TMD-Tradeoff and State Entropy Loss Considerations of Streamcipher MICKEY

Jin Hong and Woo-Hwan Kim

National Security Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, Korea
{jinhong,whkim5}@etri.re.kr

Abstract. We give three weaknesses of a recently proposed streamcipher MICKEY. A small class of weak keys is found and we show time-memory-data tradeoff is applicable. We also show that the state update function reduces entropy of the internal state as it is iterated, resulting in keystreams that start out differently but become merged together towards the end.

Keywords: MICKEY, stream cipher, time memory data tradeoff, internal state entropy, weak key

1 Introduction

Streamcipher MICKEY [4] is a low-end hardware oriented cipher, designed by S. Babbage and M. Dodd, with 80-bit intended security level, and submitted to ECRYPT for competition in the streamcipher project [1]. It was one of the algorithms selected and presented at the Symmetric Key Encryption Workshop (SKEW, Århus, Denmark, May, 2005).

Internal state of the cipher consists of two 80-bit feedback shift registers that are carefully designed to mutually and irregularly clock each other. The output keystream is a simple XOR of the two register outputs and hence not much more than the two registers themselves are needed for a hardware implementation of MICKEY. Since the minimum internal state size for a streamcipher aiming for 80-bit security is 160 bits, the cipher seems to be quite near the minimum point reachable with respect to hardware implementation cost and this makes it a very attractive proposal for constrained hardware environments.

In this paper, three undesirable properties of MICKEY, in relation to its security aspects, are discussed. The first is that time-memory-data tradeoff is applicable with online complexity lower than exhaustive search. It is widely believed that using an internal state size of twice the key size eliminates TMD tradeoff attacks completely. Here, we show that, with a technique called BSW sampling, we can effectively reduce the search space and apply TMD tradeoff to a reduced internal state.

Our second remark starts from the fact that the state update function of MICKEY is not bijective. This reduces entropy of the internal state as the keystream generator is run forward and we study the iterative entropy loss quantitatively. We show that although the restriction to 2^{40}-bits of keystream usage per key setup, set up by the designers, prevents any keystream from circling back through a whole period, it does not stop two long keystreams that started out differently from merging together towards the end.

The last observation we make is the existence of a small family of weak keys.

After briefly describing MICKEY, we shall go through the three weaknesses in the same order introduced above. In the appendix, we consider application of some of these ideas to the higher security cousin MICKEY-128.

2 MICKEY

Let us very briefly recall the specifications of MICKEY and fix notation. As we shall not be working with the key schedule, just the main body will be described.

MICKEY uses two 80-bit registers named $\mathcal{R}$ and $\mathcal{S}$. Register $\mathcal{R}$ is a linear feedback shift register whose cells we denote by r_i $(0 \leq i \leq 79)$. Depending on a *control bit*, it is clocked in two different ways. If the control bit is set to 0, it is clocked in the normal way. When the control bit is set to 1, it is clocked $2^{40} - 23$ times. Of course, a way to do this without actually repeatedly clocking the register is given, but we shall not be concerned with such implementation aspects. Register $\mathcal{S}$ is a nonlinear feedback shift register whose cells we denote by s_i $(0 \leq i \leq 79)$. As with $\mathcal{R}$, there are two ways to clock $\mathcal{S}$, the choice being made through another *control bit*.

Since $\mathcal{R}$ is an LFSR, it is clear that, once the control bit used for the most recent clocking is known, it may be inverted one step back. The inversion map will be linear for both cases of the control bit. Similarly, as the designers state in their specifications, the clocking for $\mathcal{S}$ is also invertible, once the control bit is fixed. The actual inverse clocking method is easy to find once the normal clocking method is fully understood.

One bit of keystream is produced *before* each internal state update by XORing two bits contained in cells r_0 and s_0. To update the whole generator, the XOR $s_{27} \oplus r_{53}$ of two bits is set as control bit for register $\mathcal{R}$, the bit $s_{53} \oplus r_{26}$ is set as the control bit for register $\mathcal{S}$, and the two 80-bit registers are clocked accordingly. After each key-IV loading, which we do not describe, the keystream generator may be used to produce at most 2^{40} bits of keystream.

The logics of this paper should be understandable with what has so far been explained of MICKEY. But those interested in following actual computations should consult the original specification [4] for more information.

3 TMD tradeoff with BSW sampling

In this section, we shall see that time-memory-data tradeoff is applicable to the internal state of MICKEY with online complexity less than exhaustive search.

3.1 BS-tradeoff

The problem of recovering the internal state of a streamcipher, given multiple keystream segments, may be seen as a special case of the following more general *inversion* problem.

> Given a oneway function $f : \mathcal{X} \to \mathcal{Y}$ and a set $\mathcal{D} = \{y_i\}_i \subset \mathcal{Y}$, find at least one $x_i \in X$ for which $f(x_i) = y_i$.

Our interest lies in the case where $\mathcal{X}$ is the set of all internal states, $\mathcal{Y}$ is the set of keystream segments long enough to reliably distinguish states, and f is the mapping of state to finite keystream segment. Once the internal state corresponding to some keystream segment is obtained by inverting this mapping, the cipher can be run forward to obtain all future keystream for that session.

