



ELSEVIER

Available online at www.sciencedirect.com

 ScienceDirect

Information Sciences 177 (2007) 1715–1727

INFORMATION
SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Finding an internal state of RC4 stream cipher [☆]

Violeta Tomašević ^{a,1}, Slobodan Bojanić ^{b,*}, Octavio Nieto-Taladriz ^{b,2}

^a *Institute of Mihajlo Pupin, Volgina 15, 11060 Belgrade, Serbia and Montenegro*

^b *Universidad Politécnica de Madrid, Departamento de Ingeniería Electrónica, Ciudad Universitaria s/n, 28040 Madrid, Spain*

Received 4 November 2004; received in revised form 16 October 2006; accepted 26 October 2006

Abstract

The RC4 is a stream cipher widely deployed in software applications due to its simplicity and efficiency. The paper presents a cryptanalytic attack that employs the tree representation of this cipher and introduces an abstraction in the form of general conditions for managing the information about its internal state. In order to find the initial state, the tree of general conditions is searched applying the hill-climbing strategy. The complexity of this attack is lower than that of an exhaustive search. The attack is derived from a general cryptanalytic approach for a class of table-shuffling ciphers, whose next-state function permutes the table entries. Incorporating the general conditions in the existing backtracking algorithm, the estimated complexity of the cryptanalytic attack is decreased below the best published result but the RC4 still remains a quite secure cipher in practice.

© 2006 Elsevier Inc. All rights reserved.

Keywords: Algorithm; Complexity; Cryptanalysis; Stream cipher; RC4

1. Introduction

The RC4 is a stream cipher widely deployed in software applications due to its simplicity and efficiency. The RC4 Keystream generator (Ron's code #4, RSA Data Security Inc., 1987) is based on the table-shuffling principle and is designed for fast software implementation [14]. It is used in many commercial products, including Lotus Notes, Oracle Secure SQL, Microsoft Windows, as well as some standards such as Secure Sockets Layer standard SSL 3.0.

The RC4 is, in fact, a family of algorithms parameterized by a positive integer n , which denotes the width of a table entry and the width of output symbol in bits, as well (usually $n = 8$). The initial table is derived from a secret key. The table varies slowly in time steps by swapping two entries indexed by two pointers, while all

[☆] This work has been partially supported by the Ministries of Science and Technology of Serbia (# IT.1.24.0041) and Spain (# TIC2003-09061-C03-02 and the "Ramon y Cajal" program).

* Corresponding author. Tel.: +34 91 3367322; fax: +34 91 3367323.

E-mail addresses: violeta@impc.com (V. Tomašević), slobodan@die.upm.es (S. Bojanić), nieto@die.upm.es (O. Nieto-Taladriz).

URL: <http://www.imp.bg.ac.yu> (V. Tomašević).

¹ Tel.: +381 11 2774 024; fax: +381 11 2776 583.

² Tel.: +34 91 3367322; fax: +34 91 3367323.

other entries in the table remain the same. Then, the output function determines the position in the table whose value is used as the output symbol. A sequence of output symbols makes the stream, which is XORed with the plaintext to generate the final ciphertext. Since the output sequence depends on the initial table only, the knowledge of this table enables generating the output sequence without knowing the secret key.

The cryptanalysis research of the RC4 stream cipher has been mainly devoted to the statistical analysis of the RC4 output sequence [1,3,4,7,10,11], or to the RC4 initialization weaknesses [2,5,13] that can serve for the RC4 distinguishing. It is still an open question if these results can be used in the RC4 practical cryptanalysis. Although this stream cipher was published in 1994, concrete cryptanalytic algorithms have been very rare in the literature up to now. Actually nowadays, the most important algorithms for decipherment of RC4 are an algorithm formulated by Knudsen et al. [6] and the other, very similar one, by Mister and Tavares [9]. These algorithms exploit the combinatorial nature of RC4 to reduce the search space as an improvement over the exhaustive search. Although without a real threat for the security of the cipher, the mentioned algorithms could be used in completing the internal state, given some additional information [8].

