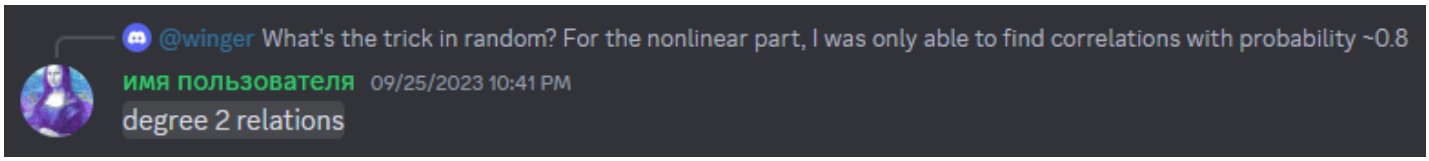
Hyper512 - crypto

tl;dr:



There are 4 LFSRs X, Y, Z, W with 128-bit state, with known taps and unknown keys k_x, k_y, k_z, k_w. The output from these 4 LFSRs are processed with a non-linearity involving sha256 to give one bit of output T. This is equivalent to the following logic table:

xi	yi	zi	wi	Ti
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

This is equivalent to

$$(x_i * y_i * z_i) \oplus (x_i * y_i * w_i) \oplus (x_i * z_i * w_i) \oplus (y_i * w_i) \oplus y_i \oplus z_i = T_i$$

We have 2^{15} samples of continuous bits of the stream output followed by the flag XOR-ed with subsequent stream output. An additional 56 samples can be acquired by crib dragging the flag format. From this, we need to recover the stream output that was XOR-ed with the flag.

Y, Z

If we restrict to cases where $T_i = 1$, we get the following relation which only involves quadratic terms.

$$(y_i * z_i) \oplus y_i \oplus z_i = 1$$

Each y_i, z_i can also be expressed as a sum of bits in k_y, k_z. Because in $\mathbb{F}_2$ multiplication distributes over XOR, we can model the $y_i * z_i$ term as the XOR of quadratic terms in $\{k_{y,i}\} \times \{k_{z,j}\}$. For example, for the 132-th (0-indexed) output bit:

$$\begin{aligned} y_{132} &= k_{y,1} \oplus k_{y,6} \oplus k_{y,10} \oplus \dots \\ z_{132} &= k_{z,0} \oplus k_{y,1} \oplus k_{y,2} \oplus \dots \\ (y_{132} * z_{132}) &= (k_{y,1} * k_{z,0}) \oplus (k_{y,1} * k_{z,1}) \oplus (k_{y,1} * k_{z,2}) \dots \\ &\quad (k_{y,6} * k_{z,0}) \oplus (k_{y,6} * k_{z,1}) \oplus (k_{y,6} * k_{z,2}) \dots \\ &\quad (k_{y,10} * k_{z,0}) \oplus (k_{y,10} * k_{z,1}) \oplus (k_{y,10} * k_{z,2}) \dots \\ &\quad \dots \end{aligned}$$

We now have a way to make relations involving $128 \times 128 = 16384$ possible quadratic terms $k_{y,i} * k_{z,j}$ to model $(y_i * z_i)$. There are only 16368 samples of $T_i = 1$ in the "gift", but using the flag format we can increase this to 16399 samples which is sufficient.

The next hurdle is that we still need a way to model $\oplus\, y_i \oplus z_i$. If we introduce degree-one terms of y_i, z_i, this is an additional 256 unknowns which won't work within our limited number of relations. We can work around this by guessing that some bits of $k_{y,I}$ and $k_{z,J}$ are 1, then using the $k_{y,i} = k_{y,i} * k_{z,J}$ and $k_{z,j} = k_{y,I} * k_{z,j}$ terms to represent the degree-one unknowns. There is a 1/4 chance of succeeding with random I, J and we can keep choosing new values until a solution is found.

We now have 16384 unknowns and 16399 linear relations in $\mathbb{F}_2$ them which can be solved efficiently, yielding the original values of keys k_y, k_z.

W, X

After obtaining the keys k_y, k_z, we can simulate the outputs of y_i, z_i locally. For each sample, we may be able to learn information of x_i, w_i. Specificallly, when $T_i = 0$ we can find the relation:

$$y_i = 1, z_i = 0 \implies w_i = 1$$

By checking for the above condition and expressing w_i in terms of $k_{w,i}$, we can solve for k_w using 128 (full-rank) relations.

Finally, we can simulate the full values of w_i and directly determine the values of x_i, after which another 128 relations are needed to solve for k_x.