Starting with the work of Hellman [9], numerous studies on time-memory-(data) tradeoff attacks have appeared in the literature, but in this work, we shall focus on the TMD tradeoff attack given by Biryukov and Shamir [5]. The BS-tradeoff, interpreted as a tool for inverting general oneway functions $f : \mathcal{X} \to \mathcal{Y}$, is as follows.

> Let $N = |\mathcal{X}|$ be the size of the search space. Given any triple (T, M, D) satisfying the *tradeoff curve*
>
> $$TM^2D^2 = N^2 \quad \text{with} \quad 1 \leq D^2 \leq T, \tag{1}$$
>
> there exists an algorithm that solves the inversion problem in the following manner.
> Before a target set $\mathcal{D}$ of size D is given, the attacker prepares tables to be stored in memory of size M through a pre-computation phase requiring (offline) time
>
> $$P = N/D. \tag{2}$$
>
> When the target data set $\mathcal{D}$ is given, with high probability, a single solution to the inversion problem is produced within (online) time T.

A typical point on the BS-tradeoff curve is

$$T = M = N^{\frac{1}{2}}, \quad D = N^{\frac{1}{4}}, \quad \text{with} \quad P = N^{\frac{3}{4}}. \tag{3}$$

Since the attack complexity of this approach is taken to be the maximum of T, M, and D, disregarding the offline time complexity, this has worked, together with birthday paradox based TMD attacks [3, 8], as a reason for many recent streamcipher designs incorporating internal state size at least twice as big as intended security level. MICKEY follows this trend and uses 160-bit state size to aim for 80-bit security.

3.2 Restricting the search space

Let us explain the basics of BSW-sampling [6]. Suppose we can find a subset $\mathcal{X}' \subset \mathcal{X}$ for which elements of $\mathcal{Y}' = f(\mathcal{X}')$ can easily be singled out from $\mathcal{Y}$.

Consider the restriction

$$f' : \mathcal{X}' \to \mathcal{Y}'$$

of f to $\mathcal{X}'$. Given a target data set $\mathcal{D} \subset \mathcal{Y}$, one *samples* the data by taking $\mathcal{D}' = \mathcal{D} \cap \mathcal{Y}'$ and apply TMD tradeoff to f' with the smaller data set $\mathcal{D}'$. As the search space $\mathcal{X}'$ is smaller than $\mathcal{X}$, this should be more efficient than applying TMD-tradeoff to f itself, and it is clear that any inversion of f' is also an inversion for f. Of course, the (pre-sampling) data requirement will be larger than the usual approach.

3.3 Sampling MICKEY keystream

The usual way of applying TMD tradeoff attack to MICKEY would be to consider the case where $\mathcal{X}$ is the set of all 160-bit internal states and $\mathcal{Y}$ is the set of all keystream segments of 160-bit length. The mapping f will send an internal state to the first 160 keystream bits obtained from the state.

To apply the sampling method, we define

$$\mathcal{Y}' = \{\text{160-bit keystream segments starting with 27 zeros}\}. \tag{4}$$

These are certainly easily distinguishable from the rest of the elements of $\mathcal{Y}$. With an understanding of BS-tradeoff algorithm, it should be clear that the only obstacle to its actually application on the restricted mapping

$$f' : \mathcal{X}' = f^{-1}(\mathcal{Y}') \to \mathcal{Y}' \tag{5}$$

is the construction of an efficiently computable, preferably injective, mapping

$$h : \mathcal{Y}' \to \mathcal{X}'. \tag{6}$$

The construction of this mapping for MICKEY is a bit tedious, so let us go through this step by step.

1. View a random element $\mathbf{y} \in \mathcal{Y}'$ as a 133-bit value by disregarding the 27 initiating zeros.
2. Fill register $\mathcal{S}$ with the first 80 bits of $\mathbf{y}$.
3. Fill cells $r_1, \ldots, r_{53}$ of register $\mathcal{R}$ with the remaining 53 bits of $\mathbf{y}$. We shall consider the remaining 27 cells, r_0 and $r_{54}, \ldots, r_{79}$ as having been filled with indeterminate variables x_0 and $x_{54}, \ldots, x_{79}$. To define the mapping (6), it suffices to determine the correct values for these variables which allow the keystream generated from this state to start with 27 zeros.
4. Since the first output bit $r_0 \oplus s_0$ must be zero, and since s_0 is already fixed, we immediately obtain x_0. Hence forth, we shall treat x_0 as a constant rather than as a variable.
5. Calculate $s_{53} \oplus r_{26}$, the control bit for $\mathcal{S}$, and also $s_{27} \oplus r_{53}$, the control bit for $\mathcal{R}$.
6. Clock register $\mathcal{S}$ according to its control bit.

7. Clock register $\mathcal{R}$ according to its control bit. In saying this, we mean to write the contents of the cells as linear functions of the remaining variables $x_{54}, \ldots, x_{79}$. In particular, using the feedback equation for $\mathcal{R}$, one can show that the new content of r_0 will be
   - x_{79}, if the control bit for $\mathcal{R}$ is zero, and
   - $x_{79} \oplus x_0$, if the control bit for $\mathcal{R}$ is one.

