Cryptanalysis of Stream Cipher COS (2, 128) Mode I

Hongjun Wu and Feng Bao

Laboratories for Information Technology
21 Heng Mui Keng Terrace, Singapore 119613
{hongjun,baofeng}@lit.org.sg

Abstract. Filiol and Fontaine recently proposed a family of stream ciphers named COS. COS is based on nonlinear feedback shift registers and was claimed to be highly secure. Babbage showed that COS (2, 128) Mode II is extremely weak. But Babbage's attack is very expensive to break the COS (2, 128) Mode I (the complexity is around 2^{52}). In this paper, we show that the COS (2, 128) Mode I is very weak. Secret information could be recovered easily with about 2^{16}-bit known plaintext.

1 Introduction

Filiol and Fontaine recently designed a family of stream ciphers called COS [3, 4,5]. The COS (2, 128) is with two 128-bit internal registers. Two versions of COS (2,128) are available: Mode II and the more secure Mode I. In [1], Babbage showed that the COS (2, 128) Mode II is too weak and the secret information could be recovered easily from a short piece of key stream. Babbage's attack also reduced the complexity of the COS (2, 128) Mode I to 2^{64}. In [2], Babbage's improved attack reduced the complexity of the COS (2, 128) Mode I to 2^{52}.

In this paper, we show that the COS (2, 128) Mode I could be broken with a short plaintext in negligible time. In average about 2^{16}-bit known plaintext is required in the attack. The time required is about 15 milliseconds on Pentium IV.

This paper is organized as follows. Section 2 introduces the COS (2, 128) stream cipher. The attack against the COS (2, 128) Mode I is given in Section 3. Section 4 concludes this paper.

2 COS Stream Cipher

We give only a brief introduction to the COS (2, 128). This version of COS cipher is with two 128-bit registers, L_1 and L_2, as the initial states. We will ignore the key setup of COS (since the key setup has no effect on our attack) and only introduce the key stream generation process.

Let $L_1 = L_{10} \parallel L_{11} \parallel L_{12} \parallel L_{13}$, $L_2 = L_{20} \parallel L_{21} \parallel L_{22} \parallel L_{23}$, where $\parallel$ indicates concatenation and each L_{ij} is a 32-bit word. At the ith step, the output key stream is generated as:

L. Batten and J. Seberry (Eds.): ACISP 2002, LNCS 2384, pp. 154–158, 2002.

1. Compute clocking value d.
   a) Compute $m = 2 \times (L_{23} \ \& \ 1) + (L_{13} \ \& \ 1)$ where & is the binary AND operator.
   b) $d = C_m$, where $C_0 = 64$, $C_1 = 65$, $C_2 = 66$, $C_3 = 64$.
2. If i is even, clock L_1 d times; otherwise, clock L_2 d times.
3. Let $H_i = H_{i0} \parallel H_{i1} \parallel H_{i2} \parallel H_{i3}$, where $H_{i0} = L_{20} \oplus L_{12}$, $H_{i1} = L_{21} \oplus L_{13}$, $H_{i2} = L_{22} \oplus L_{10}$, $H_{i3} = L_{23} \oplus L_{11}$.
4. For Mode II, the output for the ith step is given as H_i.
5. For Mode I, compute $j = (L_{13} \oplus L_{23}) \ \& \ 3$, $k = (L_{10} \oplus L_{20}) \gg 30$. If $j = k$, then let $k = j \oplus 1$. The output for the ith step is given as $H_{ij} \parallel H_{ik}$.

Two feedback boolean functions are used, $f9a$ for L_1 and $f9b$ for L_2. They use bits $2, 5, 8, 15, 26, 38, 44, 47, 57$ of L_1 and L_2 as input. These two functions are available at [3].

3 Cryptanalysis of COS

In this section we show that the COS (2, 128) Mode I is very weak. Subsection 3.1 gives a brief introduction to our attack while the detailed attack is given in Subsection 3.2. The experiment result is given in Subsection 3.3.

3.1 The Basic Idea of Our Attack

Let us take a look at any four consequent steps starting with an odd step. L_1 is clocked at the second and forth steps; L_2 is clocked at the first and third steps.

	Step 1	Step 2	Step 3	Step 4
L_1	$b \mid a$	$c \mid b'$	$c \mid b'$	$d \mid c'$
L_2	$x \mid w$	$x \mid w$	$y \mid x'$	$y \mid x'$

Fig. 1. Four Steps (starting with an odd step) of COS (2, 128)

In Fig. 1, a, b, b', c, c' and d are 64-bit words of L_1 at the end of a step; w, x, x', y are 64-bit words of L_2 at the end of a step. According to the key stream generation process, b' may be the same as b; or b' may be obtained by right shifting b one (or two) bit position and with the most significant one (or two) bit of b being filled with unknown value. The same applies to c' and c.

