PHOTON-Beetle Cipher Family

By Ajay Tarole

Introduction

1. It uses a sponge-based mode Beetle with the P_{256} being the underlying permutation.
2. We denote this permutation by $\mathtt{PHOTON}_{256}$.
3. Based on the functionalities, PHOTON-Beetle can be classified into two categories:

   (i) a family of authenticated encryptions, dubbed as PHOTON-Beetle-AEAD.

   (ii) a family of hash functions, dubbed as PHOTON-Beetle-Hash.

4. Both these families are parameterized by r, the rate of message absorption.

Notations

1. $\{0,1\}^*$ we denote the set of all strings, and by $\{0,1\}^n$ the set of strings of length n.
2. $|A|$ denotes the number of the bits in the string A.
3. We use the notation $\oplus$ and $\odot$ to refer the binary addition and matrix multiplication respectively.
4. For $A,B \in \{0,1\}^*$, $A||B$ to denotes the concatenation of A and B.
5. We use the notation $V_1\|\cdots\|V_v \xleftarrow{(a_1,\ldots,a_v)} V$ to denote parsing of the string V into v vectors of size a1 , . . . , av respectively.
6. $B \ggg k$ denotes k bit right-rotation of the bit string B.
7. The expression E? a : b evaluates to a if E holds and b otherwise.

Notations

8. (E1 and E2)? a : b : c : d evaluates to a if both E1 and E2 holds, b if

only E 1 holds, c if only E 2 holds and d otherwise.

9. Trunc(V, i) is a function that returns the most significant i bits of the V and Ozs is the function that applies 10* padding on r bits, i.e $\mathsf{Ozs}_r(V) = V\|1\|0^{r-|V|-1}$ When $|V|<r$.

10. For any two integers m and n, we use m| n to denote that m divides n.

11. For any matrix X, we use the notation X[i, j] to denote the element at i-th row and j-th column of X.

Notations

12. We represent a serial matrix Serial[a0 , a1 , a2 , a3 , a4 , a5 , a6 , a7] by

$$\mathsf{Serial}[a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7] := \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \end{pmatrix}.$$

$\mathtt{PHOTON}_{256}$ PERMUTATION

1. We use $\mathtt{PHOTON}_{256}$ as the underlying 256-bit permutation in our mode.
2. It is applied on a state of 64 elements of 4 bits each, which is represented as a (8×8) matrix X.
3. It is composed of 12 rounds, each containing four layers AddConstant, SubCells, ShiftRows and MixColumnSerial.
4. AddConstant adds fixed constants to the cells of the internal state.
5. SubCells applies an 4-bit S-Box to each of the 64 4-bit cells.
6. ShiftRows rotates the position of the cells in each of the rows.
7. MixColumnSerial linearly mixes all the columns independently using a serial matrix multiplication. The multiplication with the coefficients in the matrix is in $GF(2^4)$ with x^4 + x + 1 being the irreducible polynomial.

PHOTON$_{256}$ PERMUTATION

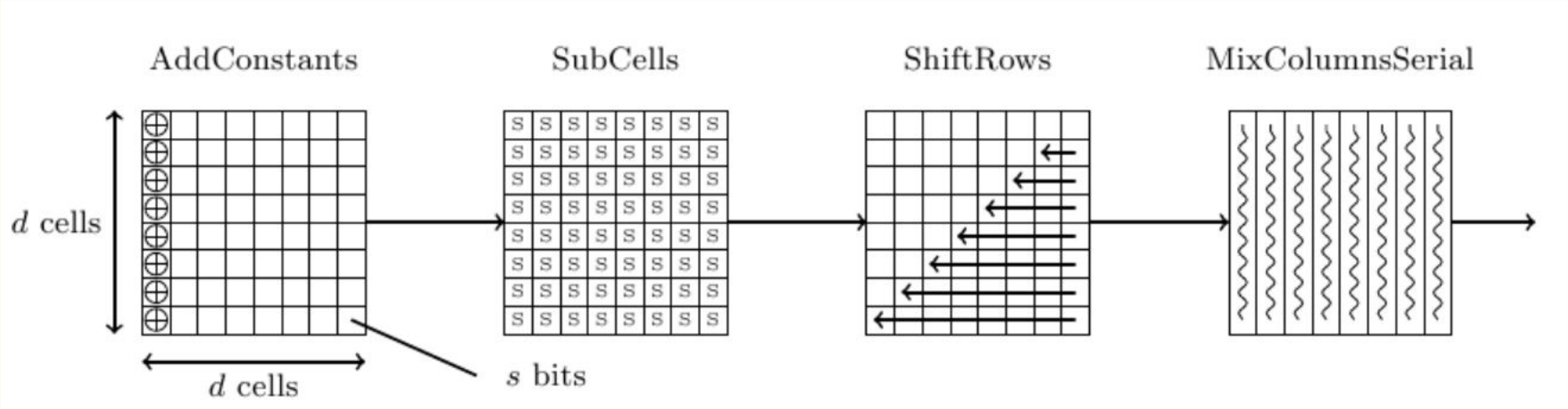


Figure 1. One Round of PHOTON Internal Permutation

The PHOTON S-box

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S-box	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Formal description of all operations

