuit is activated. In other cases, the "$\times y$" and "$\times(y+1)$" tweak

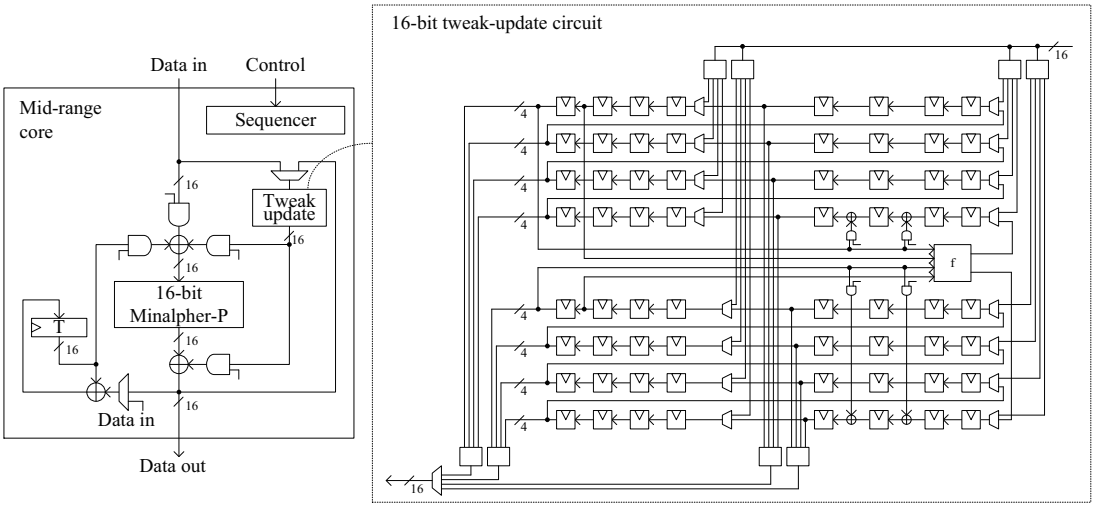


Fig. 50. Mid-range core

Table 5. Performance evaluation

Core	mode	#cycle	Synthesis	Area [kG]	Freq. [MHz]	Thr'put [Mbps]	Efficiency [Mbps/kG]
256-bit Minalpher-P circuit	ENC /DEC	18	Area	4.68	413.22	5,876.95	1,256.56
			Speed	6.25	666.67	9,481.48	1,517.36
16-bit Minalpher-P circuit	ENC /DEC	288	Area	2.70	421.94	375.06	139.05
			Speed	3.28	689.66	613.03	187.11
AES impl. 1	ENC	11	Area	10.49	136.43	1,587.50	151.32
			Speed	16.01	221.73	2,580.12	161.16
AES impl. 2	ENC	226	Area	3.71	89.21	50.52	13.63
			Speed	4.35	165.29	93.62	21.51
Minalpher High-speed core	---	18†	Area	14.32	429.18	6,103.96	426.37
			Speed	16.68	699.30	9,945.61	596.16
Minalpher Low-area coprocessor	---	304†	Area	2.81	438.60	369.34	131.52
			Speed	3.36	666.67	561.40	167.25
Minalpher Mid-range core	---	304†	Area	7.48	215.52	181.49	24.26
			Speed	8.60	595.24	501.25	58.29

updates are conducted in 1 and 64 cycles, respectively. The total number of registers added is 512 bits (i.e., 256 bits for a tweak container and 256 bits for tag accumulation).

4.4.5 Performance Evaluation

The above three cores/coprocessor are implemented with the hardware description language (HDL). They are written in the register-transfer level (RTL) *i.e.*, netlist-level (or cell-level) optimizations are not applied for fair comparison. The circuits are synthesized and evaluated with Synopsys Design Compiler Version G-2012.06-SP5 using the NanGate 45-nm CMOS standard cell library [41] at the worst condition (`NangateOpenCellLibrary_slow.db`). Synthesis is iterated with different performance targets. Other synthesis attributes are the same throughout evaluations. Evaluation results are summarized in Tab. 5[4]. Several performance indices: circuit area (indicated by GE which is normalized by the area of a 2-way NAND), throughput, and the efficiency defined as a throughput-area ratio are shown in the table.

