State Space Cryptanalysis of The MICKEY Cipher

Tor Helleseth*, Cees J.A. Jansen†, Oleksandr Kazymyrov* and Alexander Kholosha*
*The Selmer Center, Department of Informatics
University of Bergen, P.O. Box 7800, N-5020 Bergen, Norway
Email: {Tor.Helleseth, Oleksandr.Kazymyrov, Alexander.Kholosha}@ii.uib.no
†DeltaCrypto BV
Riebeeckstraat 10, 5684ej Best, The Netherlands
E-mail: cees.jansen@deltacrypto.nl

***Abstract*—In this paper, we consider the key-stream generator MICKEY, whose internal state splits into two parts that are updated both linearly and nonlinearly while clocking the generator. These state update functions also depend on the internal state of the registers, which perform the so-called self-mutual control. We suggest several attack scenarios based on the reverse clocking of the generator and analysis of the acquired backward states tree. Furthermore, we show meet-in-the-middle attack can be applied while simultaneously allowing the generation of shifted key streams for different pairs of keys and initialization vectors. In practice, our theoretical results are verified by extensive computations.**

I. Introduction

Nonlinear feedback shift registers (NFSRs) prove to be an extremely promising building block for key-stream generators. Such registers allow efficient hardware implementation and provide nonlinearity so crucial to the security of such a generator. However, the behavior of such nonlinear components is poorly understood, which, in turn, results in a lack of criteria for selecting parameters that directly affect security. The reason for this poor understanding is the difficulty inherent in the analysis of generators involving NFSRs. Despite widespread use of nonlinear registers in modern stream ciphers, security analysis of such constructions is mostly conjectural and lacks formal estimates and proofs.

Due to the specific application of symmetric ciphers, generators require a large period and linear complexity of key streams [1]. To achieve this, designers of stream ciphers often combine linear and nonlinear registers. It is believed that the linear part should guarantee the required period and the nonlinear part should increase linear complexity. Moreover, the update function of both registers may involve the state of the partner register and implement so-called 'mutual control'. In addition to a secret key, modern key-stream generators, often incorporate the public initialization value (IV), which allows the use of the same key for multiple encryptions. Prior to key-stream generation, the generator is clocked a number of times in a preclock phase and, provided that the key and IV are not set up directly into the register states, can also be clocked during key/IV-loading. Design principles for these modes differ, but in order to achieve the high efficiency in hardware implementation, the design of these modes are often similar.

The Mutual Irregular Clocking KEYstream generator (MICKEY) is an example of the generators described above. In article [2] Hong and Kim noted that the first version of MICKEY (MICKEY v1) is potentially vulnerable to time-memory-data (TMD) tradeoff attack [4], [9]. Based on this research, developers of the current (second) version of MICKEY (MICKEY v2) have modified the algorithm and shown that the new version of the cipher is resistant to this type of attack. A different part [2] deals with a comparative analysis of MICKEY v1's update function and random function as well as an estimation of collisions in a transition states graph. We are continuing the research in this direction and applying a similar technique to MICKEY v2. In addition to employing a method from [2], our approach allows us to theoretically calculate the states of whole backward states tree using update functions of registers. We have thus acquired theoretical results that give us an opportunity to verify data from [2] in different way. A procedure of generating a backward states tree results in an encryption key with a complexity lower than exhaustive search. We also present a new method for generating non-identical pairs of keys and IVs. These pairs allow us to generate the key streams that are shifted by predetermined bits. The theoretical results are also confirmed by practical calculations, and take into account all peculiarities of the most recent version of MICKEY.

The paper is organized as follows. Section II introduces the general model of the key-stream generator under analysis and underlines attack scenarios. The remaining part of the paper contains an illustration of our approach to stream cipher MICKEY v2 [**?**], [5], which is in the portfolio of hardware-oriented eStream ciphers [6]. We begin with the description of MICKEY in subsection III-A. Further in subsections III-B-III-D, we theoretically compute probabilities of branch points for all possible degrees in the state transition graph for MICKEY when run in the key-stream generation, preclock and key/IV-lode modes. The recorded values from practical verification of these results are presented in subsection III-E.

II. Key-Stream Generation Model and Attack Scenarios

There is evidence to show that, in general, cryptanalysis of stream ciphers for recovering a key is a two-step process. The first step is to retrieve the initial state of registers based

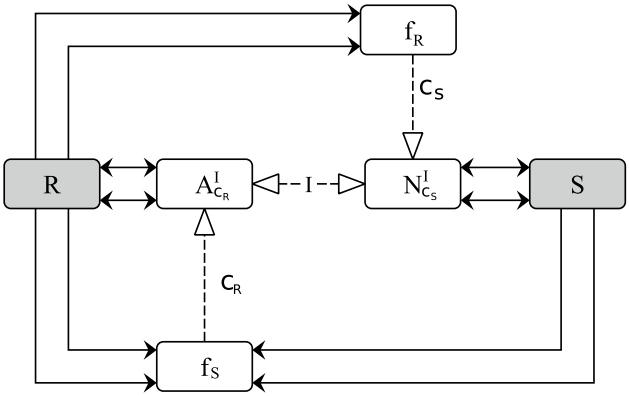

Fig. 1. Block Model of a Self-Mutual Key Generator

on encrypted data or key stream. However, the resultant state guarantees recovering messages from one session only, i.e. for one IV. In most cases, the complexity of this step is very high and often close to the exhaustive search. In order to gain complete control over the channel it is necessary to obtain a key. Therefore, an additional attack is performed at the next stage to recover a key based on the register values from the previous step. The complexity of this step is usually significantly less than the first one.

An example of such a stage is an attack on the cipher A5/1 [7], which is implemented in mobile systems of the second generation. The rainbow attack [8] is performed on the first stage and allows us to obtain the state after preclock mode. But in order to recover the key, a different type of attack was developed, a full description of which is provided in [9]. In this paper, we focus on the second stage. Thus we assume that the state of the registers is known.

