Cryptanalysis of Stream Cipher DECIM

Hongjun Wu and Bart Preneel

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

Abstract. Stream cipher DECIM is a hardware oriented cipher with 80-bit key and 64-bit IV. In this paper, we point out two serious flaws in DECIM. One flaw is in the initialization of DECIM. It causes about half of the key bits being recovered bit-by-bit when one key is used with about 2^{20} random IVs, and only the first two bytes of each keystream are needed in the attack. The amount of computations required in the attack is negligible. Another flaw is in the keystream generation algorithm of DECIM. It causes the keystream heavily biased. Any two adjacent keystream bits would be equal with probability about $\frac{1}{2} + 2^{-9}$. A message could be recovered from the ciphertexts if that message is encrypted by DECIM for about 2^{18} times. The DECIM with 80-bit key and 80-bit IV is also vulnerable to the attacks.

1 Introduction

DECIM [1] is stream cipher submitted to the ECRYPT stream cipher project. In this paper, we point out two flaws in DECIM, one in the initialization, and another one in the keystream generation algorithm. The flaw in the initialization causes the key being easily recovered from the keystreams when one key is used with about 2^{20} random IVs. The flaw in the keystream generation algorithm causes the keystream heavily biased, and thus vulnerable to the broadcast attack.

In Section 2, we illustrate the DECIM cipher. Section 3 presents the key recovery attack on DECIM. The key recovery attack on DECIM is improved in Section 4. The broadcast attack on DECIM is given in Section 5. Section 6 shows that the DECIM with 80-bit IV is also vulnerable to the attacks. Section 7 concludes this paper.

2 Stream Cipher DECIM

The main feature of the stream cipher DECIM is the use of the ABSG decimation mechanism in the keystream generation.

2.1 Keystream Generation

The keystream generation diagram of DECIM is given in Fig. 1. DECIM has a regularly clocked LFSR which is defined by the feedback polynomial

$$P(X) = X^{192} + X^{189} + X^{188} + X^{169} + X^{156} + X^{155} + X^{132} + X^{131} + X^{94} + X^{77} + X^{46} + X^{17} + X^{16} + X^5 + 1$$

over $GF(2)$. The related recursion is given as

$$s_{192+n} = s_{187+n} \oplus s_{176+n} \oplus s_{175+n} \oplus s_{146+n} \oplus s_{115+n} \oplus s_{98+n} \oplus s_{61+n} \\ \oplus s_{60+n} \oplus s_{37+n} \oplus s_{36+n} \oplus s_{23+n} \oplus s_{4+n} \oplus s_{3+n} \oplus s_n$$

At each stage, two bits are generated from the LFSR as follows:

$$y_{t,1} = f(s_{t+1}, s_{t+32}, s_{t+40}, s_{t+101}, s_{t+164}, s_{t+178}, s_{t+187}),$$

$$y_{t,2} = f(s_{t+6}, s_{t+8}, s_{t+60}, s_{t+116}, s_{t+145}, s_{t+181}, s_{t+191}),$$

where the Boolean function f is defined as

$$f(x_{i_1}, ..., x_{i_7}) = \sum_{1 \le j < k \le 7} x_{i_j} x_{i_k}$$

The binary sequence y consists of all the $y_{t,1}$ and $y_{t,2}$ as

$$y = y_{0,1} y_{0,2} y_{1,1} y_{1,2} \cdots y_{t,1} y_{t,2} \cdots$$

The keystream sequence z is generated from the binary sequence y through the ABSG decimation algorithm. The sequence y is split into subsequences of the form $(\bar{b}, b^i, \bar{b})$, with $i \ge 0$ and $b \in \{0, 1\}$; $\bar{b}$ denotes the complement of b in $\{0, 1\}$. For every subsequence $(\bar{b}, b^i, \bar{b})$, the output bit is b for $i = 0$, and $\bar{b}$ otherwise. The ABSG algorithm is given below as

Input: $(y_0, y_1, ...)$
Set: $i \leftarrow 0$; $j \leftarrow 0$;
Repeat the following steps:
    $e \leftarrow y_i$, $z_j \leftarrow y_{i+1}$;
    $i \leftarrow i + 1$;
    while $(y_i = \bar{e})$ $i \leftarrow i + 1$;
    $i \leftarrow i + 1$;
    output z_j
    $j \leftarrow j + 1$

Remarks. The above description of the ABSG and the pseudo code of ABSG are quoted from [1]. However the outputs of the pseudo code are the complements of that of the ABSG algorithm. Anyway, this difference has no effect on the security of DECIM. In the rest of the paper, we assume that the DECIM uses the pseudo code of ABSG given above.

The DECIM is designed to output one bit every two stages. A 32-bit buffer is used to ensure that the probability that there is output bit missing is extremely small (2^{-89}).

