Collision Attacks based on the Entropy Loss caused by Random Functions

Andrea Röck

Projet CODES, INRIA Paris-Rocquencourt, France
http://www-rocq.inria.fr/codes/
andrea.roeck@inria.fr

Abstract. Replacing random permutations by random functions for the update of a stream cipher introduce the problem of entropy loss. To assess the security of such a design, we need to evaluate the entropy of the inner state. We propose a new approximation of the entropy for a limited number of iterations. Subsequently, we discuss two collision attacks which are based on the entropy loss. We provide a detailed analysis of the complexity of those attacks and show an improvement using distinguished points.

1 Introduction

Recently, several stream ciphers have been proposed with a non-bijective update function. Moreover, in some cases the update function seems to behave like a random function as for the MICKEY stream cipher [BD05]. Using a random function instead of a random permutation induces an entropy loss in the state. An attacker might exploit this fact to mount an attack. We will examine the effectiveness of such an approach. At first we introduce the model with which we are going to work.

Stream cipher model The classical model of an additive synchronous stream cipher (Fig. 1) is composed of an internal state updated by applying a function Φ. Then a filter function is used to extract the keystream bits from the internal state. To receive the ciphertext we combine the keystream with the plaintext.

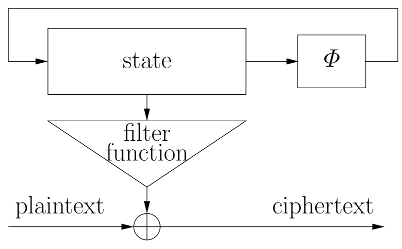

Fig. 1. Model of a simple stream cipher

The particularity of our model is that Φ is a random mapping which allows us to make some statistical statements about the properties of the stream cipher.

Definition 1. *Let $\mathcal{F}_n = \{\varphi \mid \varphi : \Omega_n \to \Omega_n\}$ be the set of all functions which map a set $\Omega_n = \{\omega_1, \omega_2, \ldots, \omega_n\}$ of n elements onto itself. We say that Φ is a* random function *or a* random mapping *if it takes each value $\varphi \in \mathcal{F}_n$ with the same probability $Pr[\Phi = \varphi] = 1/n^n$.*

For an extended definition of a random function we refer to the book of Kolchin [Kol86].

Let S_k be the random variable denoting the value of the state after k iterations of Φ, for $k \geq 0$. From the model in Fig. 1 we see that $S_k = \Phi(S_{k-1})$ where the value of Φ is the same for all iterations $k > 1$. The probability distribution of the initial state S_0 is $\{p_i\}_{i=1}^n$ such that

$$p_i = Pr[S_0 = \omega_i].$$

If we don't state otherwise, we assume a uniform distribution thus $p_i = 1/n$ for all $1 \leq i \leq n$. By

$$p_i^{\Phi}(k) = Pr[\Phi^{(k)}(S_0) = \omega_i]$$

we describe the probability of the state being ω_i after applying k times Φ on the initial state S_0. If we write only $p_i^{\varphi}(k)$ we mean the same probability but for a specific function $\varphi \in \mathcal{F}_n$. The notation above allows us to define the entropy of the state after k iterations of Φ

$$H_k^{\Phi} = \sum_{i=1}^{n} p_i^{\Phi}(k) \log_2 \left(\frac{1}{p_i^{\Phi}(k)} \right).$$

In this article we are interested in expectations where the average is taken over all functions $\varphi \in \mathcal{F}_n$. In the following, expectations denoted in bold. For instance, the formula:

$$\mathbf{H_k} = \mathbf{E}(H_k^{\Phi})$$

denotes the expected state entropy after k iterations.

