Distinguishing Attack on Stream Cipher Yamb

Hongjun Wu and Bart Preneel

Katholieke Universiteit Leuven, Dept. ESAT/COSIC
{wu.hongjun,bart.preneel}@esat.kuleuven.be

Abstract. Stream cipher Yamb is a submission to the ECRYPT stream cipher project. In this paper, we point out that the keystream generated from Yamb can be distinguished from random with about 2^{58} outputs. Only about 2^{55} simple operations are needed in the attack.

1 Stream Cipher Yamb

Brief Description of Yamb [2]. The overall structure of Yamb is given in Fig. 1. Yamb consists of three components: LFSR L of length 15 over $GF(2^{32})$; nonlinear function F_M with 32-bit input and 32-bit output, and a secret table M with 256 8-bit elements is used in F_M; shift register R of length 16 over $Z_{2^{32}}$. At each step, the LFSR L is regularly clocked, one element Y of L is passed as input to F_M, the M is updated with Y, then Y is xored with the output of F_M and the result is passed as input to R. The input and output of R are xored as the keystream.

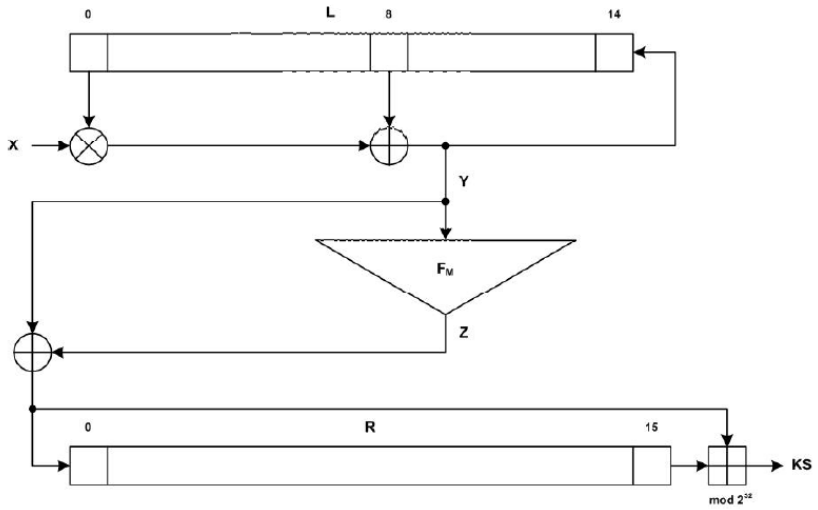

Fig. 1. Overall structure of Yamb [2]

2 Distinguishing Attack on Stream Cipher Yamb

We first focus on the randomness of $Y \oplus Z$, where Y and Z are shown in Fig. 1, and $Z = F_M(Y)$. As given in [2], the function Y_M with four inputs bytes (a, b, c, d) operate as follows:

$$
\begin{array}{l}
b := b \oplus M(a),\ M(a) := M(a) + d \bmod 256 \\
c := c \oplus M(d),\ M(d) := M(d) + b \bmod 256 \\
a := a \oplus M(b),\ M(b) := M(b) + c \bmod 256 \\
d := d \oplus M(c),\ M(c) := M(c) + a \bmod 256 \\
c := c \oplus M(a),\ M(a) := M(a) + d \bmod 256 \\
b := b \oplus M(d),\ M(d) := M(d) + c \bmod 256 \\
a := a \oplus M(c),\ M(c) := M(c) + b \bmod 256 \\
d := d \oplus M(b),\ M(b) := M(b) + a \bmod 256 \\
b := b \oplus M(a),\ M(a) := M(a) + d \bmod 256 \\
c := c \oplus M(d),\ M(d) := M(d) + b \bmod 256 \\
a := a \oplus M(b),\ M(b) := M(b) + c \bmod 256 \\
d := d \oplus M(c),\ M(c) := M(c) + a \bmod 256
\end{array}
$$

The main feature of Yamb is that the S-box M is continuously updated during the keystream generation.

We notice that when the input to F_M is with value 0 ($Y = 0$), and $M(0) = 0$, then $Y \oplus F_M = 0$. This case would happen with probability $2^{-32} \times 2^{-8}$, where 2^{-32} is the probability that $Y = 0$, and 2^{-8} is the probability that $M(0) = 0$. We assume that for $Y \neq 0$, the value of $F_M(Y)$ is randomly distributed. Based on this observation and assumption, we estimate that the value of $Y \oplus F_M$ is 0 with probability $(1 - 2^{-32}) \times 2^{-32} + 2^{-32} \times 2^{-8} \approx 2^{-32} + 2^{-40}$. Thus $Y \oplus F_M(Y)$ can be distinguished from random with about 2^{50} values with success rate 0.84 (the negative and positive false rates are 0.16).