2.2 Initialization

The secret key K is a 80-bit key. The 64-bit IV is expanded to a 80-bit length vector by adding zeros from position 64 up to position 79. The initial value of the LFSR state is loaded as follows

$$s_i = \begin{cases} K_i \vee IV_i & \text{for } 0 \leq i \leq 55 \\ K_{i-56} \wedge \overline{IV_{i-56}} & \text{for } 56 \leq i \leq 111 \\ K_{i-112} \oplus IV_{i-112} & \text{for } 112 \leq i \leq 191 \end{cases}$$

The LFSR is clocked 192 times. After the LFSR being clocked linearly at the t-th stage, the $y_{t,1}$ and $y_{t,2}$ are XORed to the $x_{t,192}$ as

$$s_{t+192} = s_{t+192} \oplus y_{t,1} \oplus y_{t,2}$$

Then one of two permutations π_1 and π_2 is applied to permute 7 elements s_{t+5}, s_{t+31}, s_{t+59}, s_{t+100}, s_{t+144}, s_{t+177}, s_{t+186}. Two bits $y_{t,1}$ and $y_{t,2}$ are input to the ABSG, if the output of the ABSG is 1, then π_1 is applied; otherwise the output of the ABSG is 0 or no output, then π_2 is applied. The two permutations are defined as

$$\pi_1 = (1\ 6\ 3)(4\ 5\ 2\ 7), \pi_2 = (1\ 4\ 7\ 3\ 5\ 2\ 6).$$

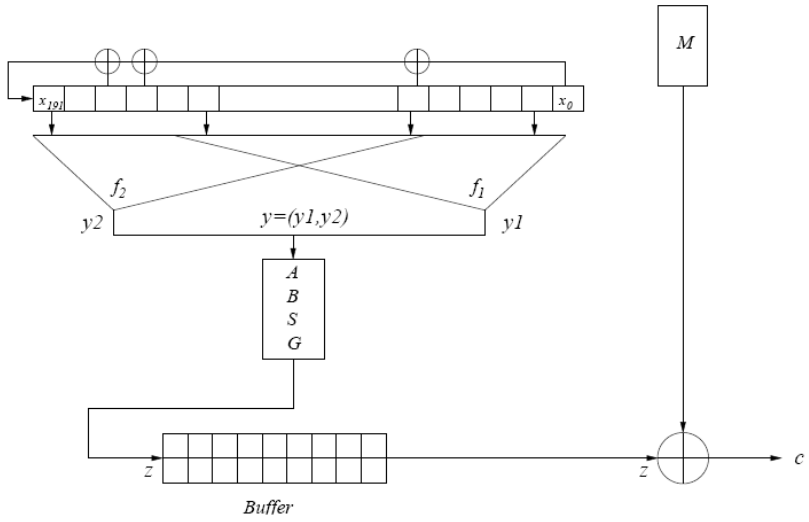

Fig. 1. Keystream Generation Diagram of DECIM [1]

3 Key Recovery Attack on DECIM

In this section, we develop attacks to recover the secret key of DECIM. The attack applies when the same secret key is used with a number of random IVs, and the first 3 bytes of each keystream are known.

3.1 The effects of the permutations π_1 and π_2

The two permutations in the initialization stage of DECIM provide high non-linearity to the initialization process. However, the permutations also cause some bits in the LFSR being updated improperly. The consequence is disastrous.

The permutation π_1 is poorly designed. To investigate the effects of this permutation, we analyze a weak version by assuming that only this permutation is used in the initialization process, i.e., we replace π_2 with π_1. The values of 140 elements in the LFSR would never be updated by the initialization process. Those 140 elements are s_5,s_6,...,s_{58}, and s_{100},s_{101},...,s_{185}. For example, s_{21} would always become s_{192+6}. The details are given below. We trace the bit s_{21}, after 16 steps it becomes s_{16+5} due to the shift of the LFSR. Then it becomes s_{16+177} due to the permutation π_1. After 33 steps, it becomes s_{49+144} due to the shift of the LFSR. Then it becomes s_{49+31} due to the permutation π_1. After 26 steps, it becomes s_{75+5} due to the shift of the LFSR. Then it becomes s_{75+177} due to the permutation π_1. This process repeats and at the end of the initialization process, it becomes s_{192+6}.

The first bit of the keystream is given as $y_{192,2}$, it is computed as $y_{192,2} = f(s_{192+6}, s_{192+8}, s_{192+60}, s_{192+116}, s_{192+145}, s_{192+181}, s_{192+191})$. By tracing the bits of LFSR during the initialization process, we know that $s_{192+6} \Leftarrow s_{21}$, $s_{192+8} \Leftarrow s_{23}$, $s_{192+116} \Leftarrow s_{132}$, $s_{192+145} \Leftarrow s_{160}$, $s_{192+181} \Leftarrow s_{33}$. If every key and IV pair is randomly generated, then according to the loading of the key and IV, we know that s_{21}, s_{23}, and s_{33} are with value 1 with probability 0.75. Thus according to the definition of the function f, the value of $y_{192,2}$ is 0 with probability 0.582. So the first bit of the keystream is heavily biased. It shows that the effect of the permutation π_1 is terrible.