In this paper, the efforts regarding RC4 stream cipher cryptanalysis are devoted to finding maximum amount of information about the current state available at a given time and to formulating a cryptanalytic attack based on this information. Therefore, we propose the tree representation [16] of the RC4 algorithm which contains a set of trees, each for one output symbol. The nodes and branches of these trees encompass all possible information at a given time. However, since the trees are progressively large, we could not exploit practically all available information in the attack. This problem is immanent in all other attacks. In our algorithm we propose an analytical abstraction named *the general conditions* of the tree information in order to consider a reasonable amount of information [15]. Each general condition practically represents all conditions from a subtree. The general conditions are organized into the tree structure. Our algorithm searches this tree applying the hill-climbing strategy to find the internal state. The information gained from general conditions can also be used in other attacks on RC4 cipher and increase their efficiency. In this paper we suggest the modification of the backtracking algorithm that is given in Ref. [6].

The rest of the paper is organized in the following way. The general approach to the cryptanalysis of the stream ciphers based on table permutation is described in Section 2 with corresponding subsections about tree representation of information, the search algorithm and complexity analysis. Section 3 describes how the general approach is applied to the RC4 cryptanalysis. Section 4 presents the modified backtracking algorithm where corresponding subsections show the modification of general conditions, the effects of modified conditions examination and the complexity analysis. Conclusions are given in Section 5.

2. General approach

The stream ciphers based on the table-shuffling principle use a relatively big table that slowly varies in time under the control of itself. In this paper we consider a class of table-shuffling based stream ciphers where the next-state function permutes the internal state table entries, and the output function determines a position of output symbol in the table. As can be seen, the previous internal state directly affects the next one. Although it does not seem so at first sight, the swapping mechanism represents a very nonlinear process, which is hard to analyse.

In order to explain our general approach for finding the correct initial table using a small segment of output sequence, we introduce first the trees of information, and, after that, we represent these trees by the tree of the so-called *general conditions*, which is searched during the initial table reconstruction process. Then, complexity analysis is given.

2.1. Tree representation

Let S denote the internal state table of 2^n different n -bit words, which varies in time steps by permuting s table entries ($s < 2^n$). All words except the permuted ones remain the same. At time t , let the entries of current state table, S_{t-1} , that will be permuted be denoted by $q_i^1, q_i^2, \dots, q_i^s$, and the permuted entries of the S_t table by $p_i^1, p_i^2, \dots, p_i^s$. Although they belong to the S_{t-1} table, positions q_i^j , $i \in [1, s]$ have an index t , because it shows the origin of elements on positions p_i^j , $j \in [1, s]$. That means, a word at q_i^j position in the S_{t-1} table is the same

as the word at p_i^j position in the table S_i . The q_i^j and p_i^j , $i \in [1, s]$ sets of entries are equal. Now, the content of the table S_i can be described as

$$S_i(k \neq p_i^j, i \in [1, s]) = S_{i-1}(k), \tag{1}$$

$$S_i(k = p_i^j) = S_{i-1}(q_i^j) \quad \text{for } i \in [1, s]. \tag{2}$$

The output n -bit symbol Z_t is given by

$$Z_t = S_t(L_t). \tag{3}$$

Applying Eqs. (1) and (2), it follows that Eq. (3) can be represented by the tree structure given in Fig. 1.

The tree consists of nodes distributed at $t + 1$ levels. Nodes at level i , $0 < i \leq t$ refer to the set of all possible positions in the S_{t-i} table where Z_t could be found. The nodes are connected by the branches, which represent the conditions to pass from one node to another. Each path in this tree represents an AND function of the conditions expressed by the branches on this path.

At time t , there is a set of t trees, each for one output symbol. These trees contain all possible information about the current state because they refer to all positions (and related conditions) in the S_i tables, $0 \leq i \leq t$, where the output symbols could be found.

Each path from root to some leaf can be represented by an abstraction in the form of general condition derived by applying the AND logic function to all conditions given in the branches on this path. In that way, the tree corresponding to the Z_i output symbol can be described by $(s + 1)^i$ general conditions.

The probabilities that general conditions lead to the solution are generally unequal and can be determined by multiplying the known probabilities on the path to the given node.

2.2. Search algorithm

To formulate our attack, we organize the general conditions into the tree structure shown in Fig. 2. At time t , this tree has $t + 1$ levels. The level zero serves as a tree root only, and its node is not examined. Nodes at tree level i , $0 < i \leq t$ correspond to the $(s + 1)^i$ general conditions at time i . They represent the general conditions in a decreasing order of their probabilities, and are denoted as C_k^i , $1 \leq k \leq (s + 1)^i$. By observing this order, a reduction of search space is achieved.

