On Some Properties of an XSL-network

Alexey Kurochkin

TC 26

28.05.2018

Definition

Let F be a function from F_2^n into F_2^m. A zero-sum for F of size (dimension) K is a subset $\{x_1, \ldots, x_K\} \subset F_2^n$ of elements which sum to zero and for which the corresponding images by F also sum to zero, i.e.

$$\bigoplus_{i=1}^{K} x_i = \bigoplus_{i=1}^{K} F(x_i) = 0.$$

where the sum is defined by the addition in F_2^n (and in F_2^m), i.e., the bitwise exclusive-or.

Statement

Let F be a function from F_2^n into F_2^m and V the subspace of dimension $(degF) + 1$. Then, for every $V' \in \underline{V}$ is true:

$$\bigoplus_{v \in V'} F(x \oplus v) = 0.$$

The algorithm for finding zero sums for an arbitrary transformation F

Substitution F

$$F = \underbrace{R_{n_r} \circ R_{n_r-1} \circ \ldots \circ R_{n_r-r_1+1}}_{G_{r_1}} \circ \underbrace{R_{r_2} \circ \ldots \circ R_2 \circ R_1}_{G_{r_2}^{-1}}$$

Algorithm

Let $d_1 < n$ be the degree of the function composed of the last r_1 transformations, i.e., $G_{r_1} = R_{n_r} \circ \ldots \circ R_{n_r-r_1+1}$ and let $d_2 < n$ be the degree of the inverse of the first $r_2 = (n_r - r_1)$ transformations, i.e., $G_{r_2} = R_1^{-1} \circ \ldots \circ R_{r_2}^{-1}$. Then, we can find many zero-sums of size 2^{d+1} where $d = max(d_1, d_2)$ as follows:

1. Choose a set of $(n - d - 1)$ bits in the intermediate state after r_2 rounds, and fix them to an arbitrary value;
2. For each of the 2^{d+1} possible intermediate states z obtained when the other $(d + 1)$ bits take all possible values, compute r_2 rounds backwards in order to obtain the 2^{d+1} input states $x = G_{r_2}(z)$.

Modified algorithm for finding zero sums

- The general idea is to choose a subspace by the following way: it should remains a subspace after a transformation by some internal substitution.

Modified algorithm for finding zero sums

Definition

Let $V_1 \in F_2^n$ and $V_2 \in F_2^n$ are subspaces. We say that $H : F_2^n \to F_2^n$ keeps the structure of the subspace V_1 if for any $V_1^{'} \in \underline{V_1}$ there is a $V_2^{'} \in \underline{V_2}$ such that:

$$H(V_1^{'}) = V_2^{'} \text{ and denote } H(V_1 \to V_2).$$

Definition

We say that a subspace $V = V_1$ of block type $i_1, i_2, \ldots, i_t$ is consistent with the partition $(n_1, \ldots, n_t)$, if $V = \{(a_{n_1}, a_{n_2}, \ldots, a_{n_t})\}$, where if $i_j = 1$ then a_{n_j} takes all possible values from $F_2^{n_j}$, and if $i_j = 0$ then $a_{n_j} = 0$. Denote such subspaces as $V^{(n_1,\ldots,n_t)}(i_1, \ldots, i_t)$ and if the partition has the form (d), then $V^d(i_1, \ldots, i_t)$.

Modified algorithm for finding zero sums

Definition

We say that a transformation $G : F_2^n \to F_2^n$ has a block structure if there exists at least one l subspace of block type V consistent to the partition $(n_1, \ldots, n_t)$ such that:

$G(V \to V')$, where V' is a subspace of block type consistent to the partition $(m_1, \ldots, m_l)$.