Performances of two different AES implementations are also provided for comparison. They are referred to as (i) impl. 1 and (ii) impl. 2. Impl. 1 is an open-source 1 round/cycle implementation [30].

[4] Note that the number of cycles indicated with daggers (†) are determined by the maximum throughput of the underlying Minalpher-P circuits. The throughputs and efficiencies of these entries are calculated based on the cycle counts. They can be thought as asymptotic performances.

Impl. 2 is based on an unprotected compact implementation in a literature [38] [5]. Both implementations support encryption only, expecting uses for AES-GCM.

Firstly, the Minalpher-P circuits are compared. The 256-bit Minalpher-P circuit is about 3 times faster and 1/2 times smaller compared to the impl. 1. That is explained by the fact that the 8-bit AES S-box is much larger and slower than the 4-bit Minalpher S-box. The smaller number of rounds normalized by the message length (*i.e.* 18-rounds / 256-bit message v.s. 10-rounds / 128-bit message) is another reason for the better throughput. On the other hand, the 16-bit Minalpher-P circuit is about 6 times faster and 3/4 times smaller than impl. 2. The difference is due to the same reasons as the previous comparison. In addition, there is very limited number of selectors for resource sharing as shown in Fig. 49. It is contrast to impl. 2 where many selectors are required. As a result, we can say the Minalpher-P circuits are advantageous to the AES circuits both on speed and area. It is also noted that Šijačić *et al.* reported a smaller circuit area (2.58 kG) with their 32-bit architecture[49].

Secondly, performances of the high-speed core, low-area coprocessor, and mid-range core are discussed.

The high-speed core achieved 9.9 Gbps that is as fast as the 256-bit Minalpher-P circuit. That is because the critical path is within the 256-bit Minalpher-P circuit. The high-speed core used 14-17 kG that is about 3 times larger than that of the 256-bit Minalpher-P circuit. The increase is explained by many additional registers, selectors and XORs. However, the total circuit area (14–17 kG) is still comparable to impl. 1. As a result of the higher throughput and comparable circuit area, the efficiency of the high-speed core is much better than that of impl. 1 (*i.e.*, AES only).

The low-area coprocessor achieved 2.8–3.4 kGE that is comparable to the 16-bit Minalpher-P circuit. That is because the thin wrapper consumes very limited number of gates. The smallest area of 2.81 kGE is still smaller than that of the compact AES implementation (impl. 2). The score is even comparable to some light-weight block-ciphers [20]. The throughput of the low-area coprocessor (369.3–561.4 Mbps) is as fast as that of the 16-bit Minalpher-P circuit, thus it is better than that of impl. 2.

The mid-range core achieves the circuit area of 7.48 kGE. The increase from the low-area coprocessor is about 4.7 kGE. The increase is mainly from the registers (when we estimate 1-bit register with 7–9 GE, then a 512-bit register uses 3.6–4.6 kGE). Nevertheless, the circuit area is smaller than AES impl. 1. The mid-range core will be comparably small compared to a compact AES-GCM implementation. That is because (i) at least 256-bit register and (ii) an integer counter and a finite-field multiplier are added to the 3.7 kGE AES circuit. In addition, the mid-range core will be advantageous in throughput over the compact AES-GCM implementation. That is because throughput of the compact AES-GCM implementation is at most 21.5 Mbps restricted by that of impl. 2. Finally, the efficiency of the mid-range core is better even compared to the compact AES implementation only.

4.5 Protection against Side-Channel Attacks

Minalpher potentially be used in a hostile environment where side-channel attacks (SCA) are concerned. A potential target is the tweak generation in AEAD mode:

$$L \leftarrow (K \| \mathsf{flag_m} \| N) \oplus P(K \| \mathsf{flag_m} \| N).$$

An attacker firstly invokes multiple tweak generations (through dummy encryptions) with the fixed key K and variable nonce N_i. Meanwhile, the attacker records the public nonce N_i and corresponding side-channel traces in the same manner as the conventional SCA. The side-channel attacker can make a distinguisher on hypothesis of a small chunk of K, using the recorded nonce and the traces. More specifically, by guessing 8 bits of the key, the attacker can derive four S-box inputs at the 2nd round. The attacker achieves successful key recovery if sufficiently many traces are provided.