In DECIM, there are two permutations, π_1 and π_2. They are chosen according to the output of ABSG: π_1 is chosen with probability $\frac{1}{3}$, π_2 with probability $\frac{2}{3}$. Due to these two permutations, the number of bits that are not updated by the initialization process is reduced to 54.5 (obtained by running 2^{16} random key and IV pairs). It shows that the permutations π_1 and π_2 which are chosen by the output of ABSG cause severe damage to DECIM.

3.2 Recovering K_{21}

In the initialization process, we trace the bit s_{21}. s_{21} would become s_{192+6} with probability $\frac{1}{27}$. If s_{192+6} is with value 0, and assume all the other bits in the LFSR at the 192-th step are random, then the value of the first bit of the keystream is 0 with probability $q_0 = \frac{56}{128}$. If s_{192+6} is with value 1, and assume all the other bits of the LFSR at the 192-th step are random, then the value

of the first bit of the keystream is 0 with probability $q_1 = \frac{72}{128}$. Denote the probability that the value of the first keystream bit is 0 when $s_{21} = 0$ as p_0, and the probability that the value of the first keystream bit is 0 when $s_{21} = 1$ as p_1. Then $\Delta p = p_1 - p_0 = \frac{1}{27} \times (q_1 - q_0) = 2^{-7.75}$. In the experiment, we carried out 2^{20} initializations with random IVs for $s_{21} = 0$, and another 2^{20} initializations for $s_{21} = 1$, we found that $\Delta p = 2^{-7.99}$. The experiment result confirms that the theoretical result $\Delta p = 2^{-7.75}$ is correct.

The above property can be applied to recover K_{21} as follows. Suppose that the same key is used with N random IVs to generate keystreams. For the keystreams with $IV_{21} = 0$, we compute the probability that the value of the first bit is 0, and denote this probability as p'_0. For the keystreams with $IV_{21} = 1$, we compute the probability that the value of the first bit is 0, and denote this probability as p'_1. If $p'_1 - p'_0 > \frac{\Delta p}{2} = 2^{-8.75}$, we consider that $K_{21} = 0$; otherwise, $K_{21} = 1$. For $N = (\frac{\Delta p}{2})^{-2} \times 2 = 2^{18.5}$, the attack can determine the value of K_{21} with success rate 0.977.

3.3 Recovering $K_{22}K_{23} \ldots K_{30}$

By tracing the bits in the initialization process, we notice that each s_{22+i} is mapped to $s_{192+7+i}$ with probability $\frac{1}{27}$ for $0 \leq i \leq 8$ (each of them is only mapped by π_1 at s_{t+5}). We know that $s_{22+i} = K_{22+i} \vee IV_{22+i}$, and $s_{192+7+i}$, $s_{192+9+i}$ are used in the generation of $y_{193+i,2}$ for $0 \leq i \leq 10$. In this subsection, we show that the key bits $K_{22}K_{23}K_{24} \ldots K_{30}$ can be recovered from the keystream.

The attack similar to that given in Subsection 3.1 can be applied to recover the value of K_{23} from the first keystream bits generated from $2^{18.5}$ IVs.

To determine the values of K_{22} and K_{24}, we observe the second bit of the keystream. Due to the disturbance of the ABSG, $y_{193,2}$ becomes the second keystream bit with probability 0.5. Thus $\Delta p' = 0.5 \times \Delta p = 2^{-8.75}$. To recover K_{22} and K_{24}, we need $2^{20.5}$ IVs in order to obtain the success rate 0.977.

To determine the value of K_{25}, we observe the second and third bits of the keystream. $y_{194,2}$ would become the second bit of the keystream with probability $\frac{1}{8}$, and become the third bit of the keystream with probability $\frac{1}{4}$. Thus $\Delta p'' = \frac{1}{2} \times (\frac{1}{4} + \frac{1}{8}) \times \Delta p = 2^{-10.165}$. To recover K_{25}, we need $2^{22.3}$ IVs in order to obtain the success rate 0.977.

We omit the details of recovering $K_{26} \cdots K_{29}$. To recover K_{30}, we observe the fifth, sixth and seventh bits of the keystream. $y_{199,2}$ would become one of these three bits with probability $\frac{77}{256}$. Thus $\Delta p''' = \frac{1}{3} \times \frac{77}{256} \times \Delta p = 2^{-11.068}$. To recover K_{29}, we need $2^{23.5514}$ IVs in order to obtain the success rate 0.977.