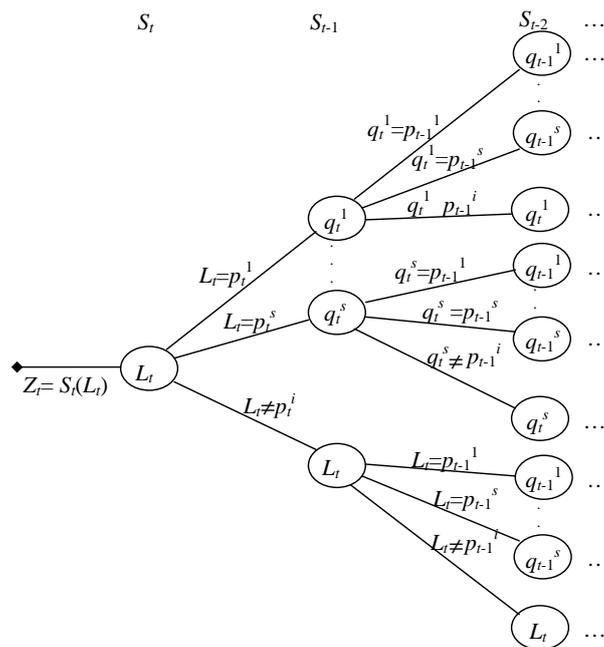


Fig. 1. Tree of information for the output symbol Z_t .

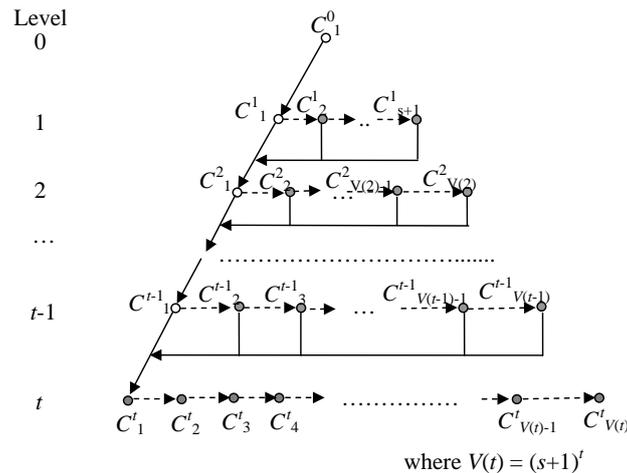


Fig. 2. Tree of general conditions.

In order to find the initial table, our algorithm examines the general conditions from the tree applying the basic principle of the hill-climbing search strategy [12]. This strategy uses the heuristic estimation function to describe how much the final state is far from the current state. In our case, it corresponds to the probability that some general condition leads to the solution. The current state of our attack contains the information collected up to the given time. The general condition has to be examined taking into account this current state. If general condition contains the information that is contrary to the current state of the system then a contradiction occurs. If this is not the case, then, as a consequence of the examination, a new state, which contains additional information, is derived. In that way, the transition from one state to another is assured, and the amount of information increases in each step. When the actual state has a sufficient amount of information, a part, or the whole initial table is found.

Initially, there is not any piece of information (an empty current state). The algorithm starts with the general conditions represented by the nodes at the first level. These nodes, as well as the nodes at some other level are examined from left to right (Fig. 2) so that the most promising states are favoured. Thus, at level i , we first examine the C^i_1 node. If the general condition represented by this node is satisfied, a new state is created, and the algorithm passes to the first node at the next level, C^{i+1}_1 (denoted by the solid line in Fig. 2). Otherwise, if a contradiction occurs, the algorithm examines the next node at the current level, in this case C^i_2 (denoted by the dashed line). The similar holds for all other nodes at the given level. As can be observed, if the general condition in some node is satisfied, then the next node to be examined is always the first one at the next level. Therefore, in Fig. 2, the first node at each level is specially denoted as a unique node at the level that can be reached from the previous level. When the state with all information needed for recovering the initial table is achieved, the algorithm stops. We expect that our algorithm inevitably leads to the solution because the set of available information is expanded in each step. This conclusion intuitively follows from nature of this information (the value of some table entry is equal to or different from some number, and the set of possible numbers is limited).