We consider a particular design of a key-stream generator that consists of two registers (R and S) of length n_R and n_S with the affine ($A^I_{C_R}$) and nonlinear ($N^I_{C_S}$) update function respectively. We also assume that the concrete state update function of registers R and S applied at a certain stage is chosen according to the current state of R and S. This can be seen as self-mutual control of register clocking. To this end, we define vectorial Boolean functions f_R and f_S of $n_S + n_R$ bits each. The outputs of these functions, which in general consist of a few bits, are denoted by c_S and c_R. Figure 1 shows the design being analyzed in this paper. We also assume that the key and IV are inserted bit-by-bit into the key-stream generator run in the key/IV-load modes (meaning that this bit, denoted I in Figure 1, in some way affects the state update function). The generator is run in the following way: IV-load, key-load, preclock and key-stream generation. The initial state of the generator is always set to a constant.

The considered key-stream generator run in a key-stream generation mode can be seen as an autonomous finite state machine and, thus, the period of the produced output cannot exceed the state space size, i.e. $2^{n_R+n_S}$. However, even if both registers are nonsingular, the way they are updated results in self-mutual control which may cause them to behave in a singular fashion individually, exhibiting orphan states (states with no predecessors) and branch point states. This indicates a reduction in period (since in the maximum period, all states are connected into a full cycle that has no orphan states or branch points). Moreover, analysis of the transitions graph implies many other interesting properties crucial for security and helpful in cryptanalysis. In what follows, by the *degree* of a branch point we understand the number of incoming edges to the branching node in the state transition graph.

In the **first scenario**, assume that an attacker, in some way, knows the internal state of the key-stream generator at some stage during preclock or key-stream generation and knows exactly how many steps the generator was stepped to end up in this state. Then the generator is stepped back appropriately to stop right before preclock (i.e., right after key/IV-load). In this process, the generator behaves as an autonomous finite state machine with a few options for the preceding state. It is necessary to consider how many candidate states we end up with. This number is equal to the number of leaves in the top level of the tree that represents state transitions (backwards states tree). Amazingly enough, some key-stream generators demonstrate only polynomial growth in the number of leaves contained in each level of the tree (as opposed to the exponential growth demonstrated, for instance, in the full binary tree). This allows us to perform computations stepping several hundred clocks backwards. Obviously, the crucial characteristic of the transition tree is the average branch number, which is defined by probabilities of branch points of different degrees. Clearly, if the expected value is close to 1, then the tree will grow much slower than 2^n.

In the **second scenario**, we assume that the attacker learns the internal state of the key-stream generator at some stage during key-load (preceding preclock) and knows exactly how many steps the generator was stepped to end up in this state. Here, while clocking the generator backwards, we have additional uncertainty in the key-bit that affects the state update function. Branch points in the corresponding tree have a higher degree, and edges are labeled with the appropriate value of the key-bit. Hence, elimination of orphan states can lead to the unique identification of key bits.

The second scenario can be extended as follows. Assume that the attacker learns the internal state of the key-stream generator after only a few steps after the generator has been run in the key-load mode. This number of steps should be small enough to roll back the generator to the beginning of the key-load (end of the IV-load). However, by knowing the initial state of the generator and the IV, it is possible to step the generator forward to the same point at the beginning of the key-load. Therefore, only one path in the tree will correspond to the real state update chain, and this will reveal the portion of the key involved. Furthermore, it is also possible to use a **meet-in-the-middle attack** attack in which a certain portion of initial key bits is checked using brute force and the following bits are recovered using backward clocking in the key-load. At the ”meeting point”, the generator should acquire the same

internal state. This criterion is used for eliminating incorrect keys.

III. Application to MICKEY

The state space of MICKEY includes branch points and orphan states, which is a consequence of the self-mutual control used in the design of the cipher. Assuming a randomly chosen state, it is fairly straightforward to express the probabilities of this state being 1, but it gives no information about the distribution of the states in the whole graph. Therefore, the following four sections show how to find the fraction of the backward states tree based on the probability of branches appearing.

A. Brief Description of MICKEY

There are two versions of the cipher MICKEY-80 v2 and MICKEY-128 v2 [**?**], [5]. Each of them takes two input parameters: initialization vector (IV) and session key (K). The general architecture of the design is shown in Figure 2.

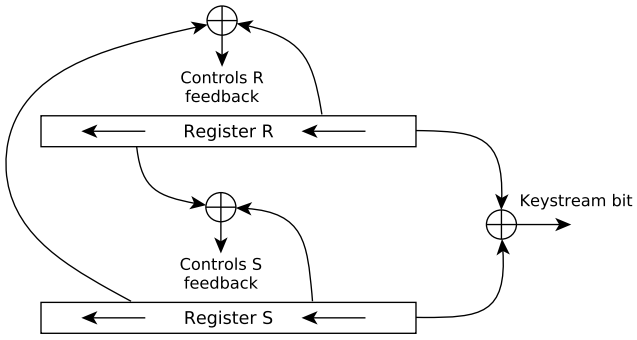

Fig. 2. Clocking Architecture of the MICKEY Cipher

Both versions are based on a combination of linear (R) and nonlinear (S) registers with length RL, which is shown in Table I. Cells of registers have bit values denoted by $r_0, r_1, \ldots, r_{RL-1}$ and $s_0, s_1, \ldots, s_{RL-1}$. Both registers, R and S, are clocked in two different ways depending on the control bit $CB\_R = CB\_SL \oplus CB\_RR$ for R and $CB\_S = CB\_SR \oplus CB\_RL$ for S respectively, where CB_XY means a certain bit of the X register from Table I (for example, $CB\_SL = s_{34}$). The update function of registers has an additional input parameter the so-called input bit (IB_S and IB_R). The difference in parameters of MICKEY-80 v2 and MICKEY-128 v2 is described in Table I. As can be seen from the table, the ratio of the register length to the other parameters is almost the same for both versions of MICKEY. Nonetheless, the state spaces for MICKEY-80 v2 and MICKEY-128 v2 differ significantly, as will be shown later.

According to the terminology accepted in section II, vectorial Boolean functions for MICKEY are defined by $f_R = CB\_SR \oplus CB\_RL$, $f_S = CB\_SL \oplus CB\_RR$ and update functions by $A^I_{C_R} = CLOCK\_R$, $N^I_{C_S} = CLOCK\_S$, where $CLOCK\_R$, $CLOCK\_S$ are registers' R and S update functions for the cipher MICKEY respectively. The length of registers is identical and equals $n_s = n_r = RL$.

