Cryptanalysis of the COS (2,128) Stream Ciphers

Steve Babbage

Vodafone Group R&D, Newbury, UK
steve.babbage@vodafone.com

Abstract: A new family of very fast stream ciphers called COS (for "crossing over system") has been proposed by Filiol and Fontaine, and seems to have been adopted for at least one commercial standard. COS(2,128) Mode I and COS(2,128) Mode II are particular members of this family for which the authors proposed a cryptanalysis challenge. The ciphers accept secret keys of 256, 192 or 128 bits. In this note we cryptanalyse both of these ciphers, using a small amount of known keystream — with negligible effort in the case of Mode II, and with effort well below that required for a single DES key search in the case of Mode I.

Keywords: COS, stream cipher, nonlinear feedback shift register, cryptanalysis.

1. The COS(2,128) stream ciphers

The COS family of stream ciphers has been introduced by Filiol and Fontaine [1,2]. They are constructed from nonlinear feedback shift registers. COS(n,2L) uses n registers each of length 2L bits, so in particular COS(2,128) uses 2 registers L_1 and L_2 each of length 128 bits. We describe the ciphers in this section.

1.1 Key loading

COS(2,128) uses a secret key of 256, 192 or 128 bits, and a "message key" of 32 bits. (The message key is not assumed to be secret; it is there to allow the same secret key to be used many times without generating the same keystream. It might be a frame counter, for instance.)

A separate nonlinear feedback shift register M, of length 256 bits, is used for the key loading. M is pictured as clocking from left to right. The process is as follows:

- Place the secret key in M. (If the secret key is less than 256 bits long, each remaining byte is derived from one of the secret key bytes by applying a look-up table — see [1,2] for details.)
- XOR the 32-bit message key onto the leftmost 32 bits of M.
- Clock M 256 times.
- Copy the rightmost 128 bits into L_1.
- Clock M 128 more times.
- Copy the leftmost 128 bits into L_2.

M uses a nonlinear feedback function computed on the bits from eleven of its stages.

1.2 Keystream generation

Once the registers L_1 and L_2 have been initialised, the following output generation step is repeated to generate keystream (with i alternating between 1 and 2):

1. Clock L_i 64, 65 or 66 times (the exact number of clocks to be applied is determined by the values of some of the register bits).
2. Generate the following blocks of 64 bits each:

   left-half $(L_1) \oplus$ *right-half* (L_2)

   right-half $(L_1) \oplus$ *left-half* (L_2)

There are two modes of operation: mode II, in which all 128 of these bits are used as keystream each time, and mode I, in which the 128 bits are considered as four words of 32 bits, and only two of the words are used as keystream each time[1] (the ones selected depending on the values of some register bits).

Each of the registers L_1 and L_2 uses a nonlinear feedback function computed on the bits from nine of its stages (a different 9-bit to 1-bit function for each register, although the bit positions used to provide the 9 inputs are the same for each register).

2. Cryptanalysis of Mode II

The analysis here appeared previously in an earlier note [3] by the present author.

2.1 The space of unknowns is trivially reduced to 64 bits

First suppose that the keystream blocks are known for two consecutive steps, and that the number of times the register was clocked between the steps was 64 (this happens with probability ½). Then the contents of the registers look like this, where α, β, γ, δ, ε are 64-bit register halves:

Register that is clocked	α	β	ε	α
Register that is not clocked	γ	δ	γ	δ
Keystream blocks	$\alpha\oplus\delta$ $\beta\oplus\gamma$		$\varepsilon\oplus\delta$ $\alpha\oplus\gamma$	

It is clear that, if any one of α, β, γ, δ or ε is known, then all the others can immediately be recovered, and so the state of both registers can be determined. This is despite the fact that the secret key used to initialise the registers has length 128, 192 or 256 bits.

But it gets much worse than that.