   Recalling the fact that x_0 has already been determined, and using the condition for the second output bit to be zero, one determines x_{79} which is actually the feedback bit. We can now treat x_{79} also as a constant, rather than as a variable.
8. Determination of the feedback bit allows cells $r_0, \ldots, r_{53}$ to be determined explicitly, and for the rest of the cells to be written as a function of the remaining 25 variables $x_{54}, \ldots, x_{78}$. For example, the current updated value of r_{79} will be either $x_{78} \oplus x_{79}$ or x_{78}, depending on the (known) control bit for $\mathcal{R}$.
9. Once again, calculate both control bits and clock the two registers accordingly.
10. This time, the new value for s_0 and the zero-output condition will determine the variable x_{78} and also the second feedback bit.
11. Continue in this manner until all variables are determined. Notice that each new clocking determines one more indeterminate and hence at most 26 clockings are needed.

We have thus defined an appropriate mapping (6) and successfully reduced the search space size from 2^{160} to $N = 2^{133}$.

3.4 TMD tradeoff with sampling on MICKEY

Let us now see what TMD tradeoff complexities we can obtain on the smaller space $\mathcal{X}'$.

Start with unfiltered data size of 2^{60}. For example, these may be obtained by sliding a window of 160-bit size over 2^{20}-many keystreams, each of $(2^{40} + 159)$-bit length. Only about 2^{-27} of these will start with 27 zeros. Hence the sampled data to be used in our TMD tradeoff attack is $D = 2^{33} = 2^{60-27}$. Values $T = 2^{66}$ and $M = 2^{67}$, together with $D = 2^{33}$ and $N = 2^{133}$, satisfy the TMD tradeoff curve (1), and the offline pre-computation complexity (2) becomes $P = 2^{100}$.

In summary, once a table costing offline time 2^{100} is built, TMD tradeoff attack with BSW sampling on MICKEY is possible with online complexity 2^{67}. This is (almost) the minimum online complexity achievable with our sampling.

To lower the pre-computation time as far as possible, we start with unfiltered data size of $2^{66.5}$. This results in the tradeoff point $T = 2^{79}$, $M = 2^{54}$, and $D = 2^{39.5}$, with pre-computation time $P = 2^{93.5}$.

Owing to the pre-computation complexity larger than exhaustive search of key, some would not view this technically as a *break* of MICKEY. But still, it does show that we cannot treat MICKEY as providing absolutely full 80-bit security. Furthermore, the fact that such sampling is possible can be viewed as a weakness in itself. This might open doors to other exploits.

Notice that ECRYPT's current recommendation for key sizes [2] seem to be viewing anything less than 81 bits to be vulnerable to exhaustive search by large agencies, hence the pre-computation time 2^{100} or $2^{93.5}$ cited above should be seen as reachable in the near future.

We refer readers to Appendix A for a related general treatment of complexities involving TMD-tradeoff with BSW sampling.

4 State entropy loss

The state update function of MICKEY starts by calculating two control bits. These bits determine the fate of the very bits used to obtain the control bits. This kind of double-use of information usually produces collisions. The state update function is not one-to-one and through repeated application of the state update function, entropy of the state is bound to decrease. In this section, we study the actual amount of entropy lost.

4.1 Preliminary

Given a set $X = \{x_i\}_{i\in I}$, with each element x_i appearing with probability p_i, the *entropy* of set X is defined as

$$H(X) = -\sum_{i\in I} p_i \log_2(p_i). \tag{7}$$

For example, if some set Y contains N elements, and all elements occur with equal probability, then we have

$$H(Y) = -\sum \frac{1}{N}\log_2(\frac{1}{N}) = \log_2(N).$$

Hence, a set of size 2^n with uniform probability distribution has entropy n.

Given a mapping φ, acting on a space of size N and of uniform distribution, let us define the notation

$$\mathrm{EL}(\varphi) = \log_2(N) - H(\mathrm{Image}(\varphi)), \tag{8}$$

$$\overline{\mathrm{EL}}(\varphi) = \log_2(N) - \log_2(\mathrm{Image}(\varphi)). \tag{9}$$

The operator EL measures the *entropy loss* suffered by a uniform space through application of a mapping. Operator $\overline{\mathrm{EL}}$ measures the same value in a rough way, by assuming that all elements of the image space occur with equal probability. We will always have $\mathrm{EL}(\varphi) \geq \overline{\mathrm{EL}}(\varphi)$ for any map φ.

4.2 Comparing update function with random mapping

There are some known results concerning the behavior of random mappings, and the following lemma may be found in [7].

Lemma 1. *For a random mapping on N elements, the expectation value for the number of its image points has the asymptotic form $(1 - \frac{1}{e})N$, as $N \to \infty$.*

Using our notation, this may be written equivalently in the following way.

Lemma 2. *For a random mapping φ on N elements, $\overline{\mathrm{EL}}(\varphi) \to -\log_2(1 - \frac{1}{e}) \sim 0.6617$ as $N \to \infty$.*

So, if a random function was chosen as the state update function for some streamcipher, the internal state would lose more than 0.66-bit entropy on its first update. Let us see how the MICKEY's state update function behaves in this regards.