The cipher runs in the following way:

- initialise the registers R and S with all zeros;
- IV-load (this mode corresponds to $CLOCK\_K\_IV$ function);
- key-load ($CLOCK\_K\_IV$);
- preclock ($CLOCK\_PRECLOCK$);
- key-stream generation ($CLOCK\_KG$).

All modes except key-stream generation work in so-called mix mode. This means that the input bit of the register R depends on a certain bit of the register S, and is denoted by CB_SM. The update clocking function of the key generator ($CLOCK\_KG(R, S, MIXING, INPUT\_BIT)$, where *INPUT_BIT* is bit of key or IV) is shown in Algorithm 1.

Algorithm 1 CLOCK_KG

Input: registers R and S, MIXING and INPUT_BIT
Output: updated states of registers R and S
CB_R = CB_SL $\oplus$ CB_RR
CB_S = CB_SR $\oplus$ CB_RL
if MIXING = TRUE **then**
  IB_R = $INPUT\_BIT \oplus$ CB_SM;
else
  IB_R = $INPUT\_BIT$
end if
IB_S = $INPUT\_BIT$
CLOCK_R(R, IB_R, CB_R)
CLOCK_S(S, IB_S, CB_S)

As already noted in section II, it is assumed that the adversary knows the state of the registers R and S. Our evaluation of the MICKEY cipher resistance is based on the construction of a backward states tree, as described in detail for A5/1 in [7]. A brief description of the tree construction with respect to the MICKEY cipher is presented below.

The previous states are computed using the functions $CLOCK\_PRECLOCK^{-1}$, $CLOCK\_K\_IV^{-1}$ and $CLOCK\_KG^{-1}$, which are inverse to the clock functions of MICKEY. The algorithm for achieving reverse states results in an exhaustive search for all possible values of input parameters (input, control and feedback bits for both registers) and the elimination of states with impossible conditions. Algorithms of function $CLOCK\_X^{-1}(R, S)$ for MICKEY-80 v2 are given in Appendix A.

It is worth noting that the previous state is not always uniquely determined. The number of branches may vary, depending on the state and mode of the cipher. Three different backward states trees can be acquired for different modes. Any of these trees can be considered as a graph of state transitions of finite-state machines. The general structure of the tree is shown in Figure 3. Hereinafter we will refer to such concepts connected to a tree as: the level is the set of all backward states that may result in an original state after a certain number of

TABLE I
PARAMETERS OF THE CIPHERS MICKEY-80 V2 AND MICKEY-128 V2

Version	RL	Key length	Preclock length	CB_SL	CB_SR	CB_SM	CB_RL	CB_RR
80	100	80	100	34	67	50	33	67
128	160	128	160	54	106	80	53	106

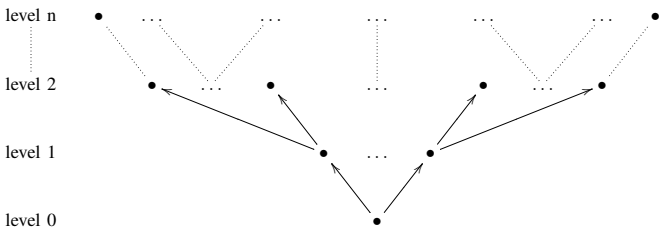

Fig. 3. A Backward States Tree

clocks; the degree of branch is the number of possible previous states, which give R and S after one clock forward. In K/IV load mode, edges are labeled with the appropriate bit of the sought-for key.

B. Key-Stream Generation Mode

Let $\underline{\rho}_0$ and $\underline{\rho}_1$ denote the two predecessors of the R-register by applying the inverses of the two different transition matrices [10]. Similarly, let $\underline{\sigma}_0$ and $\underline{\sigma}_1$ denote the two predecessors of the S-register by applying the inverses of the two different nonlinear register update function. The register self-control is obtained from two taps of both registers. These taps' indices are denoted by c and a, and the corresponding predecessor state bits are denoted by $\rho_{0,c}$, $\rho_{1,c}$, $\rho_{0,a}$, $\rho_{1,a}$, and $\sigma_{0,c}$, $\sigma_{1,c}$, $\sigma_{0,a}$, $\sigma_{1,a}$. The register control signals are denoted by $Rctrl_{ij}$ and $Sctrl_{ij}$ where ij is an index indicating the state pair $(\underline{\rho}_i, \underline{\sigma}_j)$ from which they are derived. From the definition of the MICKEY ciphers the following relations between the register control signals and the predecessor state bits are evident.

$$\begin{aligned}
(Rctrl_{00}, Sctrl_{00}) &= (\rho_{0,c} + \sigma_{0,a}, \rho_{0,a} + \sigma_{0,c}) \\
(Rctrl_{10}, Sctrl_{10}) &= (\rho_{1,c} + \sigma_{0,a}, \rho_{1,a} + \sigma_{0,c}) \\
(Rctrl_{01}, Sctrl_{01}) &= (\rho_{0,c} + \sigma_{1,a}, \rho_{0,a} + \sigma_{1,c}) \\
(Rctrl_{11}, Sctrl_{11}) &= (\rho_{1,c} + \sigma_{1,a}, \rho_{1,a} + \sigma_{1,c})
\end{aligned}$$

From the above equations it is easy to see that the 2^8 values for the 8 predecessor state bits result in 64 distinct values of the 8 $Rctrl$ and $Sctrl$ control signals. Out of these 64 combinations, 21 give rise to orphan states (BP0), 24 to regular states (BP1), 18 to states with 2 predecessors (BP2), and 1 to states with four predecessors (BP4). None of the 64 values gives rise to states with three predecessors (BP3).