$\texttt{PHOTON}_{256}(X)$

$\texttt{for } i = 0 \texttt{ to } 11:$
$\quad X \leftarrow \mathsf{AddConstant}(X, i);$
$\quad X \leftarrow \mathsf{SubCells}(X);$
$\quad X \leftarrow \mathsf{ShiftRows}(X);$
$\quad X \leftarrow \mathsf{MixColumnSerial}(X);$
$\textbf{return } X;$

$\mathsf{AddConstant}(X, k)$

$RC[12] \leftarrow \{1, 3, 7, 14, 13, 11, 6, 12, 9, 2, 5, 10\};$
$IC[8] \leftarrow \{0, 1, 3, 7, 15, 14, 12, 8\};$
$\texttt{for } i = 0 \texttt{ to } 7:$
$\quad X[i, 0] \leftarrow X[i, 0] \oplus RC[k] \oplus IC[i];$
$\textbf{return } X;$

$\mathsf{SubCells}(X)$

$\texttt{for } i = 0 \texttt{ to } 7, j = 0 \texttt{ to } 7:$
$\quad X[i, j] \leftarrow \mathsf{S\text{-}Box}(X[i, j]);$
$\textbf{return } X;$

$\mathsf{ShiftRows}(X)$

$\texttt{for } i = 0 \texttt{ to } 7, j = 0 \texttt{ to } 7:$
$\quad X'[i, j] \leftarrow X[i, (j + i)\%8]);$
$\textbf{return } X';$

$\mathsf{MixColumnSerial}(X)$

$M \leftarrow \mathsf{Serial}[2, 4, 2, 11, 2, 8, 5, 6];$
$X \leftarrow M^8 \odot X;$
$\textbf{return } X;$

Specification of PHOTON-Beetle Family

1. Mathematical Component ρ and ρ^{-1}

$\rho(S, U)$	$\rho^{-1}(S, V)$	$\mathsf{Shuffle}(S)$
$1: \quad V \leftarrow \mathsf{Trunc}(\mathsf{Shuffle}(S), \lvert U \rvert) \oplus U;$	$1: \quad U \leftarrow \mathsf{Trunc}(\mathsf{Shuffle}(S), \lvert V \rvert) \oplus V;$	$1: \quad S_1 \Vert S_2 \xleftarrow{r/2} S;$
$2: \quad S \leftarrow S \oplus \mathsf{Ozs}_r(U);$	$2: \quad S \leftarrow S \oplus \mathsf{Ozs}_r(U);$	$\textbf{return} \quad S_2 \Vert (S_1 \ggg 1);$
$\textbf{return} \quad (S, V);$	$\textbf{return} \quad (S, U);$	

Where ρ is a linear function, state $S \in \{0,1\}^r$ and input data $U \in \{0,1\}^{\leq r}$. ρ produces an output da $V \in \{0,1\}^{|U|}$.

Where ρ^{-1} is a inverse function of ρ, which takes the state S and the output data V to reproduce the input data U and update the state.

Specification of PHOTON-Beetle Family

2. PHOTON-Beetle-AEAD Authenticated Encryption

$\mathsf{PHOTON\text{-}Beetle\text{-}AEAD.ENC}[r](K, N, A, M)$

$\quad \mathsf{IV} \leftarrow N \| K;\ C \leftarrow \lambda;$
$\quad \mathtt{if}\ A = \lambda, M = \lambda:$
$\qquad T \leftarrow \mathsf{TAG}_{128}(\mathsf{IV} \oplus 1);\ \mathtt{return}\ (\lambda, T);$
$\quad c_0 \leftarrow (M \neq \lambda\ \mathtt{and}\ r \mid |A|)?\ 1:2:3:4$
$\quad c_1 \leftarrow (A \neq \lambda\ \mathtt{and}\ r \mid |M|)?\ 1:2:5:6$
$\quad \mathtt{if}\ A \neq \lambda:$
$\qquad \mathsf{IV} \leftarrow \mathsf{HASH}_r(\mathsf{IV}, A, c_0);$
$\quad \mathtt{if}\ M \neq \lambda:$
$\qquad M_1 \| \cdots \| M_m \xleftarrow{r} M;$
$\qquad \mathtt{for}\ i = 1\ \mathtt{to}\ m:$
$\qquad\quad Y \| Z \xleftarrow{(r, 256-r)} \mathsf{PHOTON}_{256}(\mathsf{IV});$
$\qquad\quad (W, C_i) \leftarrow \rho(Y, M_i);$
$\qquad\quad \mathsf{IV} \leftarrow W \| Z;$
$\qquad \mathsf{IV} \leftarrow \mathsf{IV} \oplus c_1;$
$\qquad C \leftarrow C_1 \| \cdots \| C_m;$
$\quad T \leftarrow \mathsf{TAG}_{128}(\mathsf{IV});$
$\mathbf{return}\ (C, T);$