Another potential target is the encryption *i.e.*, $C_i = \varphi_i \oplus P(M_i \oplus \varphi_i)$. There are two options namely attacks using (i) a single long message and (ii) multiple short messages. Feasibility of the attack under the case (i) is remained open. Under the case (ii), on the other hand, the attacker can recover the tweak φ_i with SCA under the fixed-nonce condition. However, since the recovered tweak φ_i is nonce-dependent, thus these approaches will be considered less attractive compared to the previous attack on the tweak generation.

[5] The gate counts 3.7 kGE of the impl. 1 is larger than the one in the original paper by Moradi *et al.* [38] (2.4 kGE) . That is because netlist-level optimizations are not applied. We stress that all the other implementations are implemented under the same restriction.

In such cases, Minalpher-P should be protected against SCA. Since many of the conventional SCA countermeasures are independent of algorithms, thus they can easily be applied to the Minalpher-P. Notably, the small 4-bit S-box is preferable to some countermeasures (*e.g.*, gate-level ones). An exception is the threshold implementation (TI) [29] in which the number of shares (*i.e.*, efficiency of the countermeasure) is dependent to an S-box specification. Therefore, some recent algorithm propositions provide notes on the possibilities of efficient TI realizations [4, 7, 48]. For Minalpher-P, the most efficient 3-share TI can be employed. That is based on the fact that the 3-share TI is available for any 4×4 S-boxes [29]. More specifically, the Minalpher S-box is an element of the affine-equivalent class G_4 [32]. Therefore, the S-box can be decomposed into a 3-stage vectorial boolean functions where their algebraic degrees are 2.

5 Design Rationale

This section explains the design rationale of Minalpher. The designers have not hidden any weaknesses in this cipher.

5.1 Design Rationale for Mode

Minalpher is designed to be block-wise priv-secure in the nonce reuse setting and to be auth-secure in the unverified plaintext release and nonce reuse setting. In addition, Minalpher is designed to be fully parallelizable.

To ensure the auth-security, we employ the Enc-then-MAC approach. This approach ensures the auth-security if the tag verification is a secure MAC. Thus, we employ PMAC [9] as the MAC, which is fully parallelizable.

To ensure the block-wise priv-security, the encryption of Minalpher employs the encryption part of OCB [45], while realizes the security and is fully parallelizable.

For the underlying primitive, Minalpher uses a permutation, while OCB and PMAC use a block-cipher. To realize a tweakable block-cipher using a permutation, we adopt the tweakable Even-Mansour [27, 28]. It is based on Even-Mansour [18] with a single key. The single-key Even-Mansour is also studied by Dunkelman et al. [17]. Thanks to the Even-Mansour like construction, tweakable Even-Mansour is secure against generic attacks as shown in this document and by others [13, 27, 28, 37].

5.1.1 Choice of Tweak Updating Function

Several modes of operation use offset. For example, OCB3 [26] uses $\gamma_i L$ and OCB2 [46] uses $\mathtt{x}^i L$, where L is a secret value and γ_i is an i-th element of Gray code. The multiplication in the offset computation is done in $\mathrm{GF}(2^n)$ and the polynomial representation is used to show an element in $\mathrm{GF}(2^n)$. Considering the implementation aspect, $\gamma_i L$ requires a precomputation table to realize an efficient implementation, and $\mathtt{x}^i L$ requires a bit-level shift which is costly for most software implementation and is not endian neutral for large words, *e.g.*, 128-bit word. To overcome these problems, we adopt a successive extension, $\mathrm{GF}(2^{256}) \cong (\mathrm{GF}(2)[\mathtt{x}]/(f(\mathtt{x})))[\mathtt{y}]/(g(\mathtt{y}))$, where f and g are an irreducible polynomial in $\mathrm{GF}(2)$ and $\mathrm{GF}(2^8)$, respectively, and $\deg f = 8$ and $\deg g = 256/\deg f = 32$.

To realize an efficient implementation, we adopt the following conditions.

1. The number of terms in $g(\mathtt{y})$ should be as small as possible.
2. coefficients of $g(\mathtt{y})$ should be in $\{0, 1, \mathtt{x}\}$ or $\{0, 1, \mathtt{x}^{-1}\}$.
3. $\deg(g(\mathtt{y}) - \mathtt{y}^{32})$ should be as small as possible.