The subsequent article is divided in two main sections. In Section 2, we discuss different possibilities to estimate the state entropy. We give a short overview of previous results from [FO90] and [HK05] in Section 2.1. Subsequently in Section 2.2, we present a new estimation which is, for small numbers of iterations, more precise than the previous one. In Section 3, we discuss two collision attacks against Mickey version 1 [BD05] presented in [HK05]. To evaluate the cost of these attacks we consider the space complexity, the query complexity and the number of different initial states needed. By the space complexity we mean the size of the memory needed, by the query complexity we mean the number of times we have to apply the update function during the attack. For the first attack, we show that we only gain a factor on the space complexity by increasing the query complexity by the same factor. For the second attack, we demonstrate that, contrary to what is expected from the results in [HK05], the complexities are equivalent to a direct collision search in the starting values. In the end we present a new variant of these attacks which allows to reduce the space complexity, however the query complexity remains the same.

2 Estimation of entropy

The entropy is a measure for the unpredictability. An entropy loss in the state facilitates the guessing of the state for an adversary. In this chapter, we therefore discuss different approaches to estimation the expected entropy of the inner state.

2.1 Previous works

Flajolet and Odlyzko provide, in [FO90], a wide range of parameters of random functions by analyzing their functional graph. A functional graph of a specific function φ is a graph which has a directed edge from vertex x to vertex y if and only if $\varphi(x) = y$. An example for $\varphi(x) = x^2 + 2 \pmod{20}$ can be seen in Fig. 2. For functions on a finite set of elements, such a graph consists of one or more separated components, where each component is build by a cycle of trees, i.e. the nodes in the cycle are the root of a tree.

To find a the expected value of a given parameter of a random function, Flajolet and Odlyzko construct the generating function of the functional graph associated with this parameter. Subsequently, they obtain an asymptotic value of the expectation by means of a singularity analysis of the generating function. All asymptotic values are for n going to $+\infty$. In the following we present some example of the examined parameters. The maximal tail length is, for each graph, the maximal number of steps before reaching a cycle. An r–node is a node in the graph with exactly r incoming nodes which is equivalent to a preimage of size r. By the image points we mean all points in the graph that are reachable after k iterations of the function. The asymptotic values of these parameters are:

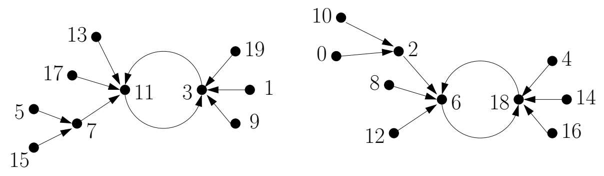

Fig. 2. Example of a functional graph for $\varphi(x) = x^2 + 2 \pmod{20}$

- the expected number of cycle point $\mathbf{cp}(n) \sim \sqrt{\pi n/2}$,
- the expected maximal tail length $\mathbf{mt}(n) \sim \sqrt{\pi n/8}$,
- the expected number of r–nodes $\mathbf{rn}(n, r) \sim \frac{n}{r!e}$ and
- the expected number of image points after k iterations $\mathbf{ip}(n, k) \sim n(1 - \tau_k)$ where $\tau_0 = 0$ and $\tau_{k+1} = e^{-1+\tau_k}$.

For all these values, the expectation is taken over all functions in $\mathcal{F}_n$.

In [HK05], Hong and Kim use the expected number of image points to give an upper bound of the state entropy after k iterations of a random function. They utilize the fact that the entropy is always smaller or equal than the logarithm of the number of points with probability larger than zero. After a finite number of steps, each point in the functional graph will reach a cycle, thus the number of image points can never drop under the number of cycle points. Therefore, the upper bound of the estimated entropy of the internal state

$$\mathbf{H_k} \leq \log_2(n) + \log_2(1 - \tau_k) \tag{1}$$

is valid only as long as $\mathbf{ip}(n, k) > \mathbf{cp}(n)$. We see that for this bound the loss of entropy only depends on k and not on n.