3.4 Recovering $K_9K_{10} \ldots K_{19}$

Tracing the bits in the initialization process, we notice that each s_{9+i} is mapped to $s_{192+166+i}$ with probability $\frac{1}{27}$ for $0 \leq i \leq 10$ (each of them is only mapped by π_1 at s_{t+5}). We know that $s_{9+i} = K_{9+i} \vee IV_{9+i}$, and $s_{192+166+i}$ is used in the

generation of $y_{194+i,1}$ for $0 \leq i \leq 10$. The attacks given in this subsection are similar to those given in the above subsection. We only illustrate how to recover K_9 and K_{19}.

To determine the value of K_9, we observe the second bit of the keystream. $y_{194,1}$ would become the second bit of the keystream with probability $\frac{1}{4}$. Thus $\Delta p^{(4)} = \frac{1}{4} \times \Delta p = 2^{-9.75}$. To recover K_9, we need $2^{22.5}$ IVs in order to obtain the success rate 0.977.

To determine the value of K_{19}, we observe the 8-th, 9-th and 10-th bits of the keystream. $y_{204,1}$ would become one of these three bits with probability 0.25966. Thus $\Delta p^{(5)} = \frac{1}{3} \times 0.25966 \times \Delta p = 2^{-11.28}$. To recover K_{19}, we need $2^{23.98}$ IVs in order to obtain the success rate 0.977.

3.5 Recovering $K_{32}K_{33}\ldots K_{46}$

Tracing the bits in the initialization process, we notice that each s_{144+i} is mapped to $s_{192+16+i}$ with probability $\frac{1}{27}$ for $0 \leq i \leq 14$ (each of them is only mapped by π_1 at s_{t+5}). We know that $s_{144+i} = K_{32+i} \vee IV_{32+i}$, and $s_{192+16+i}$ is used in the generation of $y_{200+i,1}$ for $0 \leq i \leq 14$.

Since for s_{144+i} $(0 \leq i \leq 14)$, the key bits are XORed with the IV bits, the attack is slightly modified. For example, if the probability of 0 in the keystream for $IV_{32} = 0$ is higher than the probability of 0 in the keystream for $IV_{32} = 1$, then we predict that $K_{32} = 1$; otherwise, $K_{32} = 0$.

We only illustrate how to recover K_{32} and K_{46}.

To determine the value of K_{32}, we observe the sixth, seventh and eighth bits of the keystream. $y_{200,2}$ would become one of these three bits with probability 0.28027. Thus $\Delta p^{(6)} = \frac{1}{3} \times 0.28027 \times \Delta p = 2^{-11.17}$. To recover K_{32}, we need $2^{23.755}$ IVs in order to obtain the success rate 0.977.

To determine the value of K_{46}, we assume that starting from the fourth bit of the sequence y, each bit would become the output with probability $\frac{1}{3}$. Then $y_{214,2}$ would become one of the 12th, 13th, ..., 18th bits of the keystream with probability 0.16637. Thus $\Delta p^{(7)} = \frac{1}{7} \times 0.16637 \times \Delta p = 2^{-13.145}$. To recover K_{29}, we need $2^{26.482}$ IVs in order to obtain the success rate 0.977.

The attacks given in this section recover 36 bits of the secret key with about 2^{26} random IVs. For each IV, only the first 3 bytes of the keystream are needed in the attack.

4 Improving the Key Recovery Attack

In the above attacks, we only deal with the bits permuted only by π_1 at s_{t+5}. To improve the attack above, we trace all the possibilities for each bit s_i $(0 \leq i \leq 175)$ during the initialization process to find out the distribution of that bit at the end of initialization. Then we search the optimal attack for that bit. We performed the experiment, and found that 44 key bits can be recovered with less than 2^{20} IVs, and only the first 2 bytes of the keystream are required in the attack. The preliminary experiment results are given in Table 2 in Appendix A.

5 The Keystream of DECIM Is Heavily Biased

The improperly designed function f in DECIM results in heavily biased keystream.

5.1 The keysteam is biased

We start with analyzing the function f.

$$f(x_{i_1}, ..., x_{i_7}) = \sum_{1 \le j < k \le 7} x_{i_j} x_{i_k}$$

If any one bit of the input of f is with value 1, then f generates '1' with probability $\frac{72}{128}$; otherwise it generates '1' with probability $\frac{56}{128}$. Thus for $f(x_{i_1}, ..., x_{i_7})$ and $f(x'_{i_1}, ..., x'_{i_7})$, if one bit of one input is always equal to one bit of another inputs (i.e., $x_{i_a} = x'_{i_b}$ where $0 \le a, b \le 7$), then the outputs related to these two inputs would be equal with probability $(\frac{56}{128})^2 + (\frac{72}{128})^2 = \frac{65}{128}$.