Since the characteristic of the field is 2, there is no 2-term irreducible polynomial with degree 32. There is no 3-term irreducible polynomial satisfying the above condition even if we try all possible $f(\mathtt{x})$. For 4-term irreducible polynomial, there are many candidates, and the smallest $\deg(g(\mathtt{y}) - \mathtt{y}^{32})$ is 3, and we try to choose g from these candidates. Among all these candidates, only one coefficient is $\mathtt{x}$ or $\mathtt{x}^{-1}$ and other coefficients are 0 or 1. Thus, we decide that coefficient $\mathtt{x}$ or $\mathtt{x}^{-1}$ should be located to the constant term considering the efficiency reason. Moreover, we consider that the multiplication by $\mathtt{x}$ is easier than by $\mathtt{x}^{-1}$ on some platform, and the number of terms of f is also better to be as small as possible. Finally, we get a following unique candidate.

$$g(\mathtt{y}) = \mathtt{y}^{32} + \mathtt{y}^3 + \mathtt{y}^2 + \mathtt{x}$$
$$f(\mathtt{x}) = \mathtt{x}^8 + \mathtt{x}^7 + \mathtt{x}^5 + \mathtt{x} + 1$$

Fortunately, the order of the groups generated by $\mathtt{y}$ and $\mathtt{y}+1$ is larger than 2^{128}.

$$\#\langle \mathtt{y} \rangle = (2^{256}-1)/3$$
$$\#\langle \mathtt{y}+1 \rangle = 2^{256}-1$$

Tweakable Even-Mansour uses the elements represented by $\mathtt{y}^i(\mathtt{y}+1)^j$ for offset, where $(i,j) \in \boldsymbol{Z} \times \mathbb{Z}_3 \setminus \{(0,0)\}$ and $\boldsymbol{Z} \subset \mathbb{Z}_{2^{104}}$. From the properties listed above, $\mathtt{y}^i(\mathtt{y}+1)^j \neq 1$ and $\mathtt{y}^i(\mathtt{y}+1)^j \neq \mathtt{y}^{i'}(\mathtt{y}+1)^{j'}$ for any $(i,j),(i',j') \in \boldsymbol{Z} \times \mathbb{Z}_3 \setminus \{(0,0)\}$ such that $(i,j) \neq (i',j')$.

For your curiosity, we confirmed the following condition.

$$\begin{aligned}
\mathtt{y} &= (\mathtt{y}+1)^{\mathrm{c77}} \\
\mathrm{c77} &= 3^2 \times 11 \times 41 \times \mathrm{p73} \\
\mathrm{p73} &= 76156073283962674142663423652696973472457108404329020532153867089370892 69 \quad \text{(prime)}
\end{aligned}$$

Using the polynomials, we can compute a multiplication by $\mathtt{y}$ which is used in an offset as follows.

$$\begin{aligned}
&(A_{31}\mathtt{y}^{31} + A_{30}\mathtt{y}^{30} + A_{29}\mathtt{y}^{29} + \cdots + A_3\mathtt{y}^3 + A_2\mathtt{y}^2 + A_1\mathtt{y} + A_0) \times \mathtt{y} \\
&= A_{30}\mathtt{y}^{31} + A_{29}\mathtt{y}^{30} + \cdots + A_3\mathtt{y}^4 + (A_2 + A_{31})\mathtt{y}^3 + (A_1 + A_{31})\mathtt{y}^2 + A_0\mathtt{y} + A_{31}\mathtt{x}
\end{aligned}$$

As we can see, this multiplication can be done using a left shift with one byte and two byte-wise XOR and one multiplication by $\mathtt{x}$ in $\mathrm{GF}(2^8)$. This is illustrated in Fig. 51. Similar to above computation, we

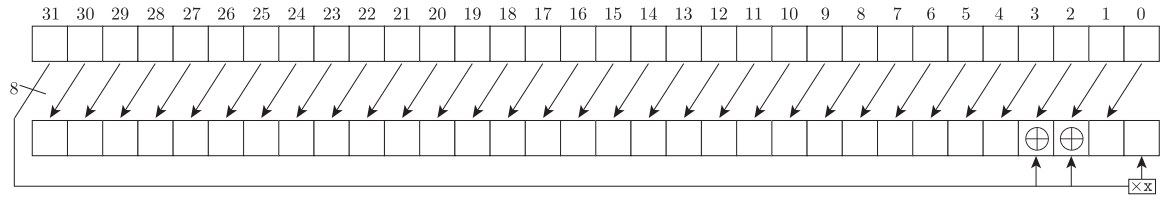


Fig. 51. Multiplication by $\mathtt{y}$

can compute a multiplication by $\mathtt{x}$ as follows.