Branch points with four predecessors have the following control signals: $Rctrl_{00} = 0$, $Sctrl_{00} = 0$, $Rctrl_{01} = 0$, $Sctrl_{01} = 1$, $Rctrl_{10} = 1$, $Sctrl_{10} = 0$, $Rctrl_{11} = 1$, and $Sctrl_{11} = 1$. These conditions result in the following six conditions on the values of the corresponding predecessor state bits.

$$\begin{aligned}
(\rho_{0,c} = \sigma_{0,a}) \quad &\wedge \quad (\rho_{0,a} = \sigma_{0,c}) \\
(\rho_{1,c} = 1 + \sigma_{0,a}) \quad &\wedge \quad (\rho_{1,a} = \sigma_{0,c}) \\
(\sigma_{0,a} = \sigma_{1,a}) \quad &\wedge \quad (\sigma_{0,c} = 1 + \sigma_{1,c})
\end{aligned}$$

If we assume a random R-register fill, then, due to the linearity of the R-register, the probabilities of all 16 combinations of the 4 ρ bits are equal to $\frac{1}{16}$. As branch points in the S-register can only occur if the most significant state bit has value 1, we arrive at the following expression for the probability of a branch point state with four predecessors.

$$\begin{aligned}
Pr(BP4) \quad = \quad & \frac{1}{2} \cdot \frac{1}{16} Pr((\sigma_{0,a} = \sigma_{1,a}) \wedge \\
& (\sigma_{0,c} = 1 + \sigma_{1,c})) \qquad (1)
\end{aligned}$$

Assuming a random S-register fill, the values are $Pr(BP4) = 0.00819, 0.00835, 0.00856, 0.00885$, for MICKEY-80 v1, -80 v2, -128 v1, -128 v2, respectively. These values have been calculated using a method described in Section 3 of [10].

In the same way the probability of other state types can be calculated. For orphan states (BP0) we find

$$\begin{aligned}
Pr(BP0) = \tfrac{1}{2} \cdot \tfrac{1}{16} \cdot \\
(4Pr((\sigma_{0,a} = \sigma_{1,a}) \wedge (\sigma_{0,c} = \sigma_{1,c})) + \\
9Pr((\sigma_{0,a} = \sigma_{1,a}) \wedge (\sigma_{0,c} = 1 + \sigma_{1,c})) + \\
4Pr((\sigma_{0,a} = 1 + \sigma_{1,a}) \wedge (\sigma_{0,c} = \sigma_{1,c})) + \\
4Pr((\sigma_{0,a} = 1 + \sigma_{1,a}) \wedge (\sigma_{0,c} = 1 + \sigma_{1,c}))). \qquad (2)
\end{aligned}$$

From expressions (1) and (2) one obtains

$$Pr(BP0) = \frac{1}{8} + 5Pr(BP4). \qquad (3)$$

Carrying on in the same way, the probabilities of BP1 and BP2 are obtained

$$Pr(BP1) = \frac{1}{4} - 8Pr(BP4) \qquad (4)$$

$$Pr(BP2) = \frac{1}{8} + 2Pr(BP4). \qquad (5)$$

The sum of all probabilities is equal to one half, because we have considered only the S-register states with msb equal to 1. S-register states with msb equal to 0, give rise to singularities in the R-register only. Again due to the linearity of the R-register, the probabilities of the three kinds of R-register states (BP0, BP1, and BP2) are equal to $\frac{1}{8}$, $\frac{1}{4}$, and $\frac{1}{8}$ respectively.

The overall probabilities are given below.

$$Pr(BP0) = \frac{1}{4} + 5Pr(BP4) \quad (6)$$

$$Pr(BP1) = \frac{1}{2} - 8Pr(BP4) \quad (7)$$

$$Pr(BP2) = \frac{1}{4} + 2Pr(BP4) \quad (8)$$

From the above expression for the state probabilities, it is seen that the average number of predecessor states equals 1, regardless of $Pr(BP4)$. The variance is given by $Var = \frac{1}{2} + 16Pr(BP4)$, which is less than one (≈ 0.63) for all MICKEY versions.

C. Preclock Mode

In preclock mode the MICKEY cipher has an additional modifier signal from the tap with index $N/2$ of the nonlinear S-register to the input of the R-register. In line with Section III-B, we write $\sigma_{0,N/2}$ and $\sigma_{0,N/2}$ for the bits of $\underline{\sigma}_0$ and $\underline{\sigma}_1$ with index $N/2$. Consequently, this S-register tap modifies the R-register by xor-ing the tap value to the feedback bit of the R-register. Therefore, if there is a 1 value at the appropriate locations c or a of the feedback vector of the R-register, this will modify the control tap values of the R-register, thereby possibly changing the branch point conditions. It turns out that, depending only on the value of the feedback vector bits of the R-register, the cycle structure of preclock mode is either equivalent to that of the key-stream generation mode, or its cycle structure changes to include branch points with three predecessors having different probabilities of occurrence from the key-stream generation mode. The latter case is dealt with in this section. It should be remarked that MICKEY-80 v2 is the only version having preclock mode cycle structures equivalent to the key-stream generation mode, all other versions have exhibiting branch points with three predecessors.

Omitting the details of the calculations, the following results apply

$$Pr(BP4) = \tfrac{1}{2} \cdot \tfrac{1}{16} Pr((\sigma_{0,a} = \sigma_{1,a}) \wedge (\sigma_{0,c} = 1 + \sigma_{1,c}) \wedge (\sigma_{0,N/2} = \sigma_{1,N/2})) \quad (9)$$

$$Pr(BP3) = \tfrac{1}{2} \cdot \tfrac{1}{16} Pr((\sigma_{0,a} = \sigma_{1,a}) \wedge (\sigma_{0,N/2} = 1 + \sigma_{1,N/2})). \quad (10)$$

Again assuming random register fills, the values are $Pr(BP4) \approx 0.0027$ and $Pr(BP3) \approx 0.019$. Note from (9) that $Pr(BP4)$ in preclock mode is less than in the key-stream generation mode. As with the key-stream generation mode, the probabilities of the other state types can be expressed in $Pr(BP4)$ and $Pr(BP3)$, resulting in the expressions below.

$$Pr(BP0) = \tfrac{1}{4} + Pr(BP3) + 5Pr(BP4) \quad (11)$$

$$Pr(BP1) = \tfrac{1}{2} - Pr(BP3) - 8Pr(BP4) \quad (12)$$

$$Pr(BP2) = \tfrac{1}{4} - Pr(BP3) + 2Pr(BP4) \quad (13)$$

From the above expressions for the state probabilities, it is again seen that the average number of predecessor states equals 1, regardless of $Pr(BP4)$ and $Pr(BP3)$. The variance is given by $Var = \frac{1}{2} + 4Pr(BP3) + 16Pr(BP4)$, which is less than one for all MICKEY versions.