Experiment. We performed the experiment with the reduced version of Yamb. In this reduced version, M is with 32 20-bit elements. We generate a number of random 20-bit Y. According to our analysis above, the value $Y \oplus F_M(Y)$ would be 0 with probability $2^{-20} + 2^{-25}$. In the experiment, we generated 2^{34} outputs and there are 17337 cases that the values of $Y \oplus F_M(Y)$ are 0. The experiment result shows that $Y \oplus F_M(Y) = 0$ with probability about $2^{-20} + 2^{-24.1}$, the bias is larger than the result of our theoretical analysis, i.e., the attack would be more efficient than that is shown in theory.

We now analyze the randomness of the keystream. Denote the keystream output at the i-th step as S_i, and denote $\alpha_i = Y_i \oplus Z_i$. We know that $S_i = \alpha_i + \alpha_{i-16}$. A non-optimized direct attack is to try all the possible values of α_{-16}, then compute α_{16i} from S_{16i} for $i = 0, 1, 2, 3, ...$ In the attack, we need to try 2^{32} values of α_{-16}, so the false negative rate of distinguishing $Y \oplus F_M(Y)$ should be smaller than 2^{-32}. If $2^{53.8}$ samples are used in the experiment, the

overall success rate of the attack is 0.84 (the false positive rate is 0.16, and the false negative rate is 0.17). Thus the attack requires the Yamb stream cipher to generate $16 \times 2^{53.8} = 2^{57.8}$ 32-bit outputs.

Then we give below an efficient way to implement the above attack. We express α_{16i} in terms of S_i and α_{-16} as follows:

$$
\begin{aligned}
&\alpha_0 = S_0 - \alpha_{-16} \\
&\alpha_{16} = S_{16} - \alpha_0 = S_{16} - S_0 + \alpha_{-16} \\
&\alpha_{32} = S_{32} - \alpha_{16} = S_{32} - S_{16} + S_0 - \alpha_{-16} \\
&\alpha_{48} = S_{48} - \alpha_{32} = S_{48} - S_{32} + S_{16} - S_0 + \alpha_{-16} \\
&\cdots\cdots \\
&\alpha_{2i\times 16} = \textstyle\sum_{j=0}^{i} S_{32j} - \sum_{j=1}^{i} S_{32j-16} - \alpha_{-16} \\
&\alpha_{(2i+1)\times 16} = \textstyle\sum_{j=0}^{i} S_{32j+16} - \sum_{j=0}^{i} S_{32j} + \alpha_{-16}
\end{aligned}
$$

Thus we can express α_i as $\alpha_{32i} = \beta_{32i} - \alpha_{-16}$, $\alpha_{32i+16} = \beta_{32i+16} + \alpha_{-16}$, where β_i are computed from S_i. For α_{32i}, we generate a table U, where each element $U[k]$ is the number of β_{32i} that satisfies $\beta_{32i} = k$. Similarly, another table V is generated for α_{32i+16}, where each element $V[k]$ is the number of β_{32i+16} that satisfies $\beta_{32i+16} = k$. We then find out the maximum value of $(U[k] + V[2^{32} - k])$ for $0 \leq k < 2^{32}$. We regard this maximum number as the number of α_{16i} that is with value 0.

To distinguish a keystream with $2^{57.8}$ 32-bit outputs, we need $2^{53.8}$ subtractions to compute all the β_{16i} if we perform the computation in the recursive way. We also need $2^{53.8}$ additions to compute the tables U and V. Finding the maximum value of $(U[k] + V[2^{32} - k])$ requires about 2^{32} additions and comparisons. Thus the attack to distinguish the keystream of Yamb requires about $2^{54.8}$ simple operations (32-bit additions or subtractions).

Remarks. We predict that the amount of keystream required in the attack could be reduced to below $2^{57.8}$ if we make full use of the complete keystream (instead of considering only the S_{16i}).

3 Conclusion

It was shown in this paper that the keystream generated from Yamb can be distinguished from random with about $2^{57.8}$ outputs and with about $2^{54.8}$ simple operations.

Babbage has discovered a weakness in the key schedule of Yamb [1]. It shows that if the same kye is used with IVs of different lengths, then the same keystream would be generated for some chosen IVs.

References

1. Steve Babbage. "Equivalent IVs in Yamb". Available at http://www.ecrypt.eu.org/stream/phorum/read.php?1,11
2. Starodubtzev Sergey A., Lebedev Anatoly N., Volchkov Alexey A., "Yamb – LAN Crypto Submission to the ECRYPT Stream Cipher Project". *ECRYPT Stream Cipher Project Report 2005/034*. Available at http://www.ecrypt.eu.org/stream/