For the rest of this section, φ will denote a *random mapping* and f will be used to denote the state update function of MICKEY. Consider the following procedure.

1. Choose random[1] states for both registers $\mathcal{R}$ and $\mathcal{S}$.
2. Calculate the reverse clocking of register $\mathcal{R}$, assuming control bit set to 0, and call the new state $\mathcal{R}_0$. Similarly, previous state of $\mathcal{R}$ assuming control bit 1 is calculated and named $\mathcal{R}_1$. The same is done for register $\mathcal{S}$ with results named $\mathcal{S}_0$ and $\mathcal{S}_1$.
3. For each of the four possible $(\mathcal{R}_i, \mathcal{S}_j)$ pairs, calculate the two control bits $(s_{27} \oplus r_{53}, s_{53} \oplus r_{26})$ and check to see if these match with the control bits (i, j) actually used. Count the number of matches.

This procedure counts the number of internal states that map to the chosen random state under the state update function f. We repeated this process for a total of 2^{20} times and have recorded the result in Table 1. Notice that a state may

inverse count	0	1	2	3	4	total
random states	307988	452017	279418	0	9153	2^{20}

Table 1. Random states according to f-inverse image counts

have from zero up to four pre-image states. Table shows, in particular, that of the 2^{20} random states, only 740588-many were image points under f. This accounts for about 0.7063 of the total states tried. Assuming that this proportion holds true for all states, we can calculate $\overline{\mathrm{EL}}(f) \sim -\log_2(0.7063) \sim 0.5017$. Comparing with 0.6617 entropy loss of a random functions stated by Lemma 2, we can see

[1] We used multiple invocations to `rand()` function provided by the C-language, initially seeded with current time.

that f loses less entropy and is closer to a one-to-one function than a random function.

We next worked with $f \circ f$, i.e., f iterated twice, in seeing the distribution of states according to inverse image counts. For every pre-image of a random state under f, we checked if this pre-image again had a pre-image under f. Result of twice iterated inverse image counting is given in Table 2. The total number of

inverse count	0	1	2	3	4	5	6	7	8	9	10	11	12	$13 \sim 16$	total
random state	$2^{18.81}$	$2^{17.95}$	$2^{17.92}$	$2^{15.82}$	$2^{14.65}$	$2^{10.35}$	$2^{11.20}$	$2^{7.93}$	$2^{7.48}$	$2^{5.55}$	$2^{3.32}$	0	$2^{4.00}$	0	2^{20}

Table 2. Random states according to $(f \circ f)$-inverse image counts

twice iterated images is 588990 and accounts for about 0.5617 of all 2^{20} states tried. As before, we can calculate $\overline{\mathrm{EL}}(f \circ f) \sim 0.8321$. From the view point of entropy preservation, $f \circ f$ is worse than random mappings.

Using the two comparisons, we conclude that, with respect to entropy preservation matters, a random mapping behaves somewhere in between f and $f \circ f$.

4.3 State update iterations

Our goal for the moment is to see how much internal state entropy is lost through 2^{40} iterated applications of state update function f. Notice that 2^{39} applications of $f \circ f$ is equal to 2^{40} applications of single f. As discussed in the previous subsection, random mapping φ is, in some sense, in the middle of f and $f \circ f$, and we can expect the behavior of a random mapping φ to be very much related to that of f.

Hence, we shall study 2^{40} applications of random mapping φ instead of working with the state update function f of MICKEY. More justification for this choice, given in terms of experiment data, will be given later.

Lemma 3. *For a random mapping on N elements, the expectation value for the number of k-th iterate image points has the asymptotic form $(1 - \tau_k)N$, as $N \to \infty$. Here, τ_k is given by the recursion formula, $\tau_0 = 0$, $\tau_{k+1} = \exp(\tau_k - 1)$.*

Let us denote the composition of k-many φ, i.e., k-many iterations of φ, by φ^k. The above lemma, found in [7], allows us to iteratively calculate $\overline{\mathrm{EL}}(\varphi^k)$.

Lemma 4. *For a random mapping φ, acting on a large enough space, we have*

$$\overline{\mathrm{EL}}(\varphi^k) \sim \log_2(k) - 1$$

for suitably large k.

We were careful not to state this as "$\overline{\mathrm{EL}}(\varphi^k) \to \log_2(k)-1$ as $k \to \infty$", since for a fixed space, repeated application of φ usually ends in a loop and hence the above is not true for all k. But as long as k is large enough, but not so large as to reach close to space size, the above should be true. We have no proof for this lemma, but experimental results given in Table 3 supports this very strongly. The table was created using Lemma 3, which allows us to assume $\overline{\mathrm{EL}}(\varphi^k) \sim -\log_2(1-\tau_k)$ for random mappings acting on large spaces.

k	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}
$\overline{\mathrm{EL}}(\varphi^k)$	0.66	1.09	1.68	2.40	3.23	4.13	5.07	6.04	7.02	8.011	9.006	10.003	11.0016

Table 3. Entropy loss estimate for iterated random mapping

Based on Lemma 4, we may say that after 2^{40} iterations of f, the internal state loses 39-bit entropy.