In Fig. 3 we compare, for $n = 2^{16}$, the values of this bound with the empirical gained average of the state entropy. To calculate this value we chose 10^4 functions, by means of the HAVEGE random number

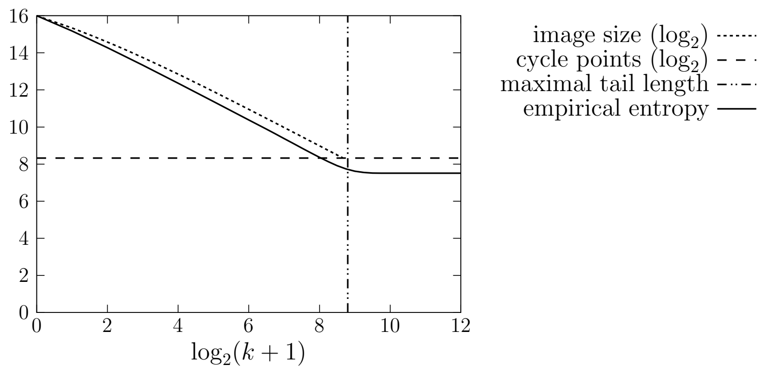

Fig. 3. Upper bound and empirical average of the entropy for $n = 2^{16}$

generator [SS03], and calculated the average entropy under the assumption of a uniform distribution of the initial state. Even if n is not very big, it is sufficient to understand the relation between the different factors. We can see in the graph that if k stays smaller than $\mathbf{mt}(n)$ this bound stays valid and does not drop under $\log_2(\mathbf{cp}(n))$.

2.2 New entropy estimation

The expected number of image points provides only an upper bound for the expected entropy. We found a more precise estimation by employing the methods and properties stated in [FO90].
For a given function $\varphi \in \mathcal{F}_n$, let ω_i be a node with r incoming nodes (an r–node). The idea is that this is equivalent to the fact that ω_i is produced by exactly r different starting values after one iteration. Thus, if the initial distribution of the state is uniform, this node has the probability $p_i^{\varphi}(1) = r/n$.
Let $\mathbf{rn_k}(n,r)$ denote the expected number of points that are reached by exactly r nodes after k iterations. Then in the case of a uniform initial distribution the expected entropy is

$$\mathbf{H_k} = \sum_{r=1}^{n} \mathbf{rn_k}(n,r) \frac{r}{n} \log_2 \frac{n}{r} = \log_2(n) - \sum_{r=1}^{n} \mathbf{rn_k}(n,r)\ \frac{r}{n}\ \log_2(r). \tag{2}$$

The equality at the right hand side is due to the fact that $\sum_{r=1}^{n} r\ \mathbf{rn_k}(n,r) = n$ since each of the n points has exactly one successor is thus thus counted in one $\mathbf{rn_k}(n,r)$.
To approximate $\mathbf{rn_k}(n,r)$ for one iteration, we use directly the results for the expected number of r–nodes, given in [FO90], since $\mathbf{rn_1}(n,r) = \mathbf{rn}(n,k) \sim \frac{n}{r!e}$. For more than one iteration we need to define a new parameter.
By a cycle–nodes we denote the nodes in the functional graph which are part of a cycle and thus the root of a tree. The other nodes we denote by tree–nodes. We have to make this distinction, since for a cycle–node we do not only have the incoming nodes from its associated tree but we also the incoming nodes from the previous position in the cycle.
To define our new parameter we use the fact that a tree–node which is exactly reached by r nodes after k iterations has

- j incoming nodes, where $1 \leq j \leq r$. Each of these nodes is root of tree with depth larger or equal to $k-1$ and is reached respectively by $i_1, \ldots, i_j$ nodes after $k-1$ iterations such that $i_1 + \cdots + i_j = r$.
- an arbitrary number of incoming nodes which are trees of depth strictly smaller than $k-1$.