D. Key/IV Load Mode

Loading key and IV bits into the MICKEY ciphers is realized in mix mode, by shifting in key and IV bits in both registers in parallel. The structure of the state space in this mode can be adapted to include the uncertainty about the values of the key bits, when attempting to step backwards in the state space. This adaptation consists of taking into account branch points arising from one or more states leading to one next state when using two different values for the key bit input. In general, the number of possible predecessors doubles in this model, ranging from 0 (orphan states) through 8 (branch points with four predecessors for both values of the key bit). We will refer to this model as the compound state space. The compound state space can be viewed as the union of the state spaces for all combinations of key bit values. For the MICKEY ciphers there are two different compound state spaces, in a similar way as explained for preclock mode. In this respect, MICKEY-80 v2 is different from the other published versions. For example, MICKEY-80 v2 has no branch points with five and seven predecessors.

Let us denote the probability that a randomly chosen state has i predecessors for a key bit of value k and j predecessors for a key bit of value $k \oplus 1$ by P_{ij}. Then from symmetry arguments, $P_{ij} = P_{ji}$. For MICKEY-80 v2, $P_{04} = 0$, $P_{14} = 0$, and $P_{12} = P_{01}$. the probabilities of all nonzero state types can be expressed in four of them, i.e. P_{02}, P_{22}, P_{24}, and P_{44}, obtaining the expressions below for states with nonzero probabilities.

$$P_{00} = P_{22} + 4P_{24} + 3P_{44} \quad (14)$$

$$P_{01} = \frac{1}{4} - P_{02} - P_{22} + P_{24} + 2P_{44} \quad (15)$$

$$P_{11} = 2P_{02} + 2P_{22} - 10P_{24} - 12P_{44} \quad (16)$$

$$(17)$$

The probabilities of the various state types in both models have been determined by calculations and verified by experiments. The results are given in Table II, showing the differences in the two compound state spaces. The average number of predecessors is two in both cases, one for each key bit value. The variances are significantly different: $Var \approx 2.88$ for MICKEY-80 v2, and $Var \approx 0.89$ for the other MICKEY ciphers., indicating that stepping backwards is harder for MICKEY-80 v2 than for the other MICKEY ciphers.

Of interest is the probability of a state being an orphan for one key bit value and a state with one or more predecessors for the complement of the key bit value. In this case, the uncertainty in the key bit value is resolved, and therefore, the total uncertainty in the key is reduced by one bit. In the case of MICKEY-80 v2, however, this probability is very small ($\approx 3 \cdot 10^{-4}$).

E. Computational Results

The tree construction method, described in section III-A, allows to evaluate following properties of a finite state machine.

TABLE II
STATE TYPE PROBABILITIES FOR THE COMPOUND STATE SPACE MODEL OF THE MICKEY-80 V2 IN KEY/IV LOAD MODE. STATE TYPE n-m MEANS A STATE WITH n PREDECESSOR STATES FOR A KEY BIT WITH VALUE 0, AND WITH m PREDECESSOR STATES FOR A KEY BIT WITH VALUE 1.

State type	80 v2
0	0.32
1	$1.1 \cdot 10^{-4}$
2	0.39
3	$1.1 \cdot 10^{-4}$
4	0.28
6	$1.1 \cdot 10^{-5}$
8	0.014

1) Degree Probability.: The backward tree was constructed by using functions $CLOCK\_X^{-1}$ (Appendix A). Probabilities of degrees were calculated with accuracy limited by 18-level tree for random and real values of registers R and S. Tables III and IV show the difference in the degree probability distribution for different initial values and modes.

TABLE III
THE DEGREE PROBABILITY FOR RANDOM INITIAL STATES.

Degree	Key/IV load		Preclock		KG	
	80 v2	128 v2	80 v2	128 v2	80 v2	128 v2
0	0.2982	0.198	0.2802	0.2825	0.3014	0.2718
1	0.00009	0.1031	0.4377	0.459	0.4052	0.4281
2	0.4229	0.4022	0.2735	0.2294	0.2844	0.29
3	0.0001	0.1087	-	0.0256	-	-
4	0.2698	0.1703	0.0085	0.0035	0.0090	0.0101
6	0.00001	0.0177	-	-	-	-
8	0.0089	-	-	-	-	-

TABLE IV
THE DEGREE PROBABILITY FOR REAL INITIAL STATES.

Degree	Key/IV load		Preclock mode		KG	
	80 v2	128 v2	80 v2	128 v2	80 v2	128 v2
0	0.2773	0.2186	0.3052	0.29	0.3041	0.3038
1	0.00001	0.1047	0.4345	0.4534	0.4323	0.4154
2	0.4331	0.3753	0.2523	0.2256	0.2558	0.2698
3	0.00002	0.1029	-	0.0289	-	-
4	0.28	0.1783	0.008	0.0021	0.0079	0.0111
6	0.00007	0.0203	-	-	-	-
8	0.0095	-	-	-	-	-

The results for real and random points approximately coincide with the theoretical ones from Table II. Thus, the degree determination method described in subsections III-B-III-D could be applied to a stream cipher with the structure given in section II at the designing stage.

In key/IV load mode the expectation value of branch points degree for all versions approximately equals 2. Appropriate value for preclock and KG mode is approximately equal to 1. Thus, no matter in what mode the values of the registers were obtained. It is always possible to perform reverse steps and acquire the state after key initialization function.

Table V shows the average number of possible states at each level. For reducing the dependency on the initial state 1000 transformations with random values was performed. The results for the other modes are given in Appendix B. Clearly, the number of states increases in accordance with the expectation value.

TABLE V
THE NUMBER OF BACKWARD STATES DEPENDING ON THE LEVEL OF A TREE IN THE KEY LOAD MODE.

Level	Number of states	
	80 v2	128 v2
0	1	1
1	3	3
2	9	7
3	25	18
4	45	39
5	143	82
6	247	171
7	523	347
8	1183	703
9	2221	1435
10	5075	2904
11	9453	5849
12	18694	11834
13	37702	23801
14	70675	47759
15	136867	95716

2) Determination of Key Bits Based on a Backward States Tree.: Each reverse step increases the probability of subtree cutting off with all previous states. This property exists since there is a high probability of orphan states (see Table IV). Therefore a key bit could be found uniquely. An example of that situation is shown in Table VI.