4.4 Test data

Let us calculate the entropy of Image(f). Going back to Table 1, one can assume

$$\frac{307988}{2^{20}} \cdot 2^{160} \sim 2^{158.23}$$

internal states never appear as image point under f. Probability of any one of these $2^{158.23}$-many states to be equal to a randomly produced image point under f is 0.

Similarly, $2^{158.79} \sim \frac{452017}{2^{20}} \cdot 2^{160}$ internal states have exactly one pre-image under f. If we fix any one of these and are given a randomly produced image point under f, then the two are equal with probability roughly $1/2^{160}$. There are $2^{158.09} \sim \frac{279418}{2^{20}} \cdot 2^{160}$ internal states with two f pre-images. If we fix any one of these, the probability of it coinciding with a randomly produced f-image state is $2/2^{160}$. Finally, for $2^{153.16} \sim \frac{9153}{2^{20}} \cdot 2^{160}$ states, their probability of coinciding with a randomly produced f-image is $4/2^{160}$.

The sum of all probabilities

$$0 \cdot 2^{158.23} + \frac{1}{2^{160}} \cdot 2^{158.79} + \frac{2}{2^{160}} \cdot 2^{158.09} + \frac{3}{2^{160}} \cdot 0 + \frac{4}{2^{160}} \cdot 2^{153.16} \sim 0.99894$$

is not exactly 1, but we shall ignore this.

Placing all information into the definition (7) for entropy gives us

$$\begin{aligned}
H(\mathrm{Image}(f)) &= -\Big\{2^{158.79}\frac{1}{2^{160}}\log_2(\frac{1}{2^{160}}) + 2^{158.09}\frac{2}{2^{160}}\log_2(\frac{2}{2^{160}}) \\
&\qquad + 0\frac{3}{2^{160}}\log_2(\frac{3}{2^{160}}) + 2^{153.16}\frac{4}{2^{160}}\log_2(\frac{4}{2^{160}})\Big\} \\
&= 160 - \Big\{\frac{279418}{2^{20}}2\log_2 2 + \frac{0}{2^{20}}3\log_2 3 + \frac{9153}{2^{20}}4\log_2 4\Big\} \\
&\sim 160 - 0.6028.
\end{aligned}$$

We have calculated the entropy loss $\mathrm{EL}(f) = 0.6028$ of state update function f. As expected, this is greater than the rough measure $\overline{\mathrm{EL}}(f) = 0.5017$, which we obtained earlier in Section 4.2. This process can be repeated for $f \circ f$ to obtain $\mathrm{EL}(f \circ f) = 0.9913$.

Let us denoting k-many compositions of f as f^k. Through the method just explained, we explicitly obtained $\mathrm{EL}(f^k)$ values for many values of k and we also did a similar job to find $\overline{\mathrm{EL}}(f^k)$. The results are given as a graph in Figure 1. The graph shows four curves, namely, three *rough* entropy measures $\overline{\mathrm{EL}}(f^k)$, $\overline{\mathrm{EL}}(\varphi^k)$, $\overline{\mathrm{EL}}((f \circ f)^k)$, and the curve $\mathrm{EL}(f^k)$, which we are most interested in. The graph for $\overline{\mathrm{EL}}(\varphi^k)$ is obtained through calculation, rather than experiment. The x-axis gives the logarithm of k-values and the y-axis gives the entropy loss in bits. The curve $\overline{\mathrm{EL}}((f \circ f)^k)$ is a left-shift of $\overline{\mathrm{EL}}(f^k)$ by one. As was argued in Section 4.2, we can confirm that the curve $\overline{\mathrm{EL}}(\varphi^k)$, which gives a rough measure of entropy loss for random mappings, sits comfortable in between respective curves for f and $f \circ f$.

We can also see that (by accident) the real entropy loss $\mathrm{EL}(f^k)$ for state update function follows the rough entropy loss $\overline{\mathrm{EL}}(\varphi^k)$ of random mapping very closely. Hence our use of $\overline{\mathrm{EL}}(\varphi^k)$ to approximate $\mathrm{EL}(f^k)$ in Section 4.3 is justified further. In any case, there is less than 1-bit entropy loss difference between any of the four graphs, and the exact value is less important than the fact that all graphs show *logarithmic growth* with respect to k.

In gathering the data points, we started out with 2^{20} independently chosen random states for small values of k, but with the increase of k, larger such samplings were needed to obtain a meaningful figure. Also, as getting a data point became increasingly time consuming with the growth of k, we worked for data points that are farther apart for these large k. To obtain a feel for how reliable our data points were, we did multiple tests with the same k. Multiple tests indicated that our $\overline{\mathrm{EL}}(f)$ data points were accurate enough for multiple points to be almost undistinguishable on the graph, but this was not so for $\mathrm{EL}(f)$ data points.

4.5 Security implications

We have, so far, confirmed that the state update function of MICKEY behaves more like a random function than a bijection on the internal state. This, in itself, is not a desirable property for a state update function.