2.2 The effective key size is roughly one bit

Now suppose that the keystream blocks are known for three consecutive steps, that one register was clocked 64 times between the first and second step and that the other register was

[1] Note: an early draft of [2] stated that, in Mode I, two 64-bit blocks (rather than two 32-bit words) were used each time — which implied that, for COS(2,128), there was no difference between Modes I and II. That was the understanding on which [3] was written. However, the published reference code at [1] uses two 32-bit words in Mode I; and private communication with one of the designers makes it clear that this is what is intended.

clocked 65 times between the second and third step (this happens with probability ½ × ¼). Then the contents of the registers look like this, where α, β, γ, δ, ε, ζ are 64-bit register halves:

First register to be clocked	$\boxed{\alpha}\boxed{\beta}$	$\boxed{\varepsilon}\boxed{\alpha}$	$\boxed{\varepsilon}\boxed{\alpha}$
Second register to be clocked	$\boxed{\gamma}\boxed{\delta}$	$\boxed{\gamma}\boxed{\delta}$	$\boxed{\zeta}\boxed{\gamma'}$
Keystream blocks	$\alpha\oplus\delta$ $\beta\oplus\gamma$	$\varepsilon\oplus\delta$ $\alpha\oplus\gamma$	$\varepsilon\oplus\gamma'$ $\alpha\oplus\zeta$

Here γ' means γ shifted to the right by one bit, with an unknown bit appearing in the leftmost position.

Let us number the bits of the register halves, so that for instance α_0 is the rightmost bit of α.

Now guess γ_0.

- We know $\alpha\oplus\gamma$, so we recover α_0.
- We know $\beta\oplus\gamma$, so we recover β_0.
- We know $\alpha\oplus\delta$, so we recover δ_0.
- We know $\varepsilon\oplus\delta$, so we recover ε_0.
- We know $\varepsilon\oplus\gamma'$, so we recover γ_1.
- We know $\alpha\oplus\gamma$, so we recover α_1.
- We know $\beta\oplus\gamma$, so we recover β_1.
- etc etc …
- … so we recover all of α, β, γ, δ and ε. And all we have had to “search” over is the single bit γ_0.

So we have an attack that requires three known keystream blocks and a 1-bit search, and works with probability 1/8. We have implemented this attack using the reference code for COS(2,128) provided at [1], and confirmed that it works as described here.

2.3 Variations of the above attack

The attack described in the previous section requires successive clocking amounts of 64 and 65. In fact it is easy to see that the attack can be generalised to work for any two successive clocking amounts that are not both equal to 64. (You may have to search over two, three or four bits instead of one ….) So, given any three consecutive blocks of known keystream, with probability ¾ there is an attack that recovers the entire state of the registers with negligible effort.

3. Cryptanalysis of Mode I

3.1 The space of unknowns is trivially reduced to not much more than 64 bits

Suppose that the keystream blocks are known for three consecutive steps, and that the number of times the register was clocked between each of the steps was 64 (this happens with probability ½ × ½). Then the contents of the registers look like this, where the a_i and b_i are 32-bit register quarters:

First register to be clocked	a_1	a_0	a_{-1}	a_{-2}	a_3	a_2	a_1	a_0	a_3	a_2	a_1	a_0
Second register to be clocked	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_5	b_4	b_3	b_2

Now suppose that the keystream words (considered as *ordered* pairs) that are selected at each stage are as follows (this happens with probability $2^{-3} \times 2^{-3} \times 2^{-3}$ — see [1,2] for details of how the words are selected):

Keystream blocks	$a_1 \oplus b_1$	$a_1 \oplus b_3$	$a_3 \oplus b_3$
	$a_0 \oplus b_0$	$a_0 \oplus b_2$	$a_2 \oplus b_2$

It is clear that, if any one of a_1, a_3, b_1, or b_3 is known, then all the others can immediately be recovered, and similarly for a_0, a_2, b_0, and b_2. So a 64-bit guess is sufficient to determine the entire state of both registers. This is despite the fact that the secret key used to initialise the registers has length 128, 192 or 256 bits.

3.2 The space of unknowns is further reduced

The attack just described treats all the words a_i and b_i as independent unknowns. But of course they are not all independent: for instance, b_5 and b_4 are entirely determined by b_3, b_2, and b_1 and b_0. By taking advantage of the dependencies we can reduce the complexity of the attack significantly.