$\mathsf{PHOTON\text{-}Beetle\text{-}AEAD.DEC}[r](K, N, A, C, T)$

$\quad \mathsf{IV} \leftarrow N \| K;\ M \leftarrow \lambda;$
$\quad \mathtt{if}\ A = \lambda, C = \lambda:$
$\qquad T^\star \leftarrow \mathsf{TAG}_{128}(\mathsf{IV} \oplus 1);$
$\qquad \mathtt{return}\ (T = T^\star)?\ \lambda : \perp;$
$\quad c_0 \leftarrow (C \neq \lambda\ \mathtt{and}\ r \mid |A|)?\ 1:2:3:4$
$\quad c_1 \leftarrow (A \neq \lambda\ \mathtt{and}\ r \mid |C|)?\ 1:2:5:6$
$\quad \mathtt{if}\ A \neq \lambda:$
$\qquad \mathsf{IV} \leftarrow \mathsf{HASH}_r(\mathsf{IV}, A, c_0);$
$\quad \mathtt{if}\ C \neq \lambda:$
$\qquad C_1 \| \cdots \| C_m \xleftarrow{r} C;$
$\qquad \mathtt{for}\ i = 1\ \mathtt{to}\ m:$
$\qquad\quad Y \| Z \xleftarrow{(r, 256-r)} \mathsf{PHOTON}_{256}(\mathsf{IV});$
$\qquad\quad (W, M_i) \leftarrow \rho^{-1}(Y, C_i);$
$\qquad\quad \mathsf{IV} \leftarrow W \| Z;$
$\qquad \mathsf{IV} \leftarrow \mathsf{IV} \oplus c_1;$
$\qquad M \leftarrow M_1 \| \cdots \| M_m;$
$\quad T^\star \leftarrow \mathsf{TAG}_{128}(\mathsf{IV});$
$\mathbf{return}\ (T = T^\star)?\ M : \perp;$

Specification of PHOTON-Beetle Family

2. PHOTON-Beetle-AEAD Authenticated Encryption

PHOTON-Beetle-Hash$[r](M)$

$\texttt{if}\ M = \lambda:$
  $\mathsf{IV} \leftarrow 0\|0;$
  $T \leftarrow \mathsf{TAG}_{256}(\mathsf{IV} \oplus 1);\ \texttt{return}\ T;$
$\texttt{if}\ |M| \leq 128:$
  $c_0 \leftarrow (|M| < 128)?\ 1 : 2$
  $\mathsf{IV} \leftarrow \mathsf{Ozs}_{128}(M)\|0;$
  $T \leftarrow \mathsf{TAG}_{256}(\mathsf{IV} \oplus c_0);\ \texttt{return}\ T;$
$M_1\|M' \xleftarrow{(128,|M|-128)} M;$
$c_0 \leftarrow (r|\ |M'|)?\ 1 : 2$
$\mathsf{IV} \leftarrow M_1\|0$
$\mathsf{IV} \leftarrow \mathsf{HASH}_r(\mathsf{IV}, M', c_0);$
$T \leftarrow \mathsf{TAG}_{256}(\mathsf{IV});$
return $T;$

$\mathsf{HASH}_r(\mathsf{IV}, D, c_0)$

$D_1\|\cdots\|D_d \xleftarrow{r} \mathsf{Ozs}_r(D);$
$\texttt{for}\ i = 1\ \texttt{to}\ d:$
  $Y\|Z \xleftarrow{(r,256-r)} \texttt{PHOTON}_{256}(\mathsf{IV});$
  $W \leftarrow Y \oplus D_i;$
  $\mathsf{IV} \leftarrow W\|Z;$
$\mathsf{IV} \leftarrow \mathsf{IV} \oplus c_0;$
return $\mathsf{IV};$