We are now going to build a generating function where we mark the desired parameter by the variable u. To do so we need some helping functions:

- $f_1(k,z)$ defines the generating function of a set of trees with depth smaller than $k-1$. We get
$$f_1(k,z) = \begin{cases} e^z & \text{for } k=1 \\ e^{z\ f_1(k-1,z)} & \text{otherwise.} \end{cases}$$
- $Par(r)$ is the set of all partitions of the integer r.
  For example for $r=4$ we get $Par(4) = \{[1,1,1,1],[1,1,2],[2,2],[1,3],[4]\}$.
- $f_2([i_1,\ldots,i_j])$ is a correction term.
  If there are some $i_{x_1},\ldots,i_{x_\ell}$ with $1 \leq x_1 < \cdots < x_\ell \leq j$ and $i_{x_1} = \cdots = i_{x_\ell}$ we have to multiply by a factor $1/\ell!$ to compensate this repeated appearance, e.g. $f_2([1,1,1,1,2,2,3]) = \frac{1}{4!2!1!}$.

We then define the function

$$c_k(r,z) = \begin{cases} z/r! & \text{for } k=1 \\ z\ f_1(k,z) \displaystyle\sum_{[i_1,\ldots,i_j]\in Par(r)} c_{k-1}(i_1,z)\ \cdots\ c_{k-1}(i_j,z)\ f_2([i_1,\ldots,i_j]) & \text{otherwise.} \end{cases} \tag{3}$$

and write the generating function for a tree as

$$t_{r,k}(u,z) = z\ e^{t_{r,k}(u,z)} + (u-1)\ t_{r,k}(u,z)^r\ c_k(r,z),$$

where u marks the nodes in the tree which are reached by r nodes in k steps. It is easy to see that if we set $u=1$, which mean we ignore our marked parameter, we get the general generating function for a tree

$$t(z) = t_{r,k}(1,z) = z\ e^{t(z)}.$$

By applying the properties that a graph of a random function is a *set of components* where each component is a *cycle of trees* we get the generating function for a general functional graph

$$\xi_{r,k}(u,z) = \frac{1}{1 - t_{r,k}(u,z)}.$$

As described in [FO90] the generating function for the marked parameter is

$$\begin{aligned}\Xi_{r,k}(z) &= \frac{\partial}{\partial_u}\xi_{r,k}(u,z)\Big|_{u=1}\\ &= \frac{t(z)\ c_k(r,z)}{1-t(z)^3}.\end{aligned}$$

By a singularity analysis of the generating function for $z \to e^{-1}$ we finally obtain the approximation of the expected value

$$\mathbf{E}(\xi_{r,k,n}|\mathcal{F}_n) \sim n\ c_k(r,e^{-1}).$$

In this equation $\xi_{r,k,n}[\varphi]$ denotes the number of nodes which are reachable by r tree–nodes after k iterations of a specific φ and the expectation is taken over all values $\varphi \in \mathcal{F}_n$. We are going to use

$$\mathbf{rn_k}(n,r) \sim \mathbf{E}(\xi_{r,k,n}|\mathcal{F}_n)$$

ignoring that for a cycle–node we have to count not only the incoming tree–nodes but also the incoming cycle–node. However, for large n the number of cycle–nodes is $\sqrt{\pi n/2}$ which is only a small fraction of all nodes. Therefore the introduced error is negligible if k, and thus the number of nodes which reach a cycle is not too large. In the end we use

$$\mathbf{rn_k}(n,r) \sim n\ c_k(r,e^{-1}). \tag{4}$$

The calculation of $c_k(r,e^{-1})$ as defined in (3) is not very practical. In this paragraph we will show a more efficient method. For simplicity we write in the following $c_k(r,e^{-1}) = c_k(r)$ and $f_1(k,e^{-1}) = f_1(k)$. We define the new value $D(k,r,m)$ by

$$D(k,r,m) = \sum_{[i_1,\ldots,i_j]\in Par_{\geq m}(r)} c_{k-1}(i_1)\cdots c_{k-1}(i_j) f_2([i_1,\ldots,i_j])$$

where $Par_{\geq m}(r)$ is the set of all partitions of the integer r such that for each $[i_1,\ldots,i_j] \in Par_{\geq m}(r)$ must hold that $i_\ell \geq m$ for all $1 \leq \ell \leq j$. We can then give the following recursive definition of $c_k(r)$ and $D(k,r,m)$