Note that $y_{t,1}$ and $y_{t,2}$ are computed as follows

$$y_{t,1} = f(s_{t+1}, s_{t+32}, s_{t+40}, s_{t+101}, s_{t+164}, s_{t+178}, s_{t+187})$$

$$y_{t,2} = f(s_{t+6}, s_{t+8}, s_{t+60}, s_{t+116}, s_{t+145}, s_{t+181}, s_{t+191})$$

Denote $A = \{1, 32, 40, 101, 164, 178, 187\}$, $B = \{6, 8, 60, 116, 145, 181, 191\}$, and denote each element of A as a_i, and each element of B as b_i $(1 \le i \le 7)$. Then $y_{t,1} = y_{t+a_i-a_j,1}$ and $y_{t,2} = y_{t+b_i-b_j,2}$ with probability $\frac{65}{128}$ for $1 \le i, j \le 7$ and $i \ne j$. And $y_{t+b_i-a_j,1} = y_{t,2}$ with probability $\frac{65}{128}$ for $1 \le i, j \le 7$. It shows that the binary sequence y is heavily biased.

The heavily biased sequence y is used as input to the ABSG decimation algorithm. It results in heavily biased output. In the attack, we are interested in those biases in y that would not be significantly reduced by the ABSG Algorithm. Thus we will analyze the bias of $(y_{t+3,1}, y_{t,2})$, $(y_{t+4,1}, y_{t,2})$ and $(y_{t,2}, y_{t+2,2})$ to find out how they affect the randomness of the output of ABSG.

For example, we analyze the effect of the bias of $(y_{t+3,1}, y_{t,2})$. $y_{t+3,1} = y_{t,2}$ with probability $\frac{65}{128}$. Denote the i-th bit of the sequence y as y^i. Thus $y^i = y^{i+5}$ with probability $\frac{129}{256}$. (y^i, y^{i+5}) would affect the bias of the output of the ABSG in two approaches. One approach is that (y^i, y^{i+5}) would become (z_j, z_{j+2}) with probability $\frac{1}{4}$ (case 1: $y_i = y_{i-1}$, $y_{i+2} \ne y_{i+1}$ and $y_{i+3} = y_{i+2}$; case 2: $y_i \ne y_{i-1}$, $y_{i+1} = y_{i-1}$ and $y_{i+3} = y_{i+2}$). Thus for this approach, the bias of (y^i, y^{i+5}) causes that $z_j = z_{j+2}$ with probability $\frac{513}{1024}$. Another approach is that if $y_i = y_{i-1}$ and $y_{i+2} = y_{i+1}$, then (y_i, y_{i+4}) would become (z_j, z_{j+2}). Note that $y_{i+4} = y_{i-1}$ with probability $\frac{129}{256}$, so $z_j = z_{j+2}$ with probability $\frac{129}{256}$. This approach happens with probability $\frac{1}{4}$. Thus the bias of (y^i, y^{i+5}) causes that $z_j = z_{j+2}$ with probability $\frac{513}{1024}$. Combining these two approaches, we know that $z_j = z_{j+2}$ with probability $\frac{257}{512}$.

We continue analyzing the above example since the output of ABSG decimation algorithm should pass through the buffer before becoming keystream. By analyzing the ABSG decimation algorithm and the buffer, we notice that

if (y^i, y^{i+5}) becomes $z_j = z_{j+2}$ after the ABSG decimation algorithm, then it would become $z'_k = z'_{k+1}$ with probability 0.6135 after passing through the buffer; if (y^i, y^{i+4}) becomes $z_j = z_{j+2}$ after the ABSG decimation algorithm, then it would become $z'_k = z'_{k+1}$ with probability 0.5189 after passing through the buffer. Thus after passing through the buffer, the two approaches lead to $z'_k = z'_{k+1}$ with probability $\frac{1}{2} + 0.6135 \times \frac{1}{1024} + 0.5189 \times \frac{1}{1024} = \frac{1}{2} + 2^{-9.82}$.

The similar analysis can be applied to the biases resulting from $(y_{t+4,1}, y_{t,2})$ and $(y_{t,2}, y_{t+2,2})$. The bias of $(y_{t,2}, y_{t+2,2})$ would cause $z'_k = z'_{k+1}$ with probability about $\frac{1}{2} + 2^{-10.84}$, and the bias of $(y_{t+4,1}, y_{t,2})$ would cause $z'_k = z'_{k+1}$ with probability about $\frac{1}{2} + 2^{-11.73}$.

Combining the effects of $(y_{t+3,1}, y_{t,2})$, $(y_{t+4,1}, y_{t,2})$ and $(y_{t,2}, y_{t+2,2})$, the bias of $z'_k = z'_{k+1}$ is about $\frac{1}{2} + 2^{-9.82} + 2^{-10.84} + 2^{-11.73} = \frac{1}{2} + 2^{-9.00}$.

Now we verify the above analysis with experiment. We generated about 2^{30} keystream bits from DECIM and found that $z'_k = z'_{k+1}$ is about $\frac{1}{2} + 2^{-8.67}$. The experiment result shows that the analysis result is close to that obtained from the experiment.

5.2 Broadcast attack

Due to the bias in the keystream, part of the message could be recovered from the ciphertexts if the same message is encrypted for many times using DECIM with random key and IV pairs. The similar attack has been applied to RC4 [2].