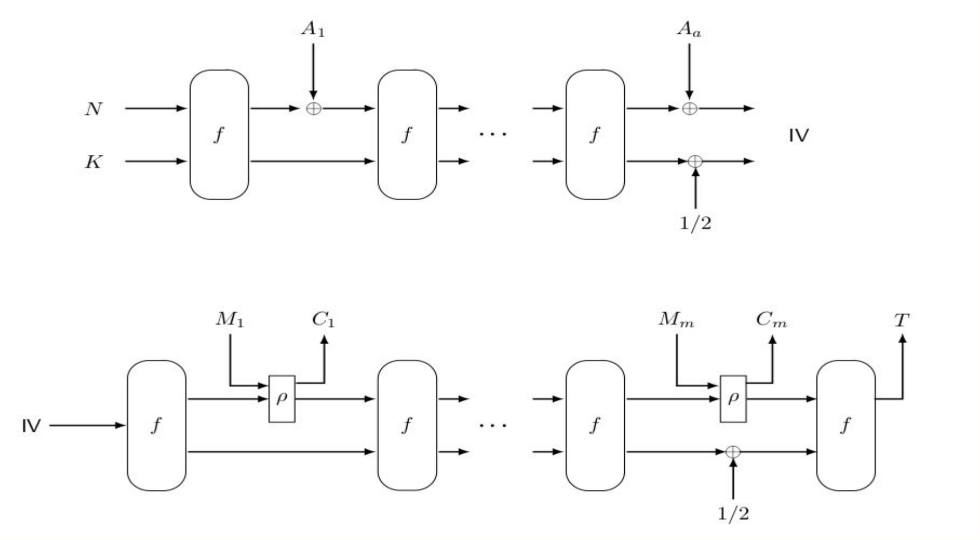
$\mathsf{TAG}_\tau(T_0)$

$\texttt{for}\ i = 1\ \texttt{to}\ \lceil \tau/128 \rceil:$
  $T_i \leftarrow \texttt{PHOTON}_{256}(T_{i-1});$
$T \leftarrow \mathsf{Trunc}(T_1, 128)\ \|\cdots\|\ \mathsf{Trunc}(T_{\tau/128}, 128);$
return $T;$

Specification of PHOTON-Beetle Family

PHOTON-Beetle-AEAD Authenticated Encryption

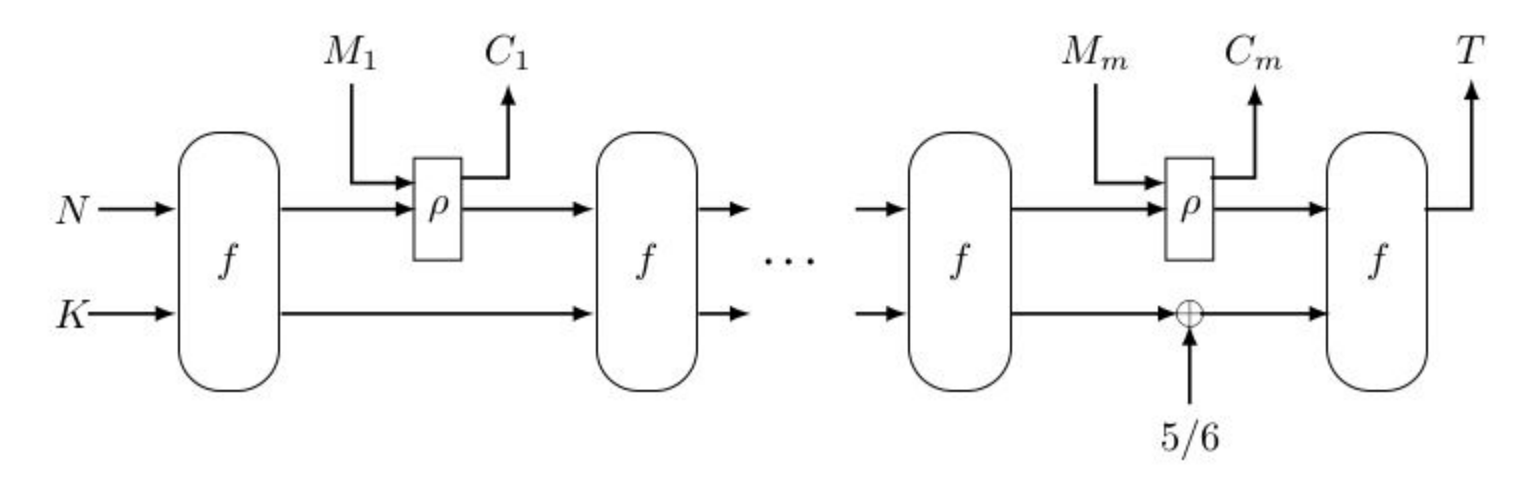
Case 1. PHOTON-Beetle-AEAD.ENC with a AD blocks and m message blocks.