$$c_k(r) = \begin{cases} \frac{1}{r!e} & \text{if } k=1\\ D(k,r,1)f_1(k)\frac{1}{e} & \text{otherwise}\end{cases}$$

$$D(k,r,m) = \begin{cases} 1 & \text{if } r=0\\ 0 & \text{if } 0<r<m\\ \sum_{u=0}^{\lfloor r/m\rfloor} \frac{c_{k-1}(m)^u}{u!} D(k,r-m\ u,m+1) & \text{otherwise.}\end{cases}$$

This construction allows us to calculate all the value $c_k(r)$ for $k \leq K$ and $r \leq R$ in a time complexity of $O\left(K\ R^2\log(R)\right)$.
Lets go back to the entropy estimation in (2). By using (4) we can write

$$\mathbf{H_k} \sim \underbrace{\log_2(n) - \sum_{r=1}^{R} c_k(r)\ r\ \log_2(r)}_{(a)} \underbrace{- \sum_{r=R+1}^{n} c_k(r)\ r\ \log_2(r)}_{(b)},$$

where (a) represents an estimation of the entropy loss which does not depend on n and (b) is an error term. In Fig. 4 we see that the value $c_k(r)\ r\ \log_2(r)$ decreases fast with growing r. However, for larger k this decrease becomes slower. We thus see that if we want (b) to be negligible for larger k we also need a larger value for R. In Table 1 we compare our entropy estimator

$$\mathbf{H_k} \sim H_k(R) = \log_2(n) - \sum_{r=1}^{R} c_k(r)\ r\ \log_2(r) \tag{5}$$

with the estimated lower bound of the loss given by the expected number of image points (1) and the empirical results from the experiment presented in Fig. 3. We can see that for small k we reach a much

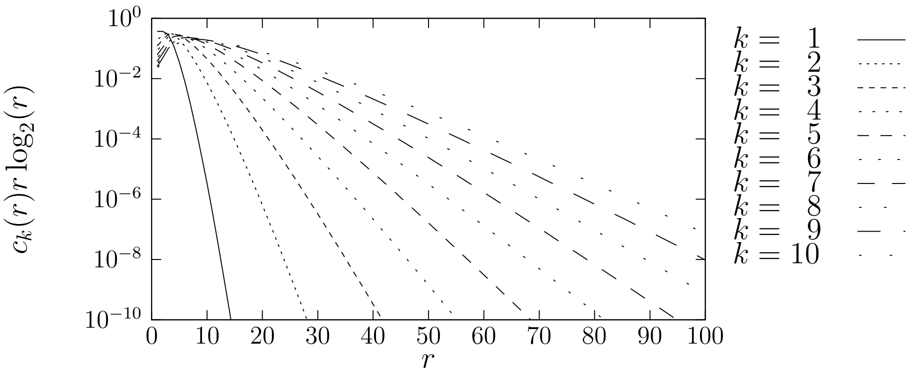

Fig. 4. The course of $c_k(r)\ r\ \log_2(r)$ for different values of k and r.

k		0	1	2	3	$\cdots$	10	$\cdots$	50	$\cdots$	100
empirical data, $n = 2^{16}$		0.0000	0.8273	1.3458	1.7254	$\cdots$	3.1130	$\cdots$	5.2937	$\cdots$	6.2529
image points (1)		0.0000	0.6617	1.0938	1.4186	$\cdots$	2.6599	$\cdots$	4.7312	$\cdots$	5.6913
$H_k(R)$**, (5)**	$R = 50$	0.0000	0.8272	1.3457	1.7254	$\cdots$	3.1084	$\cdots$	2.6894	$\cdots$	1.2524
	$R = 200$	0.0000	0.8272	1.3457	1.7254	$\cdots$	3.1129	$\cdots$	5.2661	$\cdots$	5.5172
	$R = 1000$	0.0000	0.8272	1.3457	1.7254	$\cdots$	3.1129	$\cdots$	5.2918	$\cdots$	6.2729

Table 1. Comparison of different methods to estimate the entropy loss.

better approximation than the upper bound (1). However, for bigger values of k we also need a bigger R and, thus, this method gets computationally expensive.