Suppose that one message bit is encrypted for N times, and each keystream bit is 0 with probability $\frac{1}{2} + \Delta p$ with $\Delta p > 0$. Denote n_0 as the number of '0' in the ciphertext bits. If $n_0 > \frac{N}{2}$, we conclude that the message bit is with value 0; otherwise, we conclude that the message bit is with value 1. For $N = \Delta p^{-2}$, the message bit could be recovered with success rate 0.977.

Thus if one message is encrypted with different keys and IVs for about 2^{18} times, the message could be recovered from the ciphertexts.

6 Attacks on DECIM with 80-bit IV

The keystream generation algorithm of DECIM with 80-bit IV is the same as DECIM with 64-bit IV. Thus DECIM with 80-bit IV still generates heavily biased keystream and vulnerable to the broadcast attack.

The initialization process of DECIM with 80-bit IV is slightly different from the 64-bit IV version. The key and IV are loaded into the LFSR as

$$s_i = \begin{cases} 0 & \text{for } 0 \le i \le 31 \\ K_{i-32} \oplus IV_{i-32} & \text{for } 32 \le i \le 111 \\ K_{i-112} & \text{for } 112 \le i \le 191 \end{cases}$$

Similar to the attack given in Section 4, we carry out the experiment to compute the IVs required to recover each bit. With 2^{21} IVs, 41 bits of the secret key could be recovered. Only the first 2 bytes of the keystream are required in the attack. The experiment results are given in Table 3 in Appendix A.

7 Conclusion

DECIM is insecure.

References

1. C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin and H. Sibert. "Decim - A New Stream Cipher for Hardware Applications". *ECRYPT Stream Cipher Project Report 2005/004*. Available at http://www.ecrypt.eu.org/stream/
2. I. Mantin, A. Shamir. "A Practical Attack on Broadcast RC4". *Fast Software Encryption (FSE2001)*, LNCS2335, pp. 152-164, Springer-Verlag, 2001.

A The Amount of IVs Required to Break DECIM

Table 2 gives the amount of IVs required to break DECIM with 64-bit IV. 44 key bits can be recovered with less than 2^{20} IVs. Table 3 gives the amount of IVs required to break DECIM with 80-bit IV. 41 key bits can be recovered with less than 2^{21} IVs. Only the first 2 bytes of the keystream are required in the attack, and the amount of computations required in the attacks is negligible.

We explain Table 2 with K_0 as an example. K_0 is related to s_{112} since $s_{112} = K_0 \oplus IV_0$. s_{112} is mapped to s_{192+60} with probability 0.0318 (this probability is obtained by tracing s_{112} through the initialization process). Thus K_0 could be recovered by observing the first bits of the keystreams. About $2^{18.95}$ IVs are required to achieve the success rate 0.977.

Table 1. Amount of IVs required to recover the key bits (64-bit IV)