Specification of PHOTON-Beetle Family

PHOTON-Beetle-AEAD Authenticated Encryption

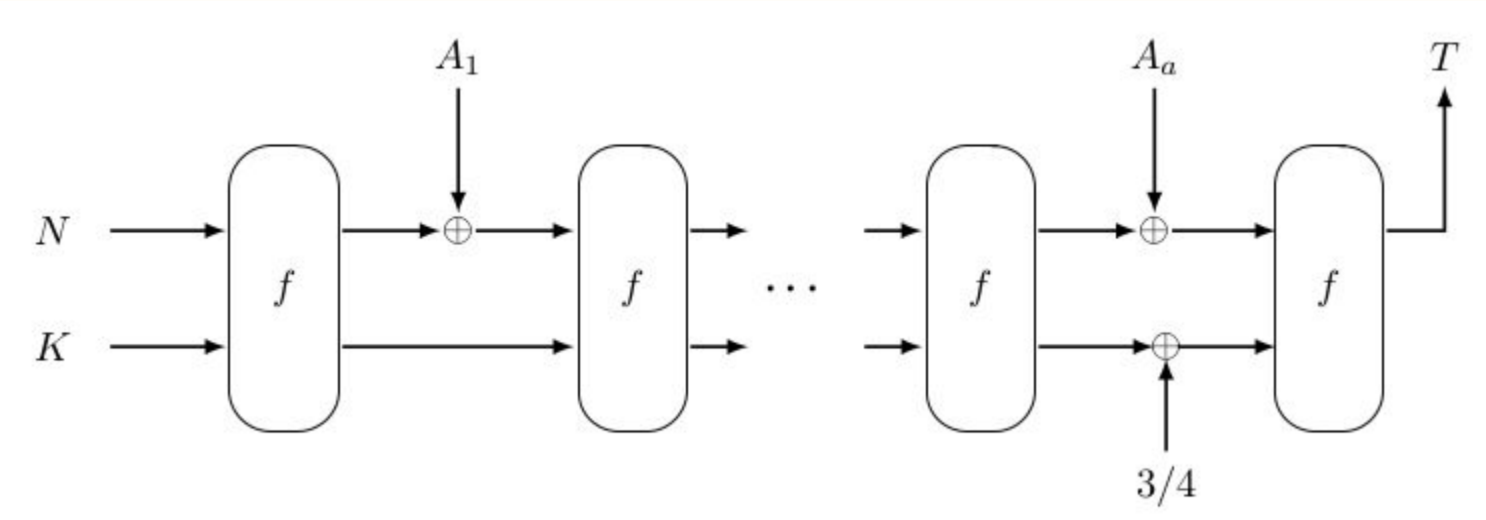
Case 2. PHOTON-Beetle-AEAD.ENC with empty AD and m message blocks.



Specification of PHOTON-Beetle Family

PHOTON-Beetle-AEAD Authenticated Encryption

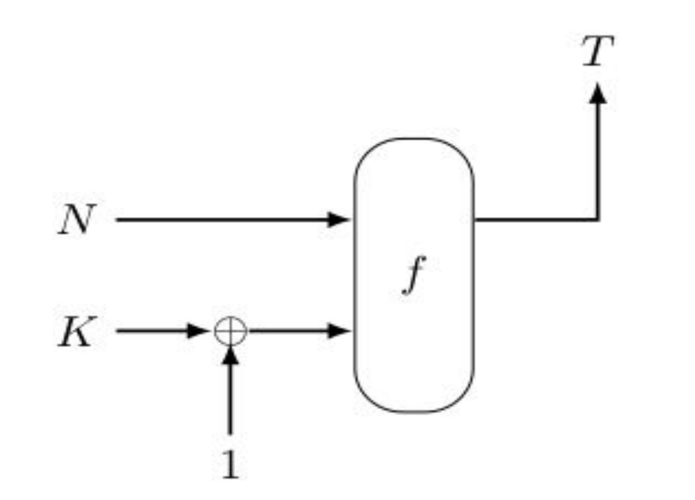
Case 3. PHOTON-Beetle-AEAD.ENC Construction with a AD blocks and empty message.



Specification of PHOTON-Beetle Family

PHOTON-Beetle-AEAD Authenticated Encryption

Case 4. PHOTON-Beetle-AEAD.ENC Construction with empty AD and empty message.