The value of (c, b') could be recovered if the following two conditions are satisfied:

Condition 1. The outputs at the first, second, third and forth steps are given as $b \oplus w$, $c \oplus w$, $b' \oplus y$ and $c' \oplus y$ respectively, i.e., (j, k) is $(2, 3)$ or $(3, 2)$ at Step 1 and Step 2 and $(1, 0)$ or $(0, 1)$ at Step 3 and Step 4.

Condition 2. One of b' and c' is not the same as b and c respectively, and b' and c' are not obtained by right shifting b and c (respectively) by the same position.

From Condition 1, we could obtain the values of $b \oplus c$ and $b' \oplus c'$ from the output key stream of these four steps. Once Condition 1 and Condition 2 are satisfied, it is trivial to compute (c, b').

In the next subsection, we will illustrate the idea above in detail and give the estimated results.

3.2 The Detailed Attack

Before introducing the attack in detail, we give the following two observations:

Observation 1. For the COS $(2, 128)$ Mode I, at each step the probability that (j, k) is $(2, 3)$ is 2^{-3}. The same probability holds for (j, k) being $(3, 2)$, $(1, 0)$, $(0, 1)$.

Observation 2. At the ith step, if j is 0 or 2, the clocking value at the next step is 64. If j is 1 or 3, the clocking value at the next step is 65 or 66.

These two observations are trivial according to the specifications of the COS cipher.

We now list in Table 1 those 16 cases that satisfy Condition 1. According to Observation 1, each case appears with probability 2^{-12}.

Table 1. (j, k) values for those 16 cases satisfying Condition 1

	Step 1	Step 2	Step 3	Step 4
Case 1	(2,3)	(2,3)	(0,1)	(0,1)
Case 2	(2,3)	(2,3)	(0,1)	(1,0)
Case 3	(2,3)	(2,3)	(1,0)	(0,1)
Case 4	(2,3)	(2,3)	(1,0)	(1,0)
Case 5	(2,3)	(3,2)	(0,1)	(0,1)
Case 6	(2,3)	(3,2)	(0,1)	(1,0)
Case 7	(2,3)	(3,2)	(1,0)	(0,1)
Case 8	(2,3)	(3,2)	(1,0)	(1,0)
Case 9	(3,2)	(2,3)	(0,1)	(0,1)
Case 10	(3,2)	(2,3)	(0,1)	(1,0)
Case 11	(3,2)	(2,3)	(1,0)	(0,1)
Case 12	(3,2)	(2,3)	(1,0)	(1,0)
Case 13	(3,2)	(3,2)	(0,1)	(0,1)
Case 14	(3,2)	(3,2)	(0,1)	(1,0)
Case 15	(3,2)	(3,2)	(1,0)	(0,1)
Case 16	(3,2)	(3,2)	(1,0)	(1,0)

However, not all those 16 cases satisfy Condition 2. According to Observation 2, Cases 1, 2, 5, 6 do not satisfy Condition 2 since $b' = b$ and $c' = c$; Cases 11, 12, 15, 16 satisfy Condition 2 with probability 0.5; the other eight cases all satisfy Condition 2. Thus for every four steps starting with an odd step, Conditions 1

and 2 are satisfied with probability $10 \times 2^{-12} \approx 2^{-8.7}$. To determine the value of L_1, this attack requires the output of about 820 steps in average.

In the following we estimate the amount of (c, b') being produced in each case, and show how to filter the wrong values of (c, b'). We illustrate Case 4 as an example: at Step 2 L_1 is clocked for 64 times ($b = b'$); at Step 4 L_1 is clocked 65 or 66 steps. So 6 values of (c, b') are generated for every four steps starting with an odd step. For each pair of (c, b'), the values of w and y of L_2 could be obtained. Since L_2 is clocked for only 64 times at Step 3, the 7 least significant bits of y are generated from w. So the wrong (c, b') could pass this filtering process with probability 2^{-7}.