TABLE VI
THE PROBABILITY DISTRIBUTION OF KEY BITS ON A 5-LEVEL BACKWARD TREE.

Level	Probability			
	80 v2		128 v2	
	1	0	1	0
1	0.5	0.5	1	0
2	0.5	0.5	0.5	0.5
3	0.5	0.5	0	1
4	0.5	0.5	0.5	0.5
5	0.4857	0.5143	0.5	0.5

For MICKEY-128 v2 the key bit can be uniquely determined at levels 1 and 3 ("1" and "0" bits respectively). However, knowing this information does not allow attacker to definitively find the backward state. Therefore, the knowledge of key bit value does not reduce the tree of states and the complexity of determining all bits of the key. Moreover, an opportunity of finding the key bit directly depends on an initial state. For instance, for MICKEY-80 v2 it is impossible to determine the key bits with probability 1.

3) Identical Key-Streams for Different Key/IV pairs.: Functions used for different modes of the MICKEY cipher allow to generate key-streams shifted by some bits for different pairs of key and IV.

Let z_i^h be i^{th} bit of a key-stream for h^{th} pair of (K_h, IV_h). Suppose also that $K_1 = \{k_0, k_1, \ldots, k_{n-1}\}$, where n the length of the key (Table I). Then it is possible to find such (K_1, IV_1) and (K_2, IV_2) for which the states of registers will differ by one clock and the key-streams have the property

TABLE VII
DIFFERENCES BETWEEN PARAMETERS FOR VARIOUS SETS OF KEY AND IV

IV_1				K_1						Preclock					Key-Stream Generation				
iv_0	iv_1	…	iv_j	k_0	k_1	k_2	…	k_{n-2}	k_{n-1}	0	0	0	…	0	0	0	0	0	…
iv_0	iv_1	…	iv_j	k_0	k_1	k_2	…	k_{n-2}	k_{n-1}	0	0	0	…	0	0	0	0	0	…
IV_2				K_2						Preclock					Key-Stream Generation				

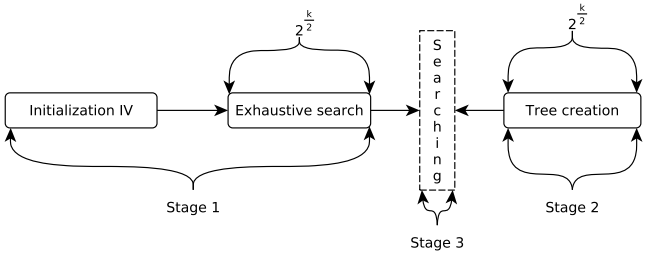

Fig. 4. Meet-in-the-middle attack on the cipher MICKEY

$z_i^2 = z_{i+1}^1$.

Assume that $IV_1 = \{iv_0, iv_1, \ldots, iv_j\}$, $IV_2 = \{iv_0, iv_1, \ldots, iv_j, k_0\}$ and

$$K_2 = K_1 << 1 = \{k_1, k_2, \ldots, k_{n-1}, 0\}. \qquad (18)$$

The relative placement of bits for various sets of K/IV is shown in Table VII.

Obviously, the state of registers will differ only by one step. Therefore, the key-stream will have the same properties, i.e. $z_i^2 = z_{i+1}^1$. Since preclock mode is equivalent to key/IV mode, when input bit is 0, then the MICKEY cipher has shifting feature. Only one condition is a necessity: the s_{50}^1 bit must equal 0 at the moment of changing between preclock and key-stream generation modes.

The quantity of IV_2 bits can be increased by different value. As a result, K_2 and a key-stream will be shifted to the same amount of bits.

Similar arguments can be applied to 128-bit version of MICKEY. Examples of (K_1, IV_1) and (K_2, IV_2) with the described property for both versions of the cipher are given in Appendix C.

4) Meet-in-The-Middle Attack on Cipher MICKEY.: The extension of the scenario 2 described in section II allows to apply "meet-in-the-middle" attack, which reduces the complexity of finding the key in significant way. Suppose the state of the registers after k-clocks from IV initialization be known. Clearly, k-clocks correspond to k bits of the unknown key. The general scheme with 3-stages of the attack is shown in Fig. 4.

At the first stage calculate input state for initialization key function using the IV value. For obtained state find all possible states for $\frac{k}{2}$ key bits, applying $CLOCK\_K\_IV$ function. At the second stage build the backward states tree up to $\frac{k}{2}$ level using $CLOCK\_K\_IV^{-1}$. Finally, find coincidences in states acquired at the previous stages.

The complexity of exhaustive search attack or building of backward states tree is approximately the same and equals 2^k. Combination of these two methods in meet-in-the-middle attack makes it more practical and leads to the complexity:

$$O_d(k) + O_i(k) + O_f(k) = 2^{\frac{k}{2}} + 2^{\frac{k}{2}} + 2^{\frac{k}{2}} \approx 2^{\frac{k}{2}+2}$$

where $O_d(k)$ is the complexity of exhaustive search for $\frac{k}{2}$ key bits, $O_i(k)$ is the complexity of the backward states tree construction and $O_f(k)$ is the search complexity using hash tables [11].

IV. CONCLUSIONS

In this article the analysis of self-controlled key-stream generators based on the MICKEY cipher was made. Stepping backwards in the state space of the cipher is possible and feasible in all modes including key/IV load mode. In mix mode the overall structure of the state space depends only on one or two bits in the feedback vector of the linear R-register.

From our analysis we conclude that the 128 bit versions are equally strong as the 80 bit version 2 cipher, and that the first 80 bit version is substantially weaker with an effective key length of 50 bits. This difference could have easily been avoided by a minor change in the feedback vector of the R-register. It is unknown to the authors if the MICKEY designers were aware of this fact when designing the MICKEY ciphers.

Proposed method allows to estimate degrees' probability at the design stage of MICKEY-like ciphers, and this was proved by practical results. Thus, it is possible to justify the choice of the encryption algorithm parameters.

REFERENCES

[1] R.A. Ruppel, "Analysis of Design of Stream Ciphers," *Communications and Control Engineering Series*, New York: Springer-Verlag NY, USA. 1986.

[2] J. Hong and Woo-Hwan Kim, "TMD-Tradeoff and State Entropy Loss Considerations of Stream Cipher MICKEY," *INDOCRYPT 2005*, pp. 169–182.