Suppose that the keystream blocks are known for four consecutive steps, and that the number of times the register was clocked between each of the steps was 64 (this happens with probability ½ × ½ × ½). Then the contents of the registers look like this, where the a_i and b_i are 32-bit register quarters:

First register to be clocked	a_1	a_0	a_{-1}	a_{-2}	a_3	a_2	a_1	a_0	a_3	a_2	a_1	a_0	a_5	a_4	a_3	a_2
Second register to be clocked	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_5	b_4	b_3	b_2	b_5	b_4	b_3	b_2

Suppose also that the keystream words selected at each stage are as follows (this happens with probability $2^{-3} \times 2^{-3} \times 2^{-3} \times 2^{-3}$):

Keystream blocks	$a_1 \oplus b_1$	$a_1 \oplus b_3$	$a_3 \oplus b_3$	$a_3 \oplus b_5$
	$a_0 \oplus b_0$	$a_0 \oplus b_2$	$a_2 \oplus b_2$	$a_2 \oplus b_4$

By guessing any one bit in any of a_1, a_3, b_1, b_3, or b_5, we immediately learn the corresponding bit in all the others. Similarly for a_0, a_2, b_0, b_2, and b_4. But when we know particular sets of nine bits, we can start to determine others for free, using the dependencies from the register clocking. It turns out that guessing only 37 bits is sufficient to determine the 64 we need. Number the bits of each word from 0 to 31, with bit 0 at the right hand end; by guessing bits 31, 28, 27, 25, 24, 21, 18, 15, 14, 13, 12, 11, 10, 9, 8, 5, 3, 2 and 0 of a_1 and bits 31, 29, 28, 27, 26, 24, 23, 21, 18, 17, 15, 14, 13, 11, 10, 8, 5 and 2 of a_0, we can deduce all of $a_0 \dots a_3$ and $b_0 \dots b_5$, i.e. the entire state of both registers. We deduce other bits by repeatedly XORing known bits with known keystream and applying the nine-bit nonlinear function to particular sets of 9 known bits.

Note: the particular set of 37 bits listed here was found by hand; we have no guarantee that 37 is optimal. It is quite possible that a smaller set might be found. Developing a method to find the smallest possible set efficiently may be an interesting research problem.

3.3 Variations of the above attack

The precise attack described in the previous section requires a 37-bit exhaustive search, and works on any block of 256 known keystream bits with probability $(½)^3 \times (2^{-3})^4 = 2^{-15}$. Given plenty (roughly 2^{21} bits[2]) of known keystream, this of course translates into an attack which can be expected to succeed, and whose expected running time is around 2^{52}.

We described a scenario in which the registers were clocked exactly 64 bits each time. That was for ease of explanation; a similar attack can be pursued no matter what the number of clocks at each stage[3]. We do need to take account of the additional uncertainty as to how many clocks were applied — but this only increases the complexity of the attack by one bit (i.e. by a factor of two), because the bits that determine the clocks are all in the same positions in the registers, so that guessing one clock control bit is sufficient to deduce them all by XORing with known keystream. Applying this variation reduces the known keystream requirement by a factor of 2^3, while increasing the running time by a factor of 2.

There are also many other sets of keystream words that would do us just as well, several of which occur with the same probability as the ones we chose in section 3.2. This allows us to reduce the requirement for known keystream by a factor of around 2^7, without altering the expected running time for the attack[4]. By considering less likely sets of keystream words, we can reduce the amount of known keystream required a little further, at the expense of a modest increase in expected running time.

So applying the variations of the two previous paragraphs reduces the number of keystream bits required to around 2^{11}.

Finally, there is scope for further analysis to see if any advantage can gained by considering slightly longer sequences of known keystream with more dependencies between the unknown bits. It seems likely, though, that the additional uncertainty about the clocking of the registers and the selection of the keystream words will outweigh any reduction in the number of bits that we need to guess.

---

[2] We need about $2^{15} \times 64$ bits (not $2^{15} \times 256$, because the 256-bit blocks we consider can overlap).

[3] The precise analysis of the 37 bits we needed to guess in section 3.2 is no longer appropriate. For simplicity, we have assumed that the number of bits that would need to be guessed remains at 37; more analysis needs to be done here.