The further filtering is carried out at Step 5 and Step 6. Let $d = d_0 \parallel d_1$, $c = c'_0 \parallel c'_1$, $e = e_0 \parallel e_1$, $d' = d'_0 \parallel d'_1$, $z = z_0 \parallel z_1$, $y = y'_0 \parallel y'_1$ where d_0, d_1, c'_0, c'_1, e_0, e_1, d'_0, d'_1, z_0, z_1, y'_0 and y'_1 are 32-bit words. In Fig. 2, for each (c, b'), there are 6 values for (d, c', e, d') (L_1 is clocked 65 or 66 times at Step 4 and is clocked 64 or 65 or 66 times at Step 6). The L_2 is clocked 65 or 66 times at Step 5, so there are 6 possible values for y'. Now if any one of j or k is equal to 2 or 3 in Sep 4 or 5, then for the right (c, b'), at least one of $d_0 \oplus y'_0$, $d_1 \oplus y'_1$, $e_0 \oplus y'_0$ and $e_1 \oplus y'_1$ appears in the output. Otherwise, j and k could only be 0 or 1 at Step 4 and Step 5, the output of Step 5 is $(c'_0 \oplus z_0) \parallel (c'_1 \oplus z_1)$ or $(c'_1 \oplus z_1) \parallel (c'_0 \oplus z_0)$, that of Step 6 is $(d'_0 \oplus z_0) \parallel (d'_1 \oplus z_1)$ or $(d'_1 \oplus z_1) \parallel (d'_0 \oplus z_0)$. By xoring the outputs of Step 5 and 6 (taking into the considering whether (j, k) is $(1, 0)$ or $(0, 1)$), the right (c, b') should generate $c'_0 \oplus d'_0$ and $c'_1 \oplus d'_1$. The wrong (c, b') could pass this filtering process with probability $6 \times 6 \times 8 \times 2^{-32} \approx 2^{-23.8}$.

	Step1	Step2	Step3	Step4	Step5	Step6
L_1	$b \mid a$	$c \mid b'$	$c \mid b'$	$d \mid c'$	$d_0 \parallel d_1 \mid c'_0 \parallel c'_1$	$e_0 \parallel e_1 \mid d'_0 \parallel d'_1$
L_2	$x \mid w$	$x \mid w$	$y \mid x'$	$y \mid x'$	$z_0 \parallel z_1 \mid y'_0 \parallel y'_1$	$z_0 \parallel z_1 \mid y'_0 \parallel y'_1$

Fig. 2. The 6 Steps (starting with an odd step) of COS (2, 128)

So for every 4 steps starting with an odd step, a correct (c, b') is generated with probability 2^{-12} and a wrong (c, b') is generated with probability $6 \times 2^{-7} \times 2^{-23.8} \approx 2^{-28.2}$.

We list in Table 2 the probabilities that right and wrong (c, b') are generated for any 4 steps starting with an odd step.

So for any 4 steps starting with an odd step, a correct (c, b') is generated with probability $2^{-8.7}$ and a wrong one is generated with probability $2^{-23.6}$. It is obvious that only the correct (c, b') could pass the filtering process. Once (c, b') is determined, it is easy to recover L_2 from the values of w and y.

3.3 Experiment Result

We implemented an attack that uses only the Case 4. In average, our program recovers L_1 in about 15 milliseconds on PC (Pentium IV 1.7GHz) with the outputs of about 2^{13} steps. The COS program provided by the COS designers [3,4] is used in our experiment.

Table 2. The probabilities that correct and wrong L_1 being generated

	Correct L_1 Prob.	Wrong L_1 Prob.
Case 1	0	–
Case 2	0	–
Case 3	2^{-12}	$2^{-30.8}$
Case 4	2^{-12}	$2^{-28.2}$
Case 5	0	–
Case 6	0	–
Case 7	2^{-12}	$2^{-28.2}$
Case 8	2^{-12}	$2^{-25.6}$
Case 9	2^{-12}	$2^{-31.8}$
Case 10	2^{-12}	$2^{-29.2}$
Case 11	2^{-13}	$2^{-30.4}$
Case 12	2^{-13}	$2^{-26.8}$
Case 13	2^{-12}	$2^{-29.2}$
Case 14	2^{-12}	$2^{-26.6}$
Case 15	2^{-13}	$2^{-27.8}$
Case 16	2^{-13}	$2^{-25.2}$

4 Conclusions

In this paper, we showed that the stream cipher COS $(2, 128)$ Mode I is extremely weak and should not be used.

References

1. S.H. Babbage, "The COS Stream Ciphers are Extremely Weak", `http://eprint.iacr.org/2001/078/`
2. S.H. Babbage, "Cryptanalysis of the COS (2,128) Stream Ciphers", `http://eprint.iacr.org/2001/106/`
3. E. Filiol and C. Fontaine, "A New Ultrafast Stream Cipher Design: COS Ciphers", `http://www-rocq.inria.fr/codes/Eric.Filiol/English/COS/COS.html`
4. E. Filiol and C. Fontaine, "A New Ultrafast Stream Cipher Design: COS Ciphers", in *Proceedings of the 8th IMA Conference on Cryptography and Coding*, LNCS 2260, pp. 85-98.
5. E. Filiol, "COS Ciphers are not "extremely weak"! — the Design Rationale of COS Ciphers", `http://eprint.iacr.org/2001/080/`