Modified algorithm for finding zero sums

Statement 2

Let $G : F_2^n \to F_2^n$ be an iterative transformation of the form:

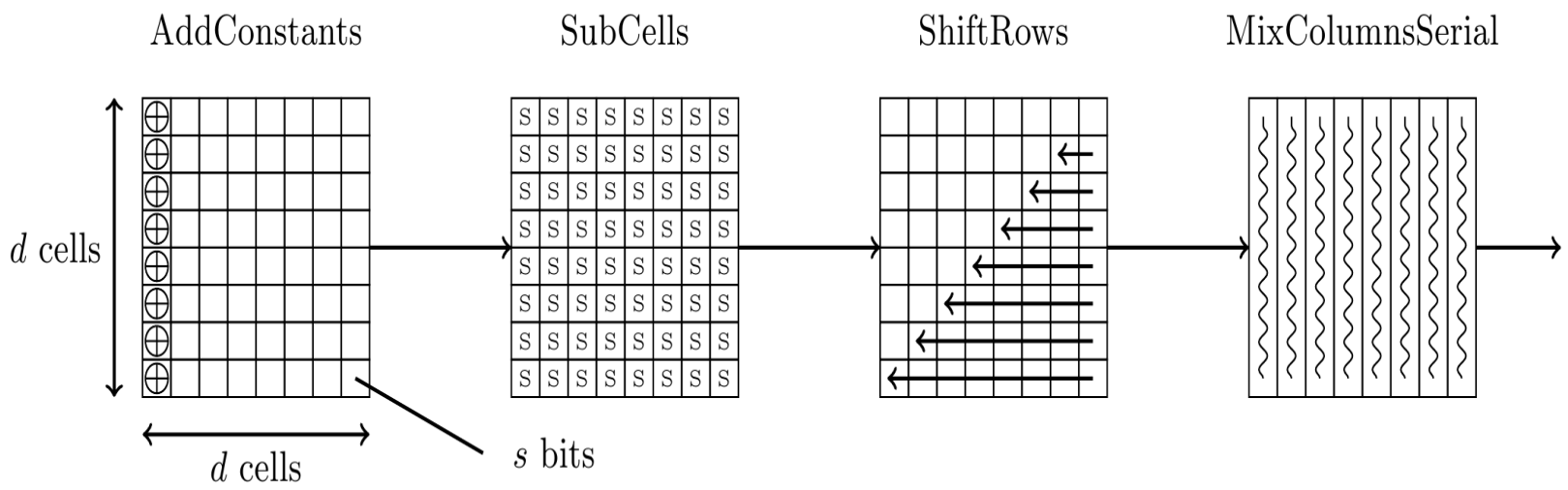
$$G = G_1 \circ G_2 \circ \ldots \circ G_{2 \cdot t}, \text{ where } G_i = X_i \circ S \circ L,\ i = \overline{1, 2 \cdot t}.$$

Then if $V \in F_2^n$ is a subspace such that:
$S \circ L \circ X \circ S(V \to V^{'})$
then in order to find the zero-sum it is necessary to construct a subspace of smaller dimension.
Let us show this. Denote by nl an algebraic degree of the function. Let:

$$\begin{gathered} h_1 = nl(G_1 \circ \ldots \circ G_t), \\ h_2 = nl(G_{t+1} \circ \ldots \circ G_{2 \cdot t}), \\ h_1^{'} = nl(G_1 \circ \ldots \circ G_{t-1}), \\ h_2^{'} = nl(G_{t+2} \circ \ldots \circ G_{2 \cdot t}), \\ d = max(h_1, h_2), \\ d^{'} = max(h_1^{'}, h_2^{'}). \end{gathered}$$

Using the algorithm for finding zero-sums from Section 1, it is easy to see, that the complexity of constructing the zero-sum became $2^{d^{'}}$, instead of 2^d.

- *Photon*, a family of sponge-like hash function proposals that was recently standardized by ISO. Authors define an AES-like function to be a fixed key permutation P applied on an internal state of d^2 elements of s bits each, which can be represented as a $(d \times d)$ matrix. P is composed of 12 rounds, each containing four layers: AddConstants (Add), SubCells (S), ShiftRows (Row), and MixColumnsSerial(Mix).