Turning to more specific numbers, first recall that the designers restricted keystream use to 2^{40} bits for each key-IV setup. From discussions of Section 4.2, we can expect $\mathrm{EL}(f^{(2^{40})}) \sim 39$. In plain terms, this means that we can expect entropy loss of internal state to be equivalent to 39 bits after 2^{40} iterations of the state update function. Even if you were defending the designers, because $f \circ f$ behaves worse than a random mapping, one would have to admit that at least 38-bit entropy is lost. As the internal state started out with 160-bit entropy, at the end of the keystream of length 2^{40}, state entropy would be equivalent to about 121 random bits.

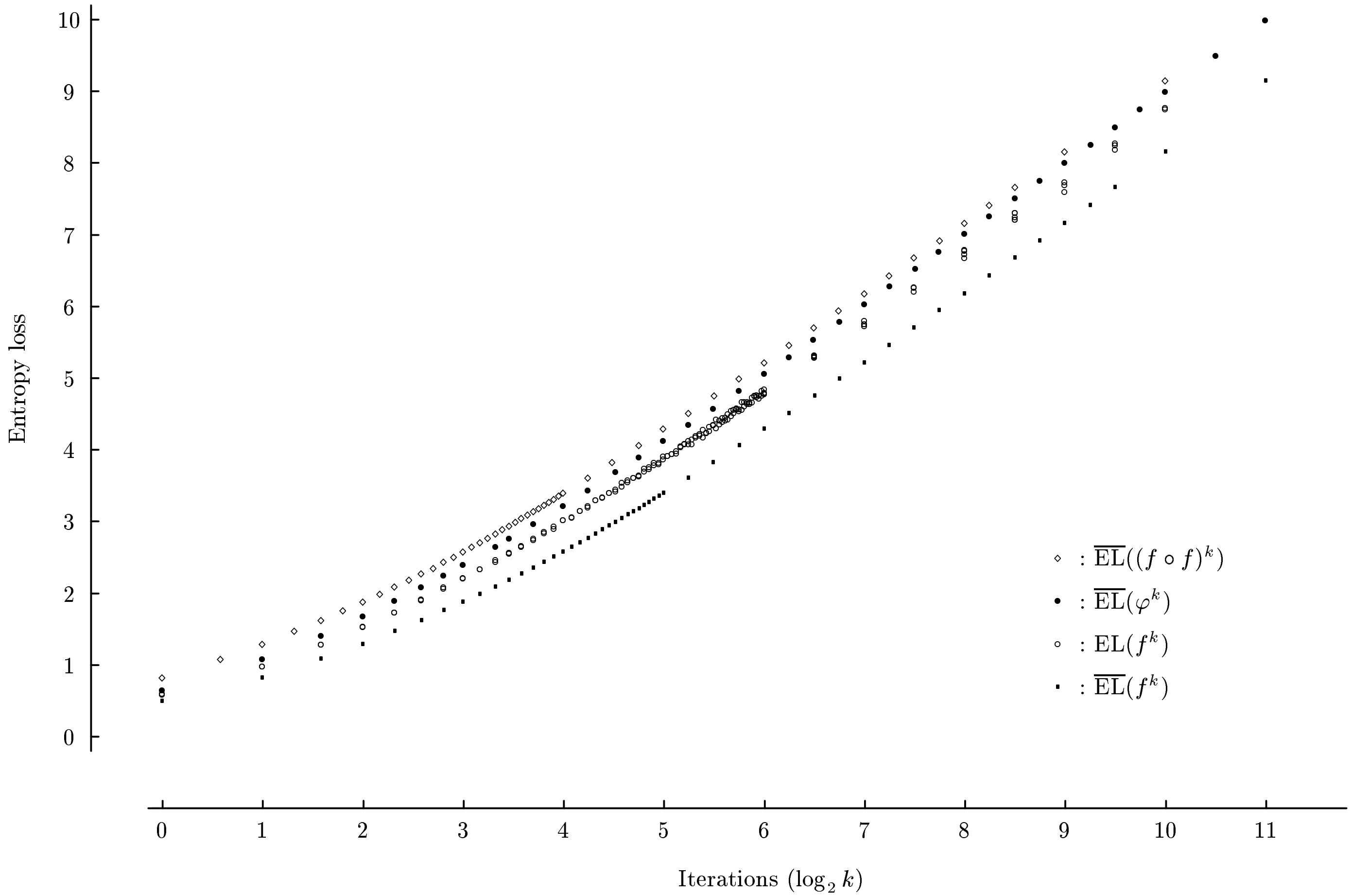

Fig. 1. Entropy loss under iteration

Now, suppose we collect long keystreams, each of length $2^{40} - \varepsilon$, where ε is small relative to 2^{40}, for example, $\varepsilon = 2^{20}$. We pair these keystreams with their respective final internal states. If we collect $2^{60.5}$-many of these, due to the birthday paradox, we are highly likely to find two keystreams with identical final internal states. Once the states are equal, keystreams of length ε produced from that point on will also be equal.

In short, if we collect large number of long MICKEY keystreams, we are bound to find two keystreams that started out differently, but ends the same, irrespective of whether the collected keystreams correspond to one or multiple keys. This may not qualify as an attack, but does show that using multiple long keystreams from MICKEY can be dangerous.