$$\begin{aligned}
&(a_7\mathtt{x}^7 + a_6\mathtt{x}^6 + a_5\mathtt{x}^5 + a_4\mathtt{x}^4 + a_3\mathtt{x}^3 + a_2\mathtt{x}^2 + a_1\mathtt{x} + a_0) \times \mathtt{x} \\
&= (a_6 + a_7)\mathtt{x}^7 + a_5\mathtt{x}^6 + (a_4 + a_7)\mathtt{x}^5 + a_3\mathtt{x}^4 + a_2\mathtt{x}^3 + a_1\mathtt{x}^2 + (a_0 + a_7)\mathtt{x} + a_7
\end{aligned}$$

Assume A as an 8-bit string, there is a well-known equation to compute a multiplication by $\mathtt{x}$.

$$A \times \mathtt{x} = (A^{\ll 1}) \oplus (\mathrm{msb}(A)\mathtt{0xA3})$$

Note that $A^{\ll 1}$ is 1-bit left shift of A and $\mathrm{msb}(A)$ is the most significant bit of A and $\mathtt{0x}$ means the hexadecimal representation.

5.2 Design Rationale for Minalpher-*P*

In this section, we explain the design rationale for Minalpher-P. Minalpher-P has the Substitution-Permutation Network, and the intermediate states are expressed using matrices like AES. Minalpher-P is designed to achieve high security and efficiency.

5.2.1 Structure

We explain that Minalpher-P is an involution permutation. Minalpher-P consists of the S-function, the T-function and the M-function. In order to design the involution permutation, we design those functions to satisfy the following criteria:

– Each of the S-function, the T-function and the M-function is involutive.
– The S-function and the T-function are commutative.

The transformation from input to output in Minalpher-P can be described as follows, where ST represents the application of the S-function and the T-function in this order, M represents the application of the M-function, and $\oplus E(i-1)$ represents the XOR of the round constant:

$$\text{IN} \rightarrow ST \rightarrow M \rightarrow \oplus E(0) \rightarrow ST \rightarrow \cdots \rightarrow M \rightarrow \oplus E(16) \rightarrow ST \rightarrow \text{OUT}.$$

From the second criterion, the order of S and T can be exchanged. Since the M-function is linear, the order of M and $\oplus E(i-1)$ can be exchanged by applying M to $E(i-1)$. Then, the transformation from output to input in Minalpher-P can be described as follows:

$$\text{IN} \leftarrow TS \leftarrow \oplus M(E(0)) \leftarrow M \leftarrow TS \leftarrow \cdots \leftarrow \oplus M(E(16)) \leftarrow M \leftarrow TS \leftarrow \text{OUT}$$

From the first criterion, each of the S-function, the T-function and the M-function is involutive. Thus, the forward operation and the inverse operation have the same operation except for the values of round constant. For the forward operation, $E(i-1)$ is used in the i-th round, while, for the inverse operation, $M(E(17-i))$ is used in the i-th round.

When the S-function is a nibble-wise involution S-box and the T-function is a nibble-wise position swap, the S-function and T-function are commutative.

5.2.2 *S*-function

5.2.2.1 4-bit S-box versus 8-bit S-box

To be efficient for both of lightweight implementations and fast implementation, we use the 4-bit S-box. To be lightweight, the 4-bit S-box is more advantageous than the 8-bit S-box. Moreover, for implementations on a high-end platform, *e.g.*, the x86(_64) processor, we can parallelly get 16 results calculated from the 4-bit S-box. This shows that the 4-bit S-box can be suitable for fast implementations on a high-end platform.

5.2.2.2 4-bit Involution S-box

For security reasons, the 4-bit S-box fulfills the following criteria:

1. The probability of the maximum differential characteristic is 2^{-2}, which is optimal.
2. The probability of the maximum linear characteristic is 2^{-2}, which is optimal.
3. Each of the 15 non-zero component functions has algebraic degree 3.
4. For all i ($\neq 0$), $s(i) \oplus i \neq s(i)$ is satisfied.