The limits of the degrees of nonlinearity as a function of the number of rounds

number of rounds	1	2	3	4	5	6	7	8	9
P_{100}	3	9	27	75	91	97	99	99	99
P_{144}	3	9	27	81	123	137	141	143	143
P_{196}	3	9	27	81	157	183	191	194	195
P_{256}	3	9	27	81	197	236	249	253	255
P_{288}	7	42	252	282	287	287	287	287	287

Modified algorithm for finding zero sums

Photon transformation

$$P = G_1 \cdot G_2 \cdot G_3 \cdot G_4 \cdot G_5 \cdot Add_6 \cdot S \cdot Row \cdot Mix \cdot Add_7 \cdot S \cdot Row \cdot Mix \cdot G_8 \cdot G_9 \cdot G_{10} \cdot G_{11} \cdot G_{12}$$

In Figure 1, white color indicates 4-bit words which take all possible values, and black color 4-bit words whose values are fixed.

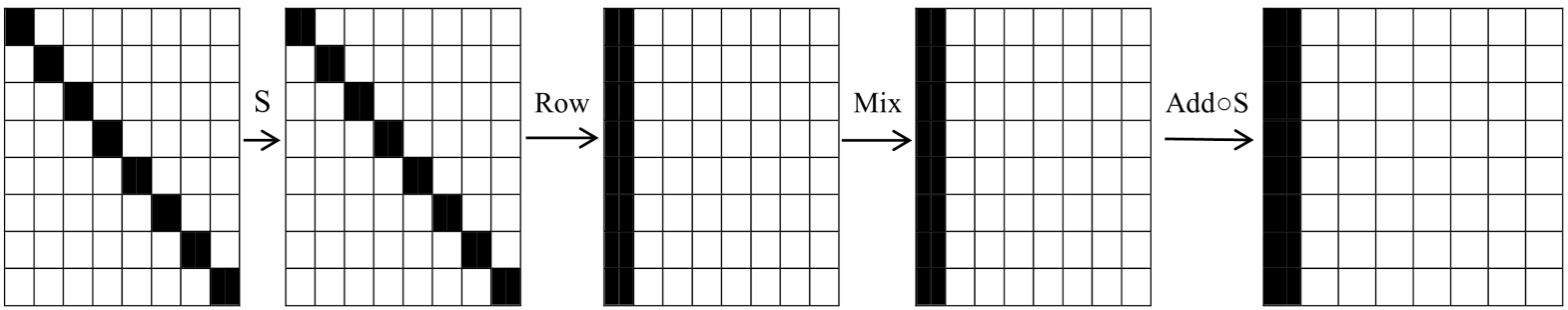


Figure 1

Modified algorithm for finding zero sums

We showed that substutition $H = S \circ Row \circ Mix \circ Add_7 \circ S$ has a block structure such that:

$$H(V_1 \to V_2).$$

$V_1 = (V^4(i_0, \ldots, i_{48})$, where $i_j = 0$, if $(j = 0 \mod (8))$, $i_j = 1$, if $(j \neq 0 \mod (8))$ $j = \overline{0, 48}$.

$V_2 = (V^4(i_0, \ldots, i_{48})$, where $i_j = 0$, if $(j = 0 \mod (7))$, $i_j = 1$, if $(j \neq 0 \mod (7))$ $j = \overline{0, 48}$.

According to Table 2, the degree of nonlinearity
$nl(G_1 \circ \ldots \circ G_6) = nl(G_7 \circ \ldots \circ G_{12}) = 183$, and
$nl(G_1 \circ \ldots \circ G_5) = nl(G_8 \circ \ldots \circ G_{12}) = 157$.