Our approach can be extended to estimate the entropy not only in the uniform case of the initial distribution but also for any arbitrary distribution $P = \{p_1, p_2, \ldots, p_n\}$. The expectation of the entropy is taken over all $\varphi \in \mathcal{F}_n$, thus every index tuple $1 \leq i_1 < i_2 < \cdots < i_r \leq n$ of size r occurs the same number of times. In total there are $n^n\ \mathbf{rn_k}(n, r)$ number of points that are reached by r nodes after k iterations and there are $\binom{n}{r}$ possible index tuples of size r, thus each element $(p_{i_1} + \cdots + p_{i_r}) \log_2 \frac{1}{p_{i_1}+\cdots+p_{i_r}}$ appears $\frac{n^n\ \mathbf{rn_k}(n,r)}{\binom{n}{r}}$ times in the sum. By dividing by n^n we get the average entropy

$$\mathbf{H_k^P} = \sum_{r=1}^{n} \mathbf{rn_k}(n, r) \frac{1}{\binom{n}{r}} \sum_{1 \leq i_1 < \cdots < i_r \leq n} (p_{i_1} + \cdots + p_{i_r}) \log_2 \frac{1}{p_{i_1} + \cdots + p_{i_r}}. \tag{6}$$

Finally, we want to consider a further special case. Let us assume we know that from the n possible initial values only m occur with probability exactly $\frac{1}{m}$. In this case we get

$$\mathbf{H_k^{P_m}} = \sum_{r=1}^{n} \mathbf{rn_k}(n, r) \frac{1}{\binom{n}{r}} \sum_{j=0}^{r} \binom{m}{j} \binom{n-m}{r-j} \frac{j}{m} \log_2 \frac{m}{j}. \tag{7}$$

In (6) as well as in (7) we can use (4) to approximate the value of $\mathbf{rn_k}(n, r)$ and thus the values $\mathbf{H_k^P}$ and $\mathbf{H_k^{P_m}}$ respectively.

In this chapter we presented a new entropy estimator. We could show that if the number of iterations is not too big, it is much more precise than the upper bound given by the image size. In addition, the same method can be used for any arbitrary initial distribution.

3 Collisions attacks

In [HK05], Hong and Kim state that the update function of the first version of the MICKEY stream cipher [BD05] behaves almost like a random function with regard to the entropy loss and the number of image points. They present two collision attacks, however without detailed complexity analysis, which try to exploit the entropy loss of the state. These attacks are directly applicable on our model.
Let us take two different initial states S_0 and S'_0 and apply the same function iteratively onto both of them. We speak about a collision if there is there exists k and k' such that $S_k = S_{k'}$, for $k \neq k'$, or $S_k = S'_{k'}$ for any arbitrary pair k, k'. The idea of Hong and Kim was that a reduced entropy leads to an increased probability of a collision. Once we have found a collision we know that the subsequent output streams are identical. Due to the birthday paradox, we assume that with an entropy of m-bits we reach a collision, with high probability, by choosing $2^{\frac{m}{2}}$ different states.
In this section we present the two attacks introduced in [HK05]. For each of them we provide a detailed complexity analysis where we examine the query and the space complexity as well as the number of necessary starting points to achieve a successful attack with high probability. By the query complexity we mean the number of all states produced by the cipher during the attack which is equivalent to the number of times the updated function is applied. By the space complexity we mean the number of states we have to store such that we can search for a collision within them. Each time we compare the results to the attempt of finding a collision directly within the initial states which has a space and query complexity of $\sim \sqrt{n}$.
In all these attacks we do not consider how many output bits we would really need to store and to recognize a collision, since this value depends on the specific filter function used. In the example of MICKEY, Hong and Kim states that they need about 2^8 bits.