$120,960$ S-boxes fulfill these criteria, and all of these S-boxes are affine equivalent.

5.2.3 *T*-function

The T-function is designed to consist of a nibble-wise position movement function ShuffleRows (SR) and its inverse function SR^{-1} as follows:

$$T(A\|B) = SR^{-1}(B)\|SR(A).$$

For any nibble-wise position movement function SR, the T-function becomes involutive. The input and output of SR and SR^{-1} are expressed by using a 4×8 matrix whose elements are nibble values. SR consists of 4 different nibble-wise position movement functions, and these 4 functions are applied to each row. Let SR_i be a nibble-wise position movement function in the i-th row. From efficiency and security reasons, we have several criteria.

For efficiency reasons, SR fulfills the following two criteria:

1. For 4 indices i, j, k and ℓ, SR_i and SR_j are inverse functions of SR_k and SR_ℓ, respectively.
2. In SR_i, 8 nibbles in each row of the state are regarded as 4 bytes. Then, 3 out of 4 bytes are moved only in byte-wise. The remaining 1 byte is first moved in byte-wise, and the first half nibble of the byte is swapped with the latter half nibble of the byte.

The first criterion is significant for reducing the code size and the table size. The second criterion is significant for fast implementations on microcontrollers.

For security reasons, SR fulfills the following two criteria:

3. For any 8-nibble difference $\Delta x \in \{\{0,1\}^4\}^8$ that 1 nibble is active and 7 nibbles are nonactive, $SR_i(\Delta x)$, $SR_j(\Delta x)$, $SR_i^{-1}(\Delta x)$ and $SR_j^{-1}(\Delta x)$ have different values.
4. Let X, Y and Z be $\left\{\{0,1\}^4\right\}^{4\times 8}$ such that $Y = SR \circ MC(X)$ and $Z = SR(X)$. For any X whose number of active columns is at least 2, the sum of the number of active columns in Y and Z is at least 5.

Criterion 4 depends on the specification of MC. Then we first determine MC from the efficiency and the simplicity, and search for SR to satisfy the criterion 4. By fulfilling four criteria, the minimum number of active S-boxes of 4-round Minalpher-P is ensured to be 22.

5.2.4 M-function

The M-function consists of XorMatrix (XM) and MixColumns (MC). The input data to the M-function is two half states A and B. In the XM operation, the half state B is XORed with the half state A. In the MC operation, 4 nibble values in the same column in the half state A and B are diffused. For efficiency, we use a simple binary matrix whose branch number is 4 so that the operation can be implemented only by XOR operations. The matrix is as follows:

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

Note that this matrix is involutive.

5.2.5 Round Constant

$E(i-1)$ is used as the i-th round constant for forward processing, and $M(E(17-i))$ is used as the i-th round constant for inverse processing. When $E(i)$ is expressed by $0\|RC_i$, $M(E(i))$ is expressed by $0\|MC(RC_i)$. When a column of RC_i is expressed as $[a, a\oplus 1, a\oplus 2, a\oplus 3]$ for any $a \in \{0,1\}^4$, the multiplication by the MC matrix is expressed as follows:

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} a \\ a\oplus 1 \\ a\oplus 2 \\ a\oplus 3 \end{bmatrix} = \begin{bmatrix} a \\ a\oplus 1 \\ a\oplus 2 \\ a\oplus 3 \end{bmatrix} \oplus \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}.$$

This multiplication is very simple, *i.e.*, the conversion from $E(i)$ to $M(E(i))$ can be performed only by adding a constant vector. To prevent the slide attack, we generate $a \in \{0,1\}^4$ from the round number.

6 Intellectual Property

We are not applying for any patent about Minalpher. To the best of our knowledge, Minalpher does not infringe upon the rights on other designs. 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."

7 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 submitters understand that if they disagree with published analyses then they are 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.