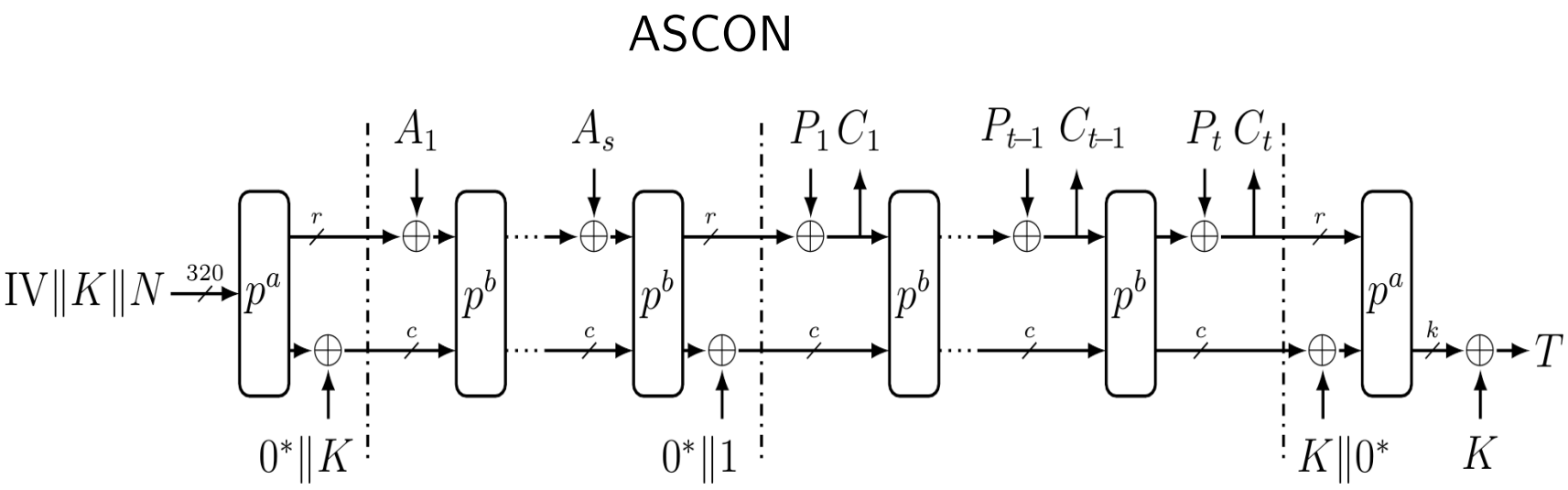
Modified algorithm for finding zero sums

Result

- Using statement 2 and the fact that the dimension of the subspace V_1 is $196 - 4 \cdot 7 = 168$, one can assert that we have found zero-sums of dimension 168 instead of the declared 183 for *Photon*(196).
- Also for the hash function *Photon*(256) using this algorithm, the complexity of finding zero sums can be reduced from 2^{236} to 2^{224}.

Zero-sum application

1. For non-key primitives – distinguishing property.
2. For key primitives – possible weaknesses.


ASCON
A_1
A_s
$P_1\,C_1$
$P_{t-1}\,C_{t-1}$
$P_t\,C_t$
$\mathrm{IV}\|K\|N$
320
p^a
p^b
r
c
k
$0^*\|K$
$0^*\|1$
$K\|0^*$
K
T

Ascon. The permutations.

The main components of Ascon are two 320-bit permutations p^a and p^b (used during data processing). The permutations iteratively apply an SPN-based round transformation p that in turn consists of three subtransformations p_C, p_S and p_L:

$$p = p_L \circ p_S \circ p_C$$

Ascon. Substitution layer.

In the substitution layer p_S, 64 parallel applications of the 5-bit $S-box$ $S(x)$ are performed on the 320-bit state. As illustrated in Figure 2, the $S-box$ is applied to each bit-slice of the five registers $x_0 \ldots x_4$.

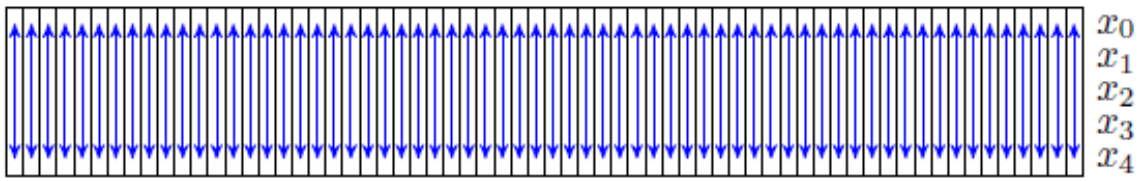


Figure 2.

Ascon. Linear diffusion layer.