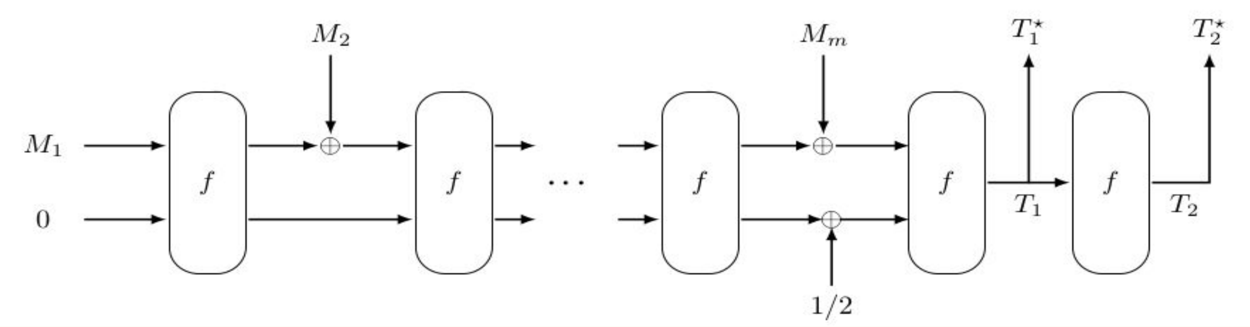


Specification of PHOTON-Beetle Family

3. PHOTON-Beetle-Hash Hash function

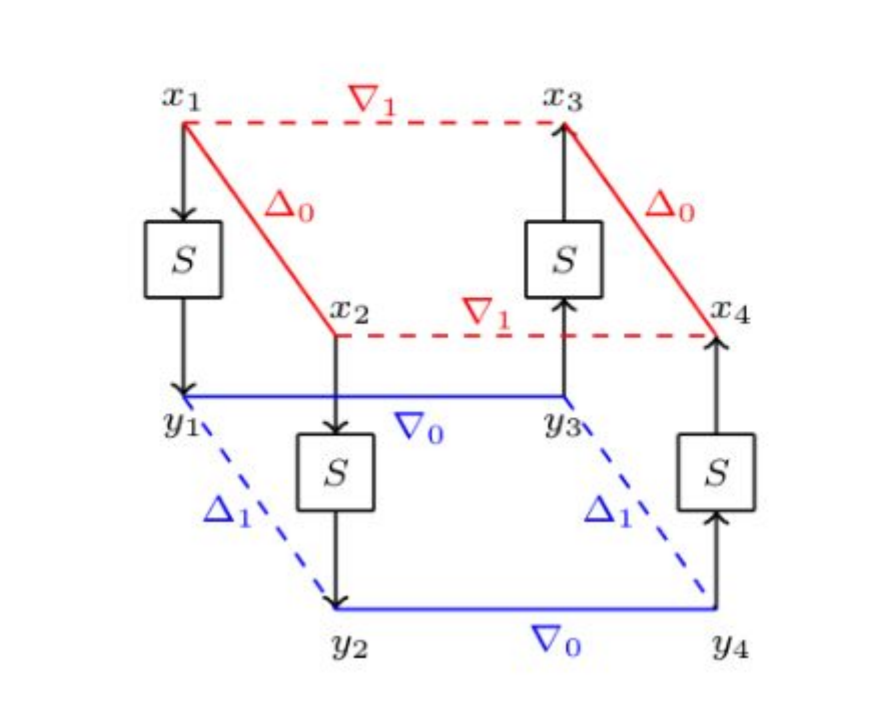
PHOTON-Beetle-Hash takes a message M ∈ {0, 1}* and generates a tag $T \in \{0,1\}^{256}$

.



Here |M1| = 128, |Mi| = r, for i = 2, . . . , m -1 and |Mm| ≤ r. The tag T is computed as T1*|| T2* , where Ti* = Trunc(T i , 128).

SBox Analysis of PHOTON Beetle



SBox Analysis of PHOTON Beetle

1. Difference Distribution Table(DDT)

$$\mathrm{DDT}(\Delta_0, \Delta_1) = \#\{x \in \{0,1\}^n | (S(x) \oplus S(x \oplus \Delta_0) = \Delta_1\}.$$

2. 1-1 bit DDT

Table: 1-1 bit DDT

$\Delta_x \setminus \Delta_y$	0001	0010	0100	1000
bit 0 = 0001	0	0	0	0
bit 1 = 0010	0	0	0	0
bit 2 = 0100	0	0	0	0
bit 3 = 1000	0	0	0	0

SBox Analysis of PHOTON Beetle

BOGI permutation:

$$|BO| \leq |GI| \Rightarrow |GI| + |GO| \geq 4$$

Note: Score of S-box $|GI| + |GO| == 8$.