2.3. Complexity analysis

The main characteristic of each deciphering cryptanalytic algorithm is its complexity. The complexity is measured by the total number of activities or operations that are necessary to perform until the solution is found. What these activities represent depends on the individual algorithm. For our algorithm, the complexity can be estimated as the number of general conditions (or nodes) that have to be examined until the initial cipher table is found. The reason for such a decision lies in the fact that the general condition examination alters the state of the system. Therefore, the complexity actually represents the number of states that should be passed.

To estimate the complexity, it is necessary to determine how many times on the average each node in the tree given in Fig. 2 is examined until the moment when the solution is found and then to sum these values. It is convenient to calculate the cumulative for each level in the tree. Let $v_s(i,k)$ denote the probability that the k th node at the level i leads to the solution. Then, if the solution is found at time $t = m$, the complexity can be calculated in the following way:

```

Complexity = 0
FirstOnLevel = 1 /* number of times for the first left node at the level */
For i from 1 to m - 1
  Value = FirstOnLevel
  For k from 2 to V(i)
    c = 1
    For j from 1 to k - 1
      c = c - Vs(i, j)
    Value = Value + FirstOnLevel * c
  FirstOnLevel = Value /* cumulative per level */
Complexity = Complexity + Value

```

As can be seen, the cumulative value for one level represents the average number of examinations of the first node at the next level. The number of examinations of remaining nodes at the level is calculated by multiplying the number that corresponds to the first node at this level, and the probability that the given node is reached, i.e. none of preceding nodes at this level was satisfied.

3. Attacking the RC4

The proposed general approach is applied to the cryptanalysis of the RC4 stream cipher. As given in [14], the internal state of RC4 at time t consists of a permutation table S_t of 2^n different n -bit values and of two n -bit pointers i_t and j_t . For the pointers i_0 and j_0 initialized to zero, the RC4 algorithm can be described as

$$i_t = t, \quad (4)$$

$$j_t = j_{t-1} + S_{t-1}(t), \quad (5)$$

$$S_t(t) = S_{t-1}(j_t), S_t(j_t) = S_{t-1}(t), \quad (6)$$

$$Z_t = S_t(S_t(t) + S_t(j_t)). \quad (7)$$

As can be seen, a new table is generated in each iteration by the permutation of two elements of the existing table. Therefore, the parameter s from the Section 2.1 becomes $s = 2$. The elements that are being permuted at time t are at the positions $q_t^1 = i_t$ and $q_t^2 = j_t$ in the table S_{t-1} and at positions $p_t^1 = j_t$ and $p_t^2 = i_t$ in the table S_t , respectively.

Eq. (7), which describes the generation of the output symbol Z_t , can be written as

$$Z_t = S_t(L_t), \quad L_t = S_t(t) + S_t(j_t). \quad (8)$$

The content of the S_t table can be given by

$$S_t(t) = S_{t-1}(j_t), \quad (9)$$

$$S_t(k \neq t) = S_{t-1}(k), \quad k \neq j_t, \quad (10)$$

$$S_t(k \neq t) = S_{t-1}(t), \quad k = j_t. \quad (11)$$

Applying Eqs. (9)–(11), there follows that Eq. (8) can be represented by the tree structure shown in Fig. 3. For larger t , the tree grows very fast, and the number of conditions that have to be checked becomes enormous. Striving to reduce the search space, we propose a method to replace the entire set of particular conditions with one analytically formulated general condition.

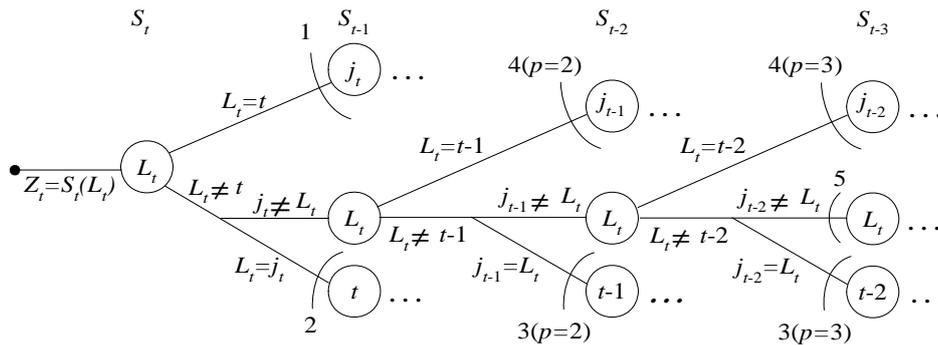


Fig. 3. Tree representation of the RC4 algorithm output symbol Z_t .