The linear diffusion layer p_L for Asconis used to provide diffusion within each of the five 64-bit register words x_i of the 320-bit state S, as illustrated in Figure 3. We apply a linear function $\sum_0(x_0), \ldots, \sum_4(x_4)$ to each word x_i separately,

$$
\begin{aligned}
\Sigma_0(x_0) &= x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28) \\
\Sigma_1(x_1) &= x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39) \\
\Sigma_2(x_2) &= x_2 \oplus (x_2 \ggg 1) \oplus (x_2 \ggg 6) \\
\Sigma_3(x_3) &= x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17) \\
\Sigma_4(x_4) &= x_4 \oplus (x_4 \ggg 7) \oplus (x_4 \ggg 41)
\end{aligned}
$$

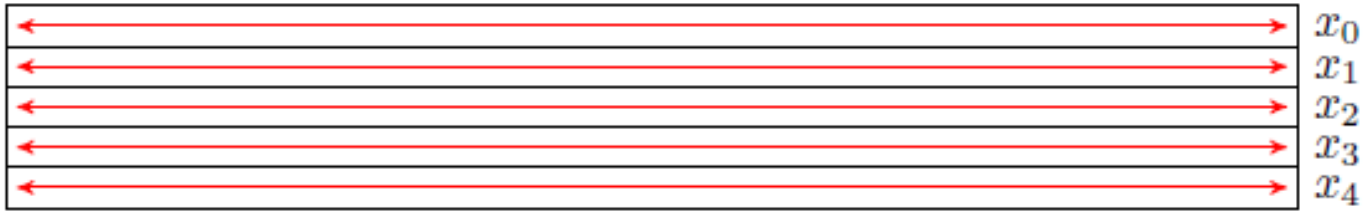


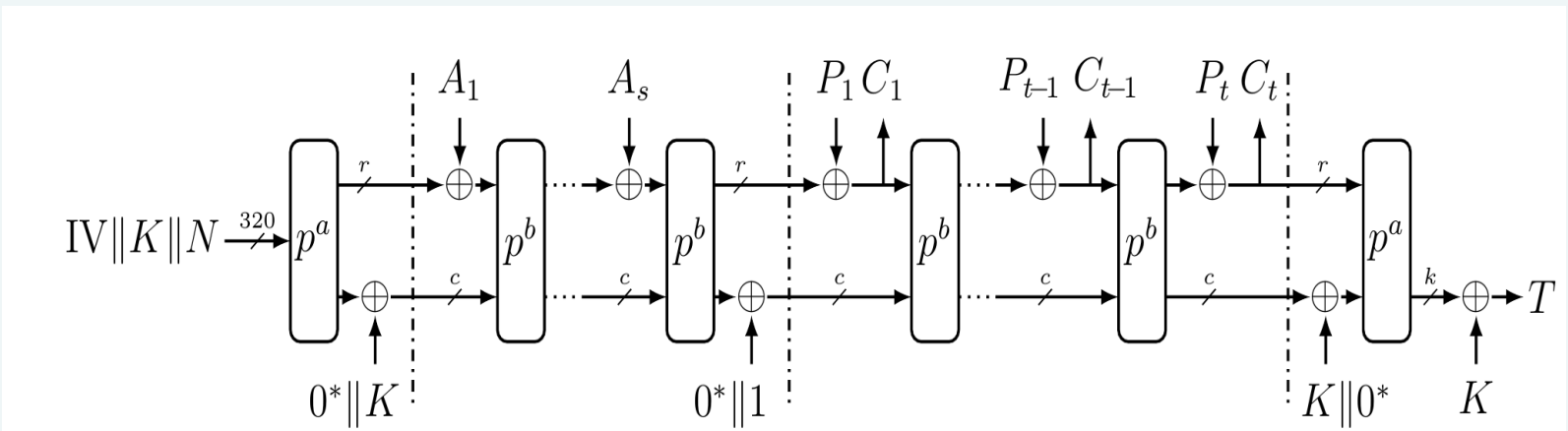
Figure 3.

Ascon. Zero-sum application

Let the input of the Ascon algorithm is a sequence of texts that have the following form:

$$A_1, A_2, \ldots, A_s, P_{1.i}, P_2, \ldots, P_n,$$

then we will know every 2^{32} blocks of text $C_{2.i} \oplus P_{2.i}$.



Thank you for attention!

Questions?