	Affected Bits	Amount of IVs (Log_2)		Affected Bits	Amount of IVs (Log_2)
K_0	$s_{112} \Rightarrow s_{192+60}$	18.95	K_1	$s_{57} \Rightarrow s_{192+122}$	20.83
K_2	$s_{58} \Rightarrow s_{192+116}$	18.80	K_3	$s_{115} \Rightarrow s_{192+104}$	20.46
K_4	$s_{116} \Rightarrow s_{192+105}$	21.41	K_5	$s_{117} \Rightarrow s_{192+106}$	21.54
K_6	$s_{118} \Rightarrow s_{192+107}$	21.67	K_7	$s_{119} \Rightarrow s_{192+108}$	21.72
K_8	$s_{120} \Rightarrow s_{192+145}$	21.21	K_9	$s_{121} \Rightarrow s_{192+110}$	21.92
K_{10}	$s_{10} \Rightarrow s_{192+116}$	17.69	K_{11}	$s_{11} \Rightarrow s_{192+117}$	19.62
K_{12}	$s_{68} \Rightarrow s_{192+6}$	18.88	K_{13}	$s_{69} \Rightarrow s_{192+7}$	20.82
K_{14}	$s_{70} \Rightarrow s_{192+8}$	18.82	K_{15}	$s_{127} \Rightarrow s_{192+116}$	16.66
K_{16}	$s_{128} \Rightarrow s_{192+117}$	18.70	K_{17}	$s_{17} \Rightarrow s_{192+6}$	16.92
K_{18}	$s_{18} \Rightarrow s_{192+7}$	18.82	K_{19}	$s_{19} \Rightarrow s_{192+8}$	16.80
K_{20}	$s_{20} \Rightarrow s_{192+9}$	18.73	K_{21}	$s_{21} \Rightarrow s_{192+6}$	18.59
K_{22}	$s_{22} \Rightarrow s_{192+7}$	20.67	K_{23}	$s_{23} \Rightarrow s_{192+8}$	18.70
K_{24}	$s_{80} \Rightarrow s_{192+146}$	20.80	K_{25}	$s_{25} \Rightarrow s_{192+116}$	17.97
K_{26}	$s_{138} \Rightarrow s_{192+6}$	17.79	K_{27}	$s_{139} \Rightarrow s_{192+7}$	19.87
K_{28}	$s_{140} \Rightarrow s_{192+8}$	17.86	K_{29}	$s_{141} \Rightarrow s_{192+9}$	19.67
K_{30}	$s_{142} \Rightarrow s_{192+10}$	21.46	K_{31}	$s_{31} \Rightarrow s_{192+182}$	18.36
K_{32}	$s_{32} \Rightarrow s_{192+183}$	20.70	K_{33}	$s_{33} \Rightarrow s_{192+113}$	20.97
K_{34}	$s_{34} \Rightarrow s_{192+114}$	21.03	K_{35}	$s_{91} \Rightarrow s_{192+116}$	19.95
K_{36}	$s_{36} \Rightarrow s_{192+116}$	15.55	K_{37}	$s_{37} \Rightarrow s_{192+117}$	17.56
K_{38}	$s_{94} \Rightarrow s_{192+145}$	18.94	K_{39}	$s_{39} \Rightarrow s_{192+104}$	19.62
K_{40}	$s_{152} \Rightarrow s_{192+60}$	16.43	K_{41}	$s_{153} \Rightarrow s_{192+116}$	17.90
K_{42}	$s_{154} \Rightarrow s_{192+117}$	19.93	K_{43}	$s_{43} \Rightarrow s_{192+108}$	20.61
K_{44}	$s_{156} \Rightarrow s_{192+145}$	16.90	K_{45}	$s_{157} \Rightarrow s_{192+146}$	18.96
K_{46}	$s_{46} \Rightarrow s_{192+35}$	20.45	K_{47}	$s_{47} \Rightarrow s_{192+6}$	16.68
K_{48}	$s_{160} \Rightarrow s_{192+145}$	18.68	K_{49}	$s_{161} \Rightarrow s_{192+181}$	15.59
K_{50}	$s_{162} \Rightarrow s_{192+182}$	17.59	K_{51}	$s_{51} \Rightarrow s_{192+116}$	15.62
K_{52}	$s_{52} \Rightarrow s_{192+117}$	17.64	K_{53}	$s_{53} \Rightarrow s_{192+118}$	19.47
K_{54}	$s_{54} \Rightarrow s_{192+119}$	20.05	K_{55}	$s_{55} \Rightarrow s_{192+120}$	20.61
K_{56}	$s_{168} \Rightarrow s_{192+76}$	22.27	K_{57}	$s_{169} \Rightarrow s_{192+103}$	18.43
K_{58}	$s_{170} \Rightarrow s_{192+104}$	18.17	K_{59}	$s_{171} \Rightarrow s_{192+105}$	18.93
K_{60}	$s_{172} \Rightarrow s_{192+106}$	19.11	K_{61}	$s_{173} \Rightarrow s_{192+107}$	19.24
K_{62}	$s_{174} \Rightarrow s_{192+108}$	19.42	K_{63}	$s_{175} \Rightarrow s_{192+109}$	19.58

Table 2. Amount of IVs required to recover the key bits (80-bit IV)