The huge number $2^{60.5}$ may give a wrong impression about the related threat level. Notice that any $(k+1)$-th iterate image state is also a k-th iterate image state. This observation implies that multiple shifts of a single keystream all qualify as a (slightly shorter) long keystream appearing in the above argument.

Let us try some more explicit numbers. One keystream of length 2^{40} can be shifted multiple times to produce $(2^{39} - \varepsilon)$-many keystreams, each of length (at least) $2^{39} + \varepsilon$. A gathering of 2^{22}-many 2^{40}-length keystreams can thus also be interpreted as about $2^{61} \sim 2^{22} \cdot (2^{39} - \varepsilon)$ keystreams of length $2^{39} + \varepsilon$. Since entropy of state after 2^{39} iterated state updates is about 122, one could say that we are likely to obtain a matching state.

In reality, more than 2^{22} base keystreams of length 2^{40} will need to be gathered before a match is found. This is because no matching will occur within the set of shifted sequences originating from the same base sequence. The exact analysis of these numbers seems to be complicated, but it is clear that the threat caused by state entropy loss cannot be dismissed lightly.

So far, we have stated the entropy loss as an undesirable property, rather than as an attack on MICKEY. But if one wants to consider the related data complexity, it should be noticed that the whole keystream need not be stored. Only the ending parts are used in looking for a match. The actual amount of data that needs processing is $C \cdot 2^{60.5} \sim 2^{68}$, where constant C is the number of keystream bits needed to reliably distinguish between different states. It is not totally clear as to which number should be taken as data complexity.

To develop a concrete attack scenario, it could be worthwhile to think about the case where an appropriate keystream set is pre-generated and compared with target online data.

Before ending this section, we add a few words of explanation on the use of EL versus $\overline{\mathrm{EL}}$. Some might believe that $\overline{\mathrm{EL}}$, which corresponds to iterated image point counts, should be considered in these arguments. The difference between the two measures is whether the uneven distribution within the image set was taken into account. Given two sets of the same size, a collision is more likely to appear in the one with a more uneven probability distribution. Hence, the distribution within the set does matter in these arguments, with the conclusion that $\mathrm{EL}(f)$, rather than $\overline{\mathrm{EL}}(f)$, is the correct measure to look at.

5 A small class of weak keys

Let us first quote from the MICKEY specification [4].

> There is a small class of arguably weak keys for MICKEY: namely, those (K, IV), pairs for which the state of R after loading is all zeroes. It is clear that, if an attacker assumes that this is the case, she can readily confirm her assumption and deduce the remainder of the generator state by analysing a short sequence of keystream. But, because this can be assumed to occur with probability roughly 2^{-80} — the same probability as for any guessed secret key to be correct — we do not think it necessary to prevent it (and so in the interests of efficiency we do not do so).

Notice that the all-zero state of register $\mathcal{R}$ is fixed under either value of its control bit.

Given the full specification for $\mathcal{S}$, one can easily check that the following is an $\mathcal{S}$-state which is fixed under either value of its control bit. The state is given in hexadecimal notation and the left end corresponds to s_0 while the right end corresponds to s_{79}.

`10f0 02a1 000b 853f fff8`

This fixed $\mathcal{S}$-state has been obtained as follows.[2] The clocking for register $\mathcal{S}$ works in a way so that fixing just two cell values s_{78} and s_{79} together with the control bit determines updated s_{79} cell value completely. Since we are looking for a fixed state, we try out all four possible values of (s_{78}, s_{79}) cell pairs and check if the updated s_{79} is equal to the chosen s_{79}. Only one of the four pairs satisfies this for both control bit values. Then successive choice of s_i cell value $(i < 78)$ determines updated s_{i+1} and we choose these so that the new s_{i+1} is equal to the old s_{i+1}, which has already been determined.

If register $\mathcal{S}$ is ever set to the above fixed $\mathcal{S}$-state, since the content of cell s_0 is 0, the keystream produced will be equal to whatever register $\mathcal{R}$, with its irregular clocking, produces. Those (key, IV) pairs, which sets register $\mathcal{S}$ to the above state after loading, forms a weak key class in the sense given by the designers. The probability of encountering one of these at random is roughly 2^{-80}.

We now have two weak key classes, and we can expect the probability of encountering one of these to be roughly[3] 2^{-79}, which is strictly greater than 2^{-80}. Deciding not to prevent either of these in view of efficiency will be harder to justify.

6 Conclusion

We have studied security aspects of a recent streamcipher proposal MICKEY. Three undesirable properties have been discovered.

[2] It turns out that the above $\mathcal{S}$-state is the unique state fixed under both control bit values.

[3] Exact expectation value would be $2^{-79} - 2^{-160}$.

1. Time-memory-data tradeoff is applicable with online complexity smaller than exhaustive key search. Pre-computation needed is more demanding than exhaustive search but this could be reachable in the near future.
2. The internal state loses entropy with repeated applications of the state update function, resulting in keystreams that start out differently but merge together towards the end.
3. There exists a small family of weak keys.