3.1 States after *k* iterations

The first attack of Hong and Kim takes randomly m different initial states, applies k times the same instance of Φ on each of them and searches a collision in the m resulting states. Using (1) we know that the average entropy after k iterations is less than $\log_2(n) + \log_2(1 - \tau_k)$. Hong and Kim conjecture, based on experimental results, that this is about the same as $\log_2(n) - \log_2(k) + 1$. Thus, with high probability we find a collision if $m > 2^{(\log_2(n) - \log_2(k)+1)/2} = \sqrt{2n/k}$.
This attack stores only the last value of the iterations and searches for a collision within this set. This leads to a space complexity of $m \sim \sqrt{2n/k}$ for large enough k. However, we have to apply k times φ on each of the chosen initial states, which results in a query complexity of $mk \sim \sqrt{2kn}$. This means that we increase the query complexity by the same factor as we decrease the space complexity and the number of starting points.
The question remains, if there exists any circumstances under which we can use this approach without increasing the query complexity. The answer is yes if the stream cipher uses a set of update functions which loose more than $2\log_2(k)$ bits of entropy after k iterations. Such a characteristic would imply that we do not use random functions to update our state, since they have different properties as we have seen before. However, the principle of the attack stays the same.

3.2 Including intermediate states

The second attack in [HK05] is equivalent to applying $2k-1$ times the same instance of φ on m different starting values and searching for a collision in all intermediate states from the k-th up to the $(2k-1)$-th iteration. Since after $k-1$ iterations we have about $\log_2(n) - \log_2(k) + 1$ bits of entropy, Hong and Kim assume that we need a m such that $mk \sim \sqrt{n/k}$. They state that this result would be a bit too optimistic since collisions within a row normally do not appear in a practical stream cipher. However, they claim that this approach still represents a realistic threat for the MICKEY stream cipher. We will show that, contrary to their conjecture, this attack has about the same complexities as a direct collision search in the starting points.
Let us take all the $2km$ intermediate states for the $2k-1$ iterations. Let $Pr[A]$ define the probability that there is no collision in the all the $2km$ intermediate states. If there is no collision in this set then there is also no collision in the km states considered by the attack. Thus, the probability of a successful

attack is even smaller than $1 - Pr[A]$. Let $Pr[I]$ be the probability of no collision in the m initial points. We can see directly that

$$\begin{aligned} Pr[A] &= Pr[A \cap I] \\ &= Pr[A|I]\ Pr[I]. \end{aligned}$$

Let us assume that we have chosen m different initial values. This happens with a probability of

$$Pr[I] = \frac{\binom{n}{m} m!}{n^m}. \tag{8}$$

In this case we have

- n^n different random functions, where each of them creates
- $\binom{n}{m} m!$ different tables. Each table can be produced more than once. There exists
- $n\ (n-1)\dots(n-2km+1)$ different tables that contain no collisions. Each of them can be generated by
- $n^{n-(2k-1)m}$ different functions, since a table determines already $(2k-1)m$ positions of φ.

Thus, we get the probability

$$Pr[A|I] = \frac{n\ (n-1)\dots(n-2km+1)\ n^{n-(2k-1)m}}{n^n \binom{n}{m} m!} \tag{9}$$

for $m > 0$ and $2km \leq n$. By combining (8) and (9) we get

$$Pr[A] = \frac{n(n-1)\dots(n-2km+1)}{n^{2km}} \tag{10}$$

where the probability is taken over all functions $\varphi \in \mathcal{F}_n$ and all possible starting values. This is exactly the probability of no collision in $2km$ random points which means that we need at least an m such that $2km \sim \sqrt{n}$. This leads to a query complexity of $\sim \sqrt{n}$ and a space complexity of $\sim \sqrt{n}/2$.