Combining Eqs. (5)–(8), there follows:

$$S_{t-1}(t) = j_t - j_{t-1}, \tag{12}$$

$$S_{t-1}(j_t) = L_t - j_t + j_{t-1}. \tag{13}$$

On the basis of Eqs. (12) and (13), it is possible to replace the whole subtrees whose roots correspond to the expressions like $S_{k-1}(k)$ and $S_{k-1}(j_k)$ by more general conditions. One general condition can be checked instead of all conditions from the given subtree. For example, the subtree rooted to the node $Z_t = S_{t-1}(j_t)$ (denoted by 1 in Fig. 3) is replaced by the general condition C1 (Table 1) in such a way that the condition on the path to this node $L_t = t$ is combined with the node equation and Eq. (13). In a similar way, the other general conditions are derived.

Since the tree is partitioned into $2t + 1$ parts, the general conditions, denoted as $C_i, i = 1, \dots, 5$, corresponding to the subtrees indicated by numbers from 1 to 5, are given in Table 1. As can be seen, the conditions C3 and C4 encompass $t - 1$ conditions each. Also, they do not exist at the initial time. The expression $e(Z_t)$ denotes the position of Z_t in the initial table. All calculations are modulo 2^n .

The probabilities that general conditions lead to the solution are given in Table 2. In order to further reduce the search space, we propose the checking of the conditions in a decreasing order of their probabilities, while the existing approaches are based on arbitrary guessing.

The general conditions are organized in a tree similar to the tree represented in Fig. 2 with the following difference: $V(t) = 2t+1$. Our attack on the RC4 is based on the hill-climbing strategy for the search of this tree.

Before we calculate the complexity of our attack by applying the algorithm presented in Section 2.3, we have to determine the number of levels, m , in the tree of general conditions. This number represents the average number of output symbols that should be known to find the solution. For the given output word length n , m is the smallest number such that

Table 1
General conditions

C1	C2	C3($p \in [2, t]$)	C4($p \in [2, t]$)	C5
$j_t = t - Z_t + j_{t-1}$	$j_t = Z_t + j_{t-1}$	$Z_t = j_{t-p+1} - j_{t-p}$	$Z_t = L_{t-p+1} - j_{t-p+1} + j_{t-p}$	$L_t \neq t, \dots, 1, j_t, \dots, j_1$
$L_t = t$	$L_t = j_t \neq t$	$L_t = j_{t-p+1} \neq t, \dots, t - p + 1, j_t, \dots, j_{t-p+2}$	$L_t = t - p + 1 \neq j_t, \dots, j_{t-p+2}$	$L_t = e(Z_t)$
$S_{t-1}(t) = t - Z_t$	$S_{t-1}(t) = Z_t$	$S_{t-p}(t - p + 1) = Z_t$	$S_{t-p+1}(t - p + 1) = Z_t$	$S_{t-1}(j_t) = e(Z_t) - j_t + j_{t-1}$
$S_{t-1}(j_t) = Z_t$	$S_{t-1}(j_t) = j_{t-1}$	$S_{t-1}(j_t) = j_{t-p+1} - j_t + j_{t-1}$	$S_{t-1}(t) = j_t - j_{t-1}$	$S_{t-1}(t) = j_t - j_{t-1}$
		$S_{t-1}(t) = j_t - j_{t-1}$	$S_{t-1}(j_t) = t - p + 1 - j_t + j_{t-1}$	

Table 2
Probabilities of the general conditions

C1	C2	C3($p \in [2, t]$)	C4($p \in [2, t]$)	C5
$1/2^n$	$(2^n - 1)/2^{2n}$	$(2^n - 1)^{p-1} (2^n - p)/2^{n(p+1)}$	$(2^n - 1)^{p-1}/2^{np}$	$(2^n - 1)^t (2^n - t)/2^{n(t+1)}$

Table 3
The required number of output symbols

Word size n	3	4	5	6	7	8
m	6	13	27	55	111	227

Table 4
Complexity of the cryptanalytic attacks on n -bit RC4

Word size n	3	4	5	6	7	8
Complexity	1.6×10^2	2.5×10^7	4.3×10^{21}	7.4×10^{57}	1.4×10^{146}	3.7×10^{363}

$$\prod_{i=1}^m (2i + 1) \succ 2^n! \tag{14}$$

The values of m for different n are given in Table 3.