SBox Analysis of PHOTON Beetle

The Boomerang Connectivity Table(BCT)

$$\mathrm{BCT}(\Delta_0, \nabla_0) = \#\{x \in \{0,1\}^n | S^{-1}(S(x) \oplus \nabla_0) \oplus S^{-1}(S(x \oplus \Delta_0) \oplus \nabla_0) = \Delta_0\}.$$

1. The ladder switch is captured by the BCT in the case where at least one of the index equals to zero.
2. The S-box switch is captured by the BCT in the case where $\Delta 1$ equals $\nabla 0$.
3. The incompatibility pointed out by Murthy simply corresponds to zero entries in the BCT.

Boomerang Connectivity Table(BCT)

In \ Out	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
1	16	0	4	4	0	16	4	4	4	4	0	0	4	4	0	0
2	16	0	0	6	0	4	6	0	0	0	2	0	2	2	2	0
3	16	2	0	6	2	4	4	2	0	0	2	2	0	0	0	0
4	16	0	0	0	0	4	2	2	0	6	2	0	6	0	2	0
5	16	2	0	0	2	4	0	0	0	6	2	2	4	2	0	0
6	16	4	2	0	4	0	2	0	2	0	0	4	2	0	4	8
7	16	4	2	0	4	0	2	0	2	0	0	4	2	0	4	8
8	16	4	0	2	4	0	0	2	0	2	0	4	0	2	4	8
9	16	4	2	0	4	0	2	0	2	0	0	4	2	0	4	8
a	16	0	2	2	0	4	0	0	6	0	2	0	0	6	2	0
b	16	2	0	0	2	4	0	0	4	2	2	2	0	6	0	0
c	16	0	6	0	0	4	0	6	2	2	2	0	0	0	2	0
d	16	2	4	2	2	4	0	6	0	0	2	2	0	0	0	0
e	16	0	2	2	0	0	2	2	2	2	0	0	2	2	0	0
f	16	8	0	0	8	0	0	0	0	0	0	8	0	0	8	16

S Box Switch

BCT($\Delta 0$, $\nabla 0$)

In \ Out	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
1	16	0	4	4	0	16	4	4	4	4	0	0	4	4	0	0
2	16	0	0	6	0	4	6	0	0	0	2	0	2	2	2	0
3	16	2	0	6	2	4	4	2	0	0	2	2	0	0	0	0
4	16	0	0	0	0	4	2	2	0	6	2	0	6	0	2	0
5	16	2	0	0	2	4	0	0	0	6	2	2	4	2	0	0
6	16	4	2	0	4	0	2	0	2	0	0	4	2	0	4	8
7	16	4	2	0	4	0	2	0	2	0	0	4	2	0	4	8
8	16	4	0	2	4	0	0	2	0	2	0	4	0	2	4	8
9	16	4	2	0	4	0	2	0	2	0	0	4	2	0	4	8
a	16	0	2	2	0	4	0	0	6	0	2	0	0	6	2	0
b	16	2	0	0	2	4	0	0	4	2	2	2	0	6	0	0
c	16	0	6	0	0	4	0	6	2	2	2	0	0	0	2	0
d	16	2	4	2	2	4	0	6	0	0	2	2	0	0	0	0
e	16	0	2	2	0	0	2	2	2	2	0	0	2	2	0	0
f	16	8	0	0	8	0	0	0	0	0	0	8	0	0	8	16

DDT($\Delta 0$, $\Delta 1$)

$\Delta_0 \setminus \Delta_1$	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	4	0	0	0	4	0	4	0	0	0	4	0	0
2	0	0	0	2	0	4	2	0	0	0	2	0	2	2	2	0
3	0	2	0	2	2	0	4	2	0	0	2	2	0	0	0	0
4	0	0	0	0	0	4	2	2	0	2	2	0	2	0	2	0
5	0	2	0	0	2	0	0	0	0	2	2	2	4	2	0	0
6	0	0	2	0	0	0	2	0	2	0	0	4	2	0	0	4
7	0	4	2	0	0	0	2	0	2	0	0	0	2	0	0	4
8	0	0	0	2	0	0	0	2	0	2	0	4	0	2	0	4
9	0	0	2	0	4	0	2	0	2	0	0	0	2	0	4	0
a	0	0	2	2	0	4	0	0	2	0	2	0	0	2	2	0
b	0	2	0	0	2	0	0	0	4	2	2	2	0	2	0	0
c	0	0	2	0	0	4	0	2	2	2	2	0	0	0	2	0
d	0	2	4	2	2	0	0	2	0	0	2	2	0	0	0	0
e	0	0	2	2	0	0	2	2	2	2	0	0	2	2	0	0
f	0	4	0	0	4	0	0	0	0	0	0	0	0	0	4	4