3.3 Improvement with distinguished points

By applying the known technique of distinguished points we can reduce the space complexity in the second attack, however the query complexity stays the same.
By *distinguished points* (DPs) we mean a subset of Ω_n which is distinguished by a certain property, e.g. by a specific number of 0's in the most significant bits. In our new variant of the second attack we iterate φ in each row up to the moment where we reach a DP. In this case we stop and store the DP. If we do not reach a DP after MAX iterations we stop as well but we store nothing. If there was a collision in any of the states in the rows where we reached a DP, the subsequent states would be the same and we would stop with the same DP. Thus it is sufficient to search for a collision in the final DPs.
Let d be the number of distinguished points in Ω_n. We assume that the ratio $c = \frac{d}{n}$ is large enough that with a very high probability we reach a DP before the end of the cycle in the functional graph. This means that the average number of iterations before arriving at a DP is much smaller than the expected length of the tail and the cycle together (which would be about $\sqrt{\frac{\pi\, n}{2}}$ due to [FO90]). We assume that in this case the average length of each row would be in the range of $1/c$ like in the case of random points. We also suppose that that we need about $m/c \sim \sqrt{n}$ query points to find a collision, like in the previous case. This leads to a query complexity of $\sim \sqrt{n}$ and a space complexity of only $\sim c\sqrt{n}$. Empirical results for example for $n = 2^{20}$, $0.7 \leq \frac{\log_2(d)}{\log_2(n)} \leq 1$ and $MAX = \sqrt{n}$ confirm our assumptions.

A summary of the complexities of all attacks can be found in Table 2. We see that either the space complexity is still in the magnitude of $\sim \sqrt{n}$ or it is reduced by the same factor as the query complexity is increased.

attack	**# starting points**	**space complexity**	**query complexity**
after k iterations, 3.1	$\sim \sqrt{2n/k}$	$\sim \sqrt{2n/k}$	$\sim \sqrt{2kn}$
with interm. states, 3.2	$\sim \sqrt{n}/2k$	$\sim \sqrt{n}/2$	$\sim \sqrt{n}$
with DPs, 3.3	$\sim c\sqrt{n}$	$\sim c\sqrt{n}$	$\sim \sqrt{n}$

Table 2. Complexities of attacks

4 Conclusion

We have introduced a new method to estimate the state entropy of a stream cipher which uses a random update function. This estimator is based on the number of points that produce the same value after k iterations. Its computation is expensive for large numbers of iterations, however, for small numbers it is much more precise than the upper bound given by the number of image points.
We have also examined the two collision attacks proposed in [HK05]. We pointed out that the first attack improves the space complexity at the cost of significantly increasing the query complexity. We proved that the complexity of the second attack is equivalent to the complexity of a direct collision search in the starting values. In addition we proposed a new attack which reduces the space complexity but leaving the query complexity constant. The use of random functions in a stream cipher introduces the problem of entropy loss. However, the proposed attacks based on this weakness are less effective than expected. Thus, we saw that the argument alone that a stream cipher uses a random function is not enough to threaten it due to a collision attack based on the entropy loss.

References

[BD05] S. Babbage and M. Dodd. The stream cipher MICKEY (version 1). eSTREAM, ECRYPT Stream Cipher Project, Report 2005/015, 2005. `http://www.ecrypt.eu.org/stream`.

[FO90] P. Flajolet and A. M. Odlyzko. Random mapping statistics. *Advances in Cryptology, Proc. Eurocrypt'98*, 434:329–354, 1990.

[HK05] J. Hong and W. H. Kim. TMD-tradeoff and state entropy loss considerations of streamcipher MICKEY. In *INDOCRYPT*, pages 169–182, 2005.

[Kol86] V. F. Kolchin. *Random Mappings.* Optimization Software, Inc, 1986.

[SS03] A. Seznec and N Sendrier. HAVEGE: A user-level software heuristic for generating empirically strong random numbers. *ACM Trans. Model. Comput. Simul.*, 13(4):334–346, 2003.