[4] Let the register state words be:

First register to be clocked	a_1	a_0	a_{-1}	a_{-2}		a_3	a_2	a_1	a_0		a_3	a_2	a_1	a_0		a_5	a_4	a_3	a_2
Second register to be clocked	b_3	b_2	b_1	b_0		b_3	b_2	b_1	b_0		b_5	b_4	b_3	b_2		b_5	b_4	b_3	b_2

Then any of the following sets of keystream words will do:

$a_1 \oplus b_1$, $a_0 \oplus b_0$,	$a_3 \oplus b_1$, $a_2 \oplus b_0$,	$a_3 \oplus b_3$, $a_2 \oplus b_2$,	$a_5 \oplus b_3$, $a_4 \oplus b_2$
$a_1 \oplus b_1$, $a_0 \oplus b_0$,	$a_3 \oplus b_1$, $a_2 \oplus b_0$,	$a_3 \oplus b_3$, $a_2 \oplus b_2$,	$a_3 \oplus b_5$, $a_2 \oplus b_4$
$a_1 \oplus b_1$, $a_0 \oplus b_0$,	$a_1 \oplus b_3$, $a_0 \oplus b_2$,	$a_3 \oplus b_3$, $a_2 \oplus b_2$,	$a_5 \oplus b_3$, $a_4 \oplus b_2$
$a_1 \oplus b_1$, $a_0 \oplus b_0$,	$a_1 \oplus b_3$, $a_0 \oplus b_2$,	$a_3 \oplus b_3$, $a_2 \oplus b_2$,	$a_3 \oplus b_5$, $a_2 \oplus b_4$
$a_1 \oplus b_1$, $a_0 \oplus b_0$,	$a_1 \oplus b_3$, $a_0 \oplus b_2$,	$a_1 \oplus b_5$, $a_0 \oplus b_4$,	$a_3 \oplus b_5$, $a_2 \oplus b_4$
$a_{-1} \oplus b_3$, $a_{-2} \oplus b_2$,	$a_1 \oplus b_3$, $a_0 \oplus b_2$,	$a_1 \oplus b_5$, $a_0 \oplus b_4$,	$a_3 \oplus b_5$, $a_2 \oplus b_4$
$a_{-1} \oplus b_3$, $a_{-2} \oplus b_2$,	$a_1 \oplus b_3$, $a_0 \oplus b_2$,	$a_3 \oplus b_3$, $a_2 \oplus b_2$,	$a_3 \oplus b_5$, $a_2 \oplus b_4$

and within any pair of keystream words we can take the words in either order.

4. An observation on the key loading

If you use a cipher like this many times with the same secret key and a different message key each time, you would hope that the keystream sequence would be different every time. COS(2,128) does not, in fact, guarantee this; there is a non-trivial probability that two different message keys will give the same keystream sequence.

Consider two message keys that differ only in the rightmost bit; and consider what happens in each case as the register M is clocked, and the single bit difference moves along the register. Every now and again, this differing bit will feature as one of the eleven inputs to the nonlinear feedback function; whether it makes any difference to the output of this function will depend on the values of the other ten input bits.

Number the stages of M 255 to 0, from left to right. Bits 31 to 0 of message key are XORed into stages 255 to 224. Now one of the inputs to the feedback function comes from stage 227; and the feedback function is *linear* in this particular input. Hence a single bit difference in any of bits 31 to 3 of the message key will cause many more differences in the feedback bits, and the two parallel instances of M will quickly diverge. But if the single bit difference is in any of bits 2 to 0, then the story is very different; it is quite possible for this one differing bit to pass along the register, never making any difference to the feedback, before it falls off the right hand end. If this happens, then it does so before any of the contents of M are copied into L_1 or L_2; the two instances of M are now exactly the same, and so the contents of L_1 and L_2 are identical in each case, and the keystream is too.

An immediate estimation suggests that this should occur with probability roughly 2^{-9}, since the difference bit appears as an input to the feedback function 9 times before it falls off the end of the register. Empirical testing suggests that the probability is closer to $2^{-9 \cdot 5}$ for the register M as defined in COS(2,128). In other words, in a multi-frame application where the message key is an incrementing frame counter, out of a few thousand consecutive frames, we can expect some pairs of frames to be encrypted with exactly the same keystream[5]. Depending on the nature of the plaintext, this may well be enough for the attacker to reconstruct some or all of the plaintext information, as is well known.