The first of these may be removed through internal state size increase, or with a more complicated output filter. But neither method would be desirable in view of efficiency. The second weakness seems more fundamental to the design of MICKEY and does not seem to be easy to fix. The third can be avoided easily with small extra cost.

It is plausible to say that none of these weaknesses are absolutely critical, but these must be taken into account by anyone intending to use MICKEY.

References

1. ECRYPT, Call for stream cipher primitives. Version 1.3, April. 2004. `http://www.ecrypt.eu.org/stream/`
2. ECRYPT, ECRYPT yearly report on algorithms and keysizes (2004). Version 1.1, March, 2005. Available from `http://www.ecrypt.org`.
3. S. H. Babbage, Improved exhaustive search attacks on stream ciphers. *European Convention on Security and Detection,* IEE Conference publication No. 408, pp. 161–166, IEE, 1995.
4. S. Babbage and M. Dodd, The stream cipher MICKEY (version 1). ECRYPT Stream Cipher Project Report 2005/015, 2005. Presented at Symmetric Key Encryption Workshop, Århus, Denmark, May, 2005. Available from `http://www.ecrypt.org/stream/`.
5. A. Biryukov and A. Shamir, Cryptanalytic time/memory/data tradeoffs for stream ciphers. *Asiacrypt 2000,* LNCS 1976, pp. 1–13, Springer-Verlag, 2000.
6. A. Biryukov, A. Shamir, and D. Wagner, Real time cryptanalysis of A5/1 on a PC. *FSE 2000*, LNCS 1978, pp. 1–18, Springer-Verlag, 2001.
7. P. Flajolet and A. M. Odlyzko, Random mapping statistics. *Eurocrypt '89*, LNCS 434, pp. 329–354, Springer-Verlag, 1990.
8. J. Dj. Golić, Cryptanalysis of alleged A5 stream cipher. *Eurocrypt'97,* LNCS 1233, pp. 239–255, Springer-Verlag, 1997.
9. M. E. Hellman, A cryptanalytic time-memory trade-off. *IEEE Trans. on Infor. Theory,* vol 26, pp. 401–406, 1980.

A General argument for TMD tradeoffs with BSW sampling

Let us work with BS-tradeoff with BSW sampling in a more general way to find the tradeoff point of minimal online complexity. Notation of Section 3 will be used.

Set $N' = |\mathcal{X}'| = |\mathcal{Y}'| = R{\cdot}N$ so that given D random target points, $D' = R{\cdot}D$ of them will belong to $\mathcal{Y}'$. The tradeoff curve will be

$$TM^2D'^2 = N'^2, \quad P = N'/D', \quad \text{with} \quad 1 \le D'^2 \le T,$$

or equivalently,

$$TM^2D^2 = N^2, \quad P = N/D, \quad D' = RD \quad \text{with} \quad 1 \le D'^2 \le T. \qquad (10)$$

A typical point on this curve is

$$T = M = D = N^{\frac{2}{5}}, \quad D' = R = N^{\frac{1}{5}}, \quad \text{and} \quad P = N^{\frac{3}{5}}. \qquad (11)$$

Compared with (3), we have achieved lowered time and memory complexity at the expense of higher data complexity. In all, the overall online complexity has dropped from $N^{\frac{1}{2}}$ to $N^{\frac{2}{5}}$. Notice that pre-computation time has decreased also.

We add a word of caution, as the complexities T and P of (3) are counted in terms of number of applications of f, where as those of (11) refer to applications of f', which is just f on smaller space, composed with h appearing in (6). So if the efficiency of function h is not comparable to that of f, the two time complexities cannot be compared in their raw form. Of course, similar remarks apply to M, with (11) at an advantage this time, but its importance is smaller.

B MICKEY-128

It is quite clear that arguments we've given for MICKEY should be applicable in much the same way to its 128-bit security cousin MICKEY-128. We briefly write down the respective results concerning TMD tradeoff and weak keys. Due to lack of time and computational power, we have not been able to study the entropy loss of MICKEY-128 state under its update function.

B.1 TMD tradeoff with BSW sampling

TMD tradeoff is applicable to MICKEY-128 in the same way as with MICKEY. Through BSW sampling, the search space size is reduced from 2^{256} to 2^{213}. Keystream segments starting with 43-many zero bits are the data we should be looking for.

A typical tradeoff point would be

$$T = 2^{106}, \quad M = 2^{107}, \quad D = 2^{53}, \quad \text{with} \quad P = 2^{160}.$$

The D above refers to filtered data, and the total data needed before filtering step is 2^{96}.

The tradeoff point giving minimal pre-computation time is

$$T = 2^{127}, \quad M = 2^{86}, \quad D = 2^{63.5}, \quad \text{with} \quad P = 2^{149.5}.$$

Again, the pre-filter data requirement is $2^{106.5}$.

B.2 A small class of weak keys

There exists one fixed $\mathcal{S}$-state which remains fixed under either value of the control bit. It is given below in hexadecimal notation with the left end corresponding to s_0.

```
05ff d071 d020 c3fc 0c1c 037f 1f3f ef98
```

This $\mathcal{S}$-state gives a small family of weak keys which may be encountered at random with probability 2^{-128}.