References

1. Jean-Philippe Aumasson, Emilia Käsper, Lars Ramkilde Knudsen, Krystian Matusiewicz, Rune Ødegård, Thomas Peyrin, and Martin Schläffer. Distinguishers for the Compression Function and Output Transformation of Hamsi-256. In Ron Steinfeld and Philip Hawkes, editors, *ACISP 2010*, volume 6168 of *LNCS*, pages 87–103. Springer, 2010.
2. Mihir Bellare and Phillip Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. Cryptology ePrint Archive, Report 2004/331, 2006.
3. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, volume 7118 of *LNCS*, pages 320–337. Springer, 2011.
4. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The Keccak reference. Keccak home page, 2011. Version 3.0, `http://keccak.noekeon.org/`.
5. Eli Biham, Orr Dunkelman, and Nathan Keller. The Rectangle Attack - Rectangling the Serpent. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *LNCS*, pages 340–357. Springer, 2001.
6. Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *CRYPTO*, volume 537 of *LNCS*, pages 2–21. Springer, 1990.
7. Begül Bilgin, Andrey Bogdanov, Miroslav Knezevic, Florian Mendel, and Qingju Wang. Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES*, volume 8086 of *LNCS*, pages 142–158. Springer, 2013.
8. Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 231–249. Springer, 2009.
9. John Black and Phillip Rogaway. A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *LNCS*, pages 384–397. Springer, 2002.
10. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.
11. Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-Order Differential Properties of Keccak and *Luffa*. In Antoine Joux, editor, *FSE*, volume 6733 of *LNCS*, pages 252–269. Springer, 2011.
12. David Chaum and Jan-Hendrik Evertse. Crytanalysis of DES with a Reduced Number of Rounds: Sequences of Linear Factors in Block Ciphers. In Hugh C. Williams, editor, *CRYPTO*, volume 218 of *LNCS*, pages 192–211. Springer, 1985.
13. Benoit Cogliati, Rodolphe Lampe, and Yannick Seurin. Tweaking Even-Mansour ciphers. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 189–208. Springer, 2015.
14. Baudoin Collard, François-Xavier Standaert, and Jean-Jacques Quisquater. Improving the Time Complexity of Matsui's Linear Cryptanalysis. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC*, volume 4817 of *LNCS*, pages 77–88. Springer, 2007.
15. Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The Block Cipher Square. In Eli Biham, editor, *FSE*, volume 1267 of *LNCS*, pages 149–165, 1997.
16. Whitfield Diffie and Martin E. Hellman. Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *Computer*, Issue: 6(10), 1977.
17. Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in cryptography: The Even-Mansour scheme revisited. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 336–354. Springer, 2012.
18. Shimon Even and Yishay Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *Journal of Cryptology*, 10(3):151–162, 1997.