The results of complexity obtained for the RC4 versions with different word size n are presented in Table 4. Although a significant improvement over exhaustive search (2.1×10^{509} for $n = 8$) has been made, it is still not enough to practically menace the security of the RC4 stream cipher.

4. Improving the backtracking algorithm

This section presents the incorporation of our approach based on the general conditions into the RC4 attack given in [6] in strive to further improve its efficiency that was the best published one until now. In the first subsection we present the modification of the original algorithm. Since in the modified backtracking algorithm the general conditions are used in a specific environment implied by the original algorithm, they can be simplified in a way presented in Section 4.2. The effects of their examination are given in Section 4.3. The complexity analysis of the modified backtracking algorithm is given in Section 4.4.

4.1. Modified backtracking algorithm

The idea of the backtracking attack was, going through the steps of RC4 algorithm and taking into account the current state of the table, to guess the values of entries in the table that have not already been assigned in order to ensure the next update of RC4. In the case of contradiction, the backtracking proceeds. The guessed values are chosen from the set of allowed values, in a simple arbitrary manner, one after one. Thus, some of them are favoured without any particular reason.

This algorithm can find the correct initial state of the cipher using a small segment of output stream. Since the number of the assignments made for all entries in the initial table until reaching the solution determines the complexity of attack, some improved strategy of selecting the values to be assigned could be very useful.

This finding has motivated our modification of the original algorithm, which is based on introducing some kind of strategy of choosing the values from the set of allowed ones. To make it possible, we have incorporated the general conditions into the original algorithm. There were two reasons for it. First, each general condition at the time t contains the information about the contents of $S_{t-1}(t)$ and $S_{t-1}(j_t)$. Such information can help to extract the set of possible values from the set of unassigned values. The second reason was that for every general condition, its probability of leading to the solution is known. Therefore, checking of the conditions in a decreasing order of their probabilities at the time t favors the sets of possible values which lead to the solution with higher probabilities.

In order to present the modified search algorithm for n -bit RC4 we will use a formalism given in Ref. [6]. Let us assume that at a given time t there are a previously assigned values in the table. Let P denote the set of possible values for the given general condition (see Section 4.3).

- Step 1:* Check whether $S_{t-1}(i_t)$ has been assigned a value:
- If it has, go to Step 2.
 - If it has not, check the general condition with the highest probability at level t . If this condition cannot be satisfied, check the next one at the same level, and so on. If none of them is satisfied, a contradiction occurs. If the checking of some condition is successful, assign a value from P to $S_{t-1}(i_t)$, increment a , and then, depending on which general condition is checked (see Section 4.2), proceed to time $t + 1$ and go to Step 1, or go to Step 2.
- Step 2:* Check whether the value Z_t has already been assigned:
- If it has, calculate the value of $S_{t-1}(j_t)$ from Eq. (7) in the RC4 description. If this does not lead to a contradiction, increment t and proceed to Step 1.
 - If it has not, go to Step 3.
- Step 3:* Check whether $S_{t-1}(j_t)$ has been assigned a value:
- If it has not, check whether the general conditions are already examined in Step 1. If they are, update obtained information (see Section 4.2); otherwise, check the general condition with the highest probability at level t . If this condition cannot be satisfied, check the next one at the same level, and so on. If none of them is satisfied, a contradiction occurs. If the checking of some condition is successful, assign a value from P to $S_{t-1}(j_t)$, increment a , and then, check the Eq. (7). If this does not lead to a contradiction, proceed to time $t + 1$ and go to Step 1.

The general conditions are successively examined in order to choose the value for assignment, which is the most promising for further evaluation. All other values are temporarily discarded. These values can be reconsidered if the previous choice does not lead to the solution. Again, the next most suitable value is picked, and the process iterates until the solution is found.

4.2. Modified conditions

Since the cryptanalytic algorithm on the RC4 stream cipher given in Ref. [6] guarantees the next update of the RC4 steps at each time t , all parameters j_i and L_i , $0 < i < t$, which appear in the general conditions, are