	Affected Bits	Amount of IVs (Log_2)		Affected Bits	Amount of IVs (Log_2)
K_0	$s_{32} \Rightarrow s_{192+183}$	20.70	K_1	$s_{33} \Rightarrow s_{192+113}$	20.97
K_2	$s_{34} \Rightarrow s_{192+114}$	21.03	K_3	$s_{35} \Rightarrow s_{192+115}$	21.13
K_4	$s_{36} \Rightarrow s_{192+116}$	15.55	K_5	$s_{37} \Rightarrow s_{192+117}$	17.56
K_6	$s_{38} \Rightarrow s_{192+118}$	19.43	K_7	$s_{39} \Rightarrow s_{192+104}$	19.62
K_8	$s_{40} \Rightarrow s_{192+105}$	20.37	K_9	$s_{41} \Rightarrow s_{192+121}$	20.30
K_{10}	$s_{42} \Rightarrow s_{192+107}$	20.48	K_{11}	$s_{43} \Rightarrow s_{192+108}$	20.61
K_{12}	$s_{44} \Rightarrow s_{192+109}$	20.77	K_{13}	$s_{45} \Rightarrow s_{192+34}$	20.70
K_{14}	$s_{46} \Rightarrow s_{192+35}$	20.45	K_{15}	$s_{47} \Rightarrow s_{192+6}$	16.68
K_{16}	$s_{48} \Rightarrow s_{192+7}$	18.72	K_{17}	$s_{49} \Rightarrow s_{192+8}$	16.68
K_{18}	$s_{50} \Rightarrow s_{192+9}$	18.66	K_{19}	$s_{51} \Rightarrow s_{192+116}$	15.62
K_{20}	$s_{52} \Rightarrow s_{192+117}$	17.64	K_{21}	$s_{53} \Rightarrow s_{192+118}$	19.47
K_{22}	$s_{54} \Rightarrow s_{192+119}$	20.05	K_{23}	$s_{55} \Rightarrow s_{192+120}$	20.61
K_{24}	$s_{56} \Rightarrow s_{192+121}$	20.63	K_{25}	$s_{57} \Rightarrow s_{192+122}$	20.83
K_{26}	$s_{58} \Rightarrow s_{192+116}$	18.80	K_{27}	$s_{59} \Rightarrow s_{192+12}$	23.00
K_{28}	$s_{60} \Rightarrow s_{192+13}$	23.41	K_{29}	$s_{61} \Rightarrow s_{192+14}$	23.66
K_{30}	$s_{62} \Rightarrow s_{192+15}$	23.78	K_{31}	$s_{63} \Rightarrow s_{192+16}$	24.09
K_{32}	$s_{64} \Rightarrow s_{192+17}$	24.00	K_{33}	$s_{65} \Rightarrow s_{192+18}$	24.19
K_{34}	$s_{66} \Rightarrow s_{192+19}$	24.22	K_{35}	$s_{67} \Rightarrow s_{192+5}$	23.44
K_{36}	$s_{68} \Rightarrow s_{192+6}$	18.88	K_{37}	$s_{69} \Rightarrow s_{192+7}$	20.82
K_{38}	$s_{70} \Rightarrow s_{192+8}$	18.82	K_{39}	$s_{71} \Rightarrow s_{192+60}$	16.77
K_{40}	$s_{72} \Rightarrow s_{192+61}$	18.75	K_{41}	$s_{73} \Rightarrow s_{192+62}$	20.59
K_{42}	$s_{74} \Rightarrow s_{192+63}$	21.11	K_{43}	$s_{75} \Rightarrow s_{192+64}$	21.71
K_{44}	$s_{76} \Rightarrow s_{192+65}$	21.67	K_{45}	$s_{77} \Rightarrow s_{192+66}$	21.85
K_{46}	$s_{78} \Rightarrow s_{192+67}$	21.81	K_{47}	$s_{79} \Rightarrow s_{192+145}$	18.82
K_{48}	$s_{80} \Rightarrow s_{192+146}$	20.80	K_{49}	$s_{81} \Rightarrow s_{192+70}$	22.05
K_{50}	$s_{82} \Rightarrow s_{192+71}$	22.18	K_{51}	$s_{83} \Rightarrow s_{192+72}$	22.40
K_{52}	$s_{84} \Rightarrow s_{192+73}$	22.43	K_{53}	$s_{85} \Rightarrow s_{192+74}$	22.42
K_{54}	$s_{86} \Rightarrow s_{192+75}$	22.43	K_{55}	$s_{87} \Rightarrow s_{192+76}$	22.55
K_{56}	$s_{88} \Rightarrow s_{192+154}$	24.02	K_{57}	$s_{89} \Rightarrow s_{192+155}$	24.04
K_{58}	$s_{90} \Rightarrow s_{192+156}$	24.15	K_{59}	$s_{91} \Rightarrow s_{192+116}$	19.95
K_{60}	$s_{92} \Rightarrow s_{192+117}$	21.97	K_{61}	$s_{93} \Rightarrow s_{192+118}$	23.77
K_{62}	$s_{94} \Rightarrow s_{192+145}$	18.94	K_{63}	$s_{95} \Rightarrow s_{192+146}$	20.91
K_{64}	$s_{96} \Rightarrow s_{192+147}$	22.79	K_{65}	$s_{97} \Rightarrow s_{192+148}$	23.33
K_{66}	$s_{98} \Rightarrow s_{192+149}$	23.77	K_{67}	$s_{99} \Rightarrow s_{192+150}$	23.64
K_{68}	$s_{100} \Rightarrow s_{192+63}$	22.65	K_{69}	$s_{101} \Rightarrow s_{192+4}$	23.12
K_{70}	$s_{102} \Rightarrow s_{192+65}$	23.66	K_{71}	$s_{103} \Rightarrow s_{192+178}$	23.80
K_{72}	$s_{104} \Rightarrow s_{192+179}$	23.77	K_{73}	$s_{105} \Rightarrow s_{192+145}$	20.94
K_{74}	$s_{106} \Rightarrow s_{192+181}$	18.24	K_{75}	$s_{107} \Rightarrow s_{192+182}$	19.97
K_{76}	$s_{108} \Rightarrow s_{192+183}$	21.81	K_{77}	$s_{109} \Rightarrow s_{192+6}$	20.86
K_{78}	$s_{110} \Rightarrow s_{192+7}$	22.83	K_{79}	$s_{111} \Rightarrow s_{192+8}$	20.94