Zero-Sum Distinguisher for PHOTON Permutation f

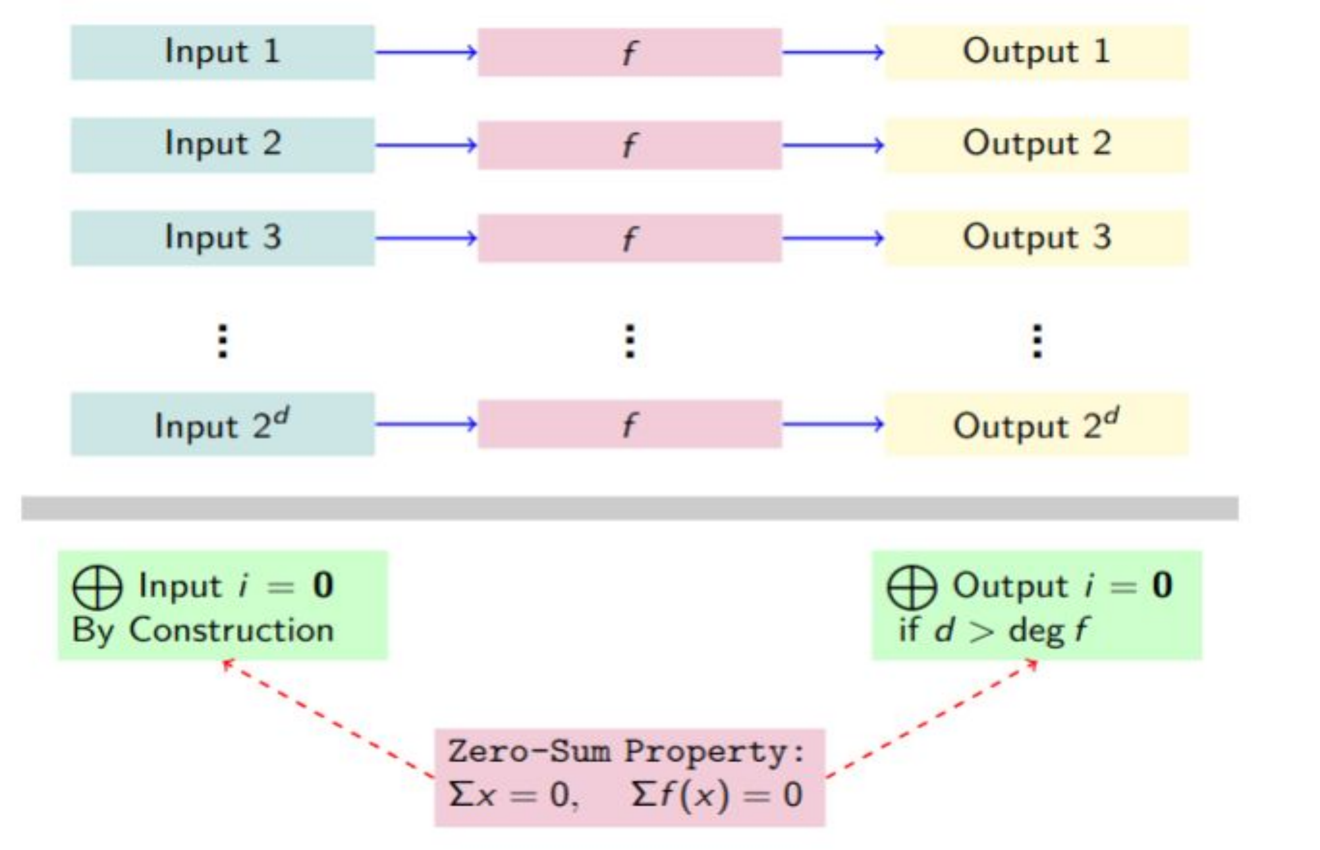
We use Derivative of Boolean Functions to construct Zero Sum Distinguisher for n-Round Distinguisher.

Degree of S Box $d = 3$ and r is the number of rounds, then we have to very our input

to $d^{r}+1$ bits.

A distinguisher is infeasible to construct for almost all values of $n > 4$ on a reasonably powered machine.

For $n=5$, our # of inputs are 2^{244}. Very big number.



Input 1
f
Output 1
Input 2
f
Output 2
Input 3
f
Output 3
Input 2^d
f
Output 2^d
$\bigoplus$ Input $i = \mathbf{0}$
By Construction
$\bigoplus$ Output $i = \mathbf{0}$
if $d > \deg f$
Zero-Sum Property:
$\Sigma x = 0, \quad \Sigma f(x) = 0$

Thank You

References:

https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/sbox.html

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9264171

https://www.isical.ac.in/~lightweight/beetle/

https://asecuritysite.com/encryption/beetle