[3] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers," *In Proceedings of ASIACRYPT 2000*, volume 1976 of Lecture Notes in Computer Science, pages 1–13. Springer-Verlag, 2000.

[4] A. Biryukov, A. Shamir and D. Wagner, "Real time cryptanalysis of A5/1 on a PC," *In Proceedings of PKC 2001*, volume 1978 of Lecture Notes in Computer Science, pages 37–44. Springer-Verlag, 2001.

[5] S. Babbage and M. Dodd, "The Stream Cipher MICKEY 2.0," *eSTREAM, ECRYPT Stream Cipher Project, End of 3rd Phase*, September 2009. http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf.

[6] Matthew J. B. Robshaw and Olivier Billet, "New Stream Cipher Designs – The eSTREAM Finalists," *volume 4986 of Lecture Notes in Computer Science*. Springer-Verlag, 2008.

[7] Jovan Dj. Golic, "Cryptanalysis of alleged A5 stream cipher," *Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques*, May 11-15, 1997, Konstanz, Germany.
[8] Karsten Nohl and Chris Paget, "A5/1 Security Project, Creating A5/1 Rainbow Tables," 2009. Available online at http://opensource.srlabs.de/.
[9] Alex Biryukov, Adi Shamir and David Wagner, "Real Time Cryptanalysis of A5/1 on a PC," *FSE 2000.* pp. 1–18.
[10] Cees J.A. Jansen, "The State Space Structure of the MICKEY Stream Cipher," *In Horlin, F, ed.: 32nd Symposium on Information Theory in the Benelux, Brussels*, Werkgemeenschap voor Informatien Communicatietheorie, 2011.
[11] Thomas H. Cormen et. al, "Introduction to Algorithms," 2^{nd} *ed., The MIT Press*, Cambridge, Mass., May 2001, Chap. 11.

APPENDIX A
DESCRIPTION OF INVERSE FUNCTIONS

A. Algorithm of CLOCK_S^{-1}

The algorithm for finding previous states of S register of cipher MICKEY-80 v2.

Algorithm 2 CLOCK_S^{-1}

Input: bits IB_S, CB_S, FB_S and register S
Output: TRUE if branch exist, otherwise FALSE
S$^{'}$ $\leftarrow$ S
if FB_S = TRUE **then**
  if CB_S = TRUE **then**
    for i=0 to 99 **do**
      $\hat{s_i} \leftarrow s_i^{'} \oplus$ FB1$_i$;
    end for
  else
    for i=0 to 99 **do**
      $\hat{s_i} \leftarrow s_i^{'} \oplus$ FB0$_i$;
    end for
  end if
else
  for i=0 to 99 **do**
    $\hat{s_i} \leftarrow s_i^{'}$;
  end for
end if
if $\hat{s_0} \neq 0$ **then**
  return FALSE
end if
$s_{99} \leftarrow$ FB_S $\oplus$ IB_S;
$s_{98} \leftarrow \hat{s}_{99}$;
for i=98 to 1 **do**
  $s_{i-1} \leftarrow \hat{s_i} \oplus ((s_i \oplus COMP0_i) \wedge (s_{i+1} \oplus COMP1_i))$;
end for
return TRUE

B. Algorithm of CLOCK_R^{-1}

The algorithm for finding previous states of R register of cipher MICKEY-80 v2.

Algorithm 3 CLOCK_R^{-1}

Input: bits IB_R, CB_R, FB_R and register R
Output: TRUE if brach exist, otherwise FALSE
R$^{'}$ $\leftarrow$ R
if CB_R = TRUE **then**
  $r_0 \leftarrow r_0 \oplus$ (FB_R $\wedge$ RTAPS$_0$);
  for i=1 to 99 **do**
    $r_i \leftarrow r_i \oplus r_{i-1} \oplus$ (FB_R $\wedge$ RTAPS$_i$);
  end for
  if $r_{99} \neq$ (FB_R $\oplus$ IB_R) **then**
    return FALSE
  end if
else
  if ($r_0 \oplus$ (FB_R $\wedge$ RTAPS$_0$)) $\neq 0$ **then**
    return FALSE
  end if
  for i=1 to 99 **do**
    $r_{i-1} \leftarrow r_i \oplus$ (FB_R $\wedge$ $RTAPS_i$);
  end for
  $r_{99} \leftarrow$ FB_R $\oplus$ IB_R;
end if
return TRUE

C. Algorithm of CLOCK_K_IV^{-1}

The algorithm for finding previous states in the K/IV mode of cipher MICKEY-80 v2.

Algorithm 4 CLOCK_K_IV^{-1}

Input: registers R and S
Output: all possible branches
branches $\leftarrow \emptyset$
for $s = 0$ to 64 **do**
  FB_S $\leftarrow (s \gg 0) \wedge 1$;
  IB_S $\leftarrow (s \gg 1) \wedge 1$;
  CB_S $\leftarrow (s \gg 2) \wedge 1$;
  FB_R $\leftarrow (s \gg 3) \wedge 1$;
  IB_R $\leftarrow (s \gg 4) \wedge 1$;
  CB_R $\leftarrow (s \gg 5) \wedge 1$;
  TS $\leftarrow$ S;
  TR $\leftarrow$ R;
  if CLOCK_S^{-1}(TS,IB_S,CB_S,FB_S) $\neq$ TRUE **then**
    continue;
  end if
  if CLOCK_R^{-1}(TR,IB_R,CB_R,FB_R) $\neq$ TRUE **then**
    continue;
  end if
  INPUT_BIT_S $\leftarrow$ IB_S;
  INPUT_BIT_R $\leftarrow$ IB_S $\oplus s_{50}$;
  CONTROL_BIT_R $\leftarrow s_{34} \oplus r_{67}$;
  CONTROL_BIT_S $\leftarrow s_{67} \oplus s_{33}$;
  if CONTROL_BIT_R = CB_R **and** CONTROL_BIT_S = CB_S **and** INPUT_BIT_R = IB_R **then**
    branches $\leftarrow$ Append(branches,[TS;TR;INPUT_BIT_S]);
  end if
end for
return branches

D. Algorithm of CLOCK_PRECLOCK^{-1}

The algorithm for finding previous states in the preclock mode of cipher MICKEY-80 v2.