19. Henri Gilbert and Thomas Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-like Permutations. In Seokhie Hong and Tetsu Iwata, editors, *FSE*, volume 6147 of *LNCS*, pages 365–383. Springer, 2010.
20. International Organization for Standardization. *ISO/IEC 29192-4:2013, Information technology – Security techniques – Lightweight cryptography – Part 4: Mechanisms*, 2013.
21. Mitsugu Iwamoto, Thomas Peyrin, and Yu Sasaki. Limited-Birthday Distinguishers for Hash Functions - Collisions beyond the Birthday Bound Can Be Meaningful. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (2)*, volume 8270 of *LNCS*, pages 504–523. Springer, 2013.
22. Jérémy Jean, María Naya-Plasencia, and Thomas Peyrin. Improved Rebound Attack on the Finalist Grøstl. In Anne Canteaut, editor, *FSE*, volume 7549 of *LNCS*, pages 110–126. Springer, 2012.
23. John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In Bruce Schneier, editor, *FSE*, volume 1978 of *LNCS*, pages 75–93. Springer, 2000.
24. Lars R. Knudsen and Vincent Rijmen. Known-Key Distinguishers for Some Block Ciphers. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *LNCS*, pages 315–324. Springer, 2007.
25. Lars R. Knudsen and David Wagner. Integral Cryptanalysis. In *FSE*, volume 2365 of *LNCS*, pages 112–127. Springer, 2002.
26. Ted Krovetz and Phillip Rogaway. The Software Performance of Authenticated-Encryption Modes. In Antoine Joux, editor, *FSE*, volume 6733 of *LNCS*, pages 306–327. Springer, 2011.
27. Kaoru Kurosawa. Power of a public random permutation and its application to authenticated-encryption. *IACR Cryptology ePrint Archive*, 2002:127, 2002.
28. Kaoru Kurosawa. Power of a public random permutation and its application to authenticated encryption. *IEEE Transactions on Information Theory*, 56(10):5366–5374, 2010.
29. Sebastian Kutzner, Phuong Ha Nguyen, and Axel Poschmann. Enabling 3-share Threshold Implementations for any 4-bit S-box. IACR Cryptology ePrint Archive, Report 2012/510, 2012.
30. Aoki Laboratory. Cryptographic Hardware Project at Tohoku University. project home page. `http://www.aoki.ecei.tohoku.ac.jp/crypto/`.
31. Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound Distinguishers: Results on the Full Whirlpool Compression Function. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *LNCS*, pages 126–143. Springer, 2009.
32. Gregor Leander and Axel Poschmann. On the Classification of 4 Bit S-Boxes. In Claude Carlet and Berk Sunar, editors, *WAIFI*, volume 4547 of *LNCS*, pages 159–176. Springer, 2007.
33. Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable Block Ciphers. *Journal of Cryptology*, 24(3):588–613, 2011.
34. Krystian Matusiewicz, María Naya-Plasencia, Ivica Nikolić, Yu Sasaki, and Martin Schläffer. Rebound Attack on the Full LANE Compression Function. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *LNCS*, pages 106–125. Springer, 2009.
35. David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *LNCS*, pages 343–355. Springer, 2004.
36. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In Orr Dunkelman, editor, *FSE*, volume 5665 of *LNCS*, pages 260–276. Springer, 2009.
37. Bart Mennink. XPX: generalized tweakable Even-Mansour with improved security guarantees. *IACR Cryptology ePrint Archive*, 2015:476, 2015.
38. Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *LNCS*, pages 69–88. Springer, 2011.
39. Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Inscrypt*, volume 7537 of *LNCS*, pages 57–76. Springer, 2011.
40. Sean Murphy. The Return of the Cryptographic Boomerang. *IEEE Transactions on Information Theory*, 57(Issue: 4):2517 – 2521, April 2011.
41. NanGate. Open Cell Library. project home page. `http://www.nangate.com/`.
42. National Institute of Standards and Technology. *NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, November 2007.
43. Ivica Nikolić, Josef Pieprzyk, Przemyslaw Sokolowski, and Ron Steinfeld. Known and Chosen Key Differential Distinguishers for Block Ciphers. In Kyung Hyune Rhee and DaeHun Nyang, editors, *ICISC*, volume 6829 of *LNCS*, pages 29–48. Springer, 2011.
44. Yuichi Niwa, Keisuke Ohashi, Kazuhiko Minematsu, and Tetsu Iwata. Constants in GCM Security Bounds Are at Least .... A talk at rump session of FSE 2014. `http://fse.2014.rump.cr.yp.to/`.
45. Phillip Rogaway and Mihir Bellare and John Black and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *ACM Conference on Computer and Communications Security*, pages 196–205. ACM, 2001.

46. Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *LNCS*, pages 16–31. Springer, 2004.
47. Yu Sasaki, Yuuki Tokushige, Lei Wang, Mitsugu Iwamoto, and Kazuo Ohta. An Automated Evaluation Tool for Improved Rebound Attack: New Distinguishers and Proposals of ShiftBytes Parameters for Grøstl. In Josh Benaloh, editor, *CT-RSA*, volume 8366 of *LNCS*, pages 424–443. Springer, 2014.
48. Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An Ultra-Lightweight Blockcipher. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES*, volume 6917 of *LNCS*, pages 342–357. Springer, 2011.
49. Danilo Šijačić, Bohan Yang, and Begül Bilgin. Minalpher & primates: Overview of lightweight hardware implementation results. CAESAR mailing list, 2015. `https://groups.google.com/group/crypto-competitions/attach/3d3bb64029b0987/minalpher_primates_hw.pdf`.
50. Yosuke Todo. FFT-Based Key Recovery for the Integral Attack. Cryptology ePrint Archive, Report 2014/187, 2014.
51. David Wagner. The Boomerang Attack. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.
52. Kan Yasuda. Incremental Authenticated Encryption. A talk at Dagstuhl Seminar 14021 Symmetric Cryptography, 2014. `http://www.dagstuhl.de/en/program/calendar/semhp/?semnr=14021`.