There is a higher probability that a single bit difference in the message key will leave the resulting contents of just L_1 unaffected; simplistic estimation suggests that this probability should be around 2^{-5}, while empirical testing suggests that it is in fact roughly 1/37. When this happens, it will be quite easy to identify from the keystream, particularly because of a remarkable property that COS(2,128) possesses: there is a significant probability that four consecutive blocks of keystream XOR to all zeroes. Again with significant probability, this allows an attacker to determine the XOR of the states of L_2 in the two frames, hence the XOR of the two keystream sequences, and hence the XOR of the two plaintext sequences. So this is almost as bad as if the two keystream sequences had been the same — except that a small amount of known keystream is required to perform the attack.

5. So does that mean that the ciphers have been broken?

In [3], as described here in section 2, the present author showed how Mode II of COS(2,128) could be broken with negligible effort and a few hundred bits of known keystream. The conclusion was that this cipher is extremely weak.

---

[5] To put this in context: in GSM mobile telephony, 1000 frames corresponds to less than 5 seconds of a phone call.

The COS designers responded in [4], saying in effect that Mode II was intended for enciphering compressed plaintext, that therefore guessing a few hundred bits of plaintext was pretty much equivalent to guessing all the plaintext, and thus that the assumption of even a few hundred bits of known keystream was quite unrealistic. So they still assert that Mode II is suitable for use.

We suggest that this shows a difference between two perspectives on whether or not a cipher is broken:

- The customer's perspective: if there is a non-trivial chance that, if I use this cipher, an eavesdropper could deduce my plaintext, then I should not use this cipher — I should consider it broken.
- The cryptanalyst's perspective: unless I can be *confident* of deducing the plaintext from an intercepted transmission, I am not happy that the cipher has been broken.

Mode II may not have been broken in the second sense: a cryptanalyst cannot be confident of being able to know or guess a few hundred bits of known plaintext[6]. But in the first sense, it certainly has been broken: even if the plaintext has been compressed, it is still very possible that enough of the uncompressed plaintext might be guessed from the context that a few hundred bits of compressed plaintext are known. (If I know the first paragraph of a document, and I apply WinZip to that document, then I will know the first few hundred bits of the resulting file.)

But it is the customer who decides whether or not to use a cipher. From the customer's perspective, this cipher is not suitable for use.

6. Conclusions

Mode II of COS(2,128) is extremely weak. The state of the generator can be recovered with negligible effort, and high probability of success, from a very small amount (a few hundred bits) of known keystream. The same is true of COS(n,2L) for any n and L.

Mode I of COS(2,128) can be broken faster than a single DES key search, given a few thousand bits of known keystream. This is despite its key size of up to 256 bits (and minimum of 128 bits).

We have also demonstrated a major flaw in the key loading process which can often result in two frames of plaintext being encrypted with exactly the same keystream.

7. Scope for further research

More work needs to be done to determine the extent to which the Mode I attack can be generalised to COS(n,2L) for n>2.

The attacks described in sections 2 and 3 break the ciphers in the sense that they recover the initial state of L_1 and L_2. Can we then work back through the key loading process to recover the secret key? Knowing the initial state of either L_1 or L_2 gives 128 bits of the state of M at some point during the key loading, so there a simple 128-bit search allows the recovery of the entire state of M; it is then possible to work back to the initial state of M, and hence to the secret key (although some states have two possible predecessors at each clock of M, the number of possible branches as we clock backwards does not grow significantly). But can we

---

[6] Is it relevant that at least one of the designers comes from a background more like that of a cryptanalyst than like that of a customer? This is idle speculation that has no place in a scientific paper.

do much better than that — perhaps using the observations of section 4 to deduce information about the secret key?

The set of 37 bits used as a “basis” for deducing 64 unknowns in section 3.2 was found by hand; finding the smallest possible set could be a nice self-contained research problem.

8. References

[1] 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

[2] E.Filiol and C.Fontaine, *A New Ultrafast Stream Cipher Design: COS Ciphers*, in Proceedings of the 8th IMA Conference on Cryptography and Coding, 2001 (to appear in the Springer Verlag Lecture Notes in Computer Science series)

[3] S.H.Babbage, *The COS Stream Ciphers are Extremely Weak*, http://eprint.iacr.org/2001/078/

[4] E.Filiol and C.Fontaine, *COS Ciphers are not “extremely weak”! — the Design Rationale of COS Ciphers*, http://eprint.iacr.org/2001/080/