Algorithm 5 CLOCK_PRECLOCK^{-1}

Input: registers R and S
Output: all possible branches
branches $\leftarrow$ Ø
for $s = 0$ to 32 **do**
  FB_S $\leftarrow (s \gg 0) \wedge 1$;
  CB_S $\leftarrow (s \gg 1) \wedge 1$;
  FB_R $\leftarrow (s \gg 2) \wedge 1$;
  IB_R $\leftarrow (s \gg 3) \wedge 1$;
  CB_R $\leftarrow (s \gg 4) \wedge 1$;
  TS $\leftarrow$ S;
  TR $\leftarrow$ R;
  if CLOCK_S^{-1}(TS,0,CB_S,FB_S) $\neq$ TRUE **then**
    continue;
  end if
  if CLOCK_R^{-1}(TR,IB_R,CB_R,FB_R) $\neq$ TRUE **then**
    continue;
  end if
  INPUT_BIT_R $\leftarrow s_{50}$;
  CONTROL_BIT_R $\leftarrow s_{34} \oplus r_{67}$;
  CONTROL_BIT_S $\leftarrow s_{67} \oplus s_{33}$;
  if CONTROL_BIT_R = CB_R **and** CONTROL_BIT_S = CB_S **and** INPUT_BIT_R = IB_R **then**
    branches $\leftarrow$ Append(branches,[TS;TR;0]);
  end if
end for
return branches

E. Algorithm of CLOCK_KG^{-1}

The algorithm for finding previous states in the key-stream generating mode of cipher MICKEY-80 v2.

Algorithm 6 CLOCK_KG^{-1}

Input: registers R and S
Output: all possible branches
branches $\leftarrow$ Ø
for $s = 0$ to 16 **do**
  FB_S $\leftarrow (s \gg 0) \wedge 1$;
  CB_S $\leftarrow (s \gg 1) \wedge 1$;
  FB_R $\leftarrow (s \gg 2) \wedge 1$;
  CB_R $\leftarrow (s \gg 3) \wedge 1$;
  TS $\leftarrow$ S;
  TR $\leftarrow$ R;
  if CLOCK_S^{-1}(TS,0,CB_S,FB_S) $\neq$ TRUE **then**
    continue;
  end if
  if CLOCK_R^{-1}(TR,0,CB_R,FB_R) $\neq$ TRUE **then**
    continue;
  end if
  CONTROL_BIT_R $\leftarrow s_{34} \oplus r_{67}$;
  CONTROL_BIT_S $\leftarrow s_{67} \oplus s_{33}$;
  if CONTROL_BIT_R = CB_R **and** CONTROL_BIT_S = CB_S **then**
    branches $\leftarrow$ Append(branches,[TS;TR;0]);
  end if
end for
return branches

APPENDIX B
THE NUMBER OF STATES ON EACH LEVEL OF THE BACKWARD TREE

The following tables show the dynamics of increasing the number of states depending on the quantity of steps taken backward.

TABLE VIII
THE NUMBER OF BACKWARD STATES DEPENDING ON THE LEVEL OF A TREE IN THE PRECLOCK MODE.

Level	Number of states	
	80 v2	128 v2
0	1	1
1	1	1
2	1	2
3	2	3
4	3	4
5	4	6
6	5	8
7	7	8
8	5	11
9	5	14
10	10	15
11	12	15
12	13	15
...	...	...
99	25	30
100	32	27
...	...	...
159	-	59
160	-	63

TABLE IX
THE NUMBER OF BACKWARD STATES DEPENDING ON THE LEVEL OF TREE IN THE KEY-STREAM GENERATION MODE.

Level	Number of states	
	80 v2	128 v2
0	1	1
1	1	2
2	1	3
3	2	4
4	3	6
5	3	6
6	4	4
7	5	3
8	3	4
...	...	...
125	17	64
126	17	81
127	16	91
128	20	97

APPENDIX C
EXAMPLE OF KEY-STREAMS WITH DIFFERENT LENGTH OF IV

In appendix examples of identical key-streams for different IV length are described. All values are presented as bytes in hexadecimal notation except the first and the last values of Z_i, which are bits.

A. MICKEY-80 v2

$$
\begin{aligned}
K_1 &= \{d3, ec, f0, 84, 8a, 1d, b1, b7, 4a, dd\} \\
IV_1 &= \{58, e5, 77, 0a, 9c, a2, 34, c7, cd, 5e\} \text{ (79 bits)} \\
K_2 &= \{a7, d9, e1, 09, 14, 3b, 63, 6e, 95, ba\} \\
IV_2 &= \{58, e5, 77, 0a, 9c, a2, 34, c7, cd, 5f\} \text{ (80 bits)} \\
Z_1 &= \{0, B7, 61, 27, 92, C5, 85, 91, 51, 18, 2A, D6, 7C, 8C, C8, \\
&\quad C7, 04\} \\
Z_2 &= \{B7, 61, 27, 92, C5, 85, 91, 51, 18, 2A, D6, 7C, 8C, C8, \\
&\quad C7, 04, 1\}
\end{aligned}
$$

B. MICKEY-128 v2

$$
\begin{aligned}
K_1 &= \{c9, 55, e7, 7a, 80, 13, 1a, ad, 40, 45, d9, 6c, 71, 04, 97, 9c\} \\
IV_1 &= \{4e, db, 6e, 01, 05, 98, 2b, 30, c3, 56, 5a, ed, 80, 85, \\
&\quad 18, aa\} \text{ (127 bits)} \\
K_2 &= \{92, ab, ce, f5, 00, 26, 35, 5a, 80, 8b, b2, d8, e2, 09, 2f, 38\} \\
IV_2 &= \{4e, db, 6e, 01, 05, 98, 2b, 30, c3, 56, 5a, ed, 80, 85, \\
&\quad 18, ab\} \text{ (128 bits)} \\
Z_1 &= \{0, 79, 23, 91, 05, E1, DD, 2D, 9D, 83, 3E, B4, 78, 52, \\
&\quad E5, A6, 66\} \\
Z_2 &= \{79, 23, 91, 05, E1, DD, 2D, 9D, 83, 3E, B4, 78, 52, \\
&\quad E5, A6, 66, 1\}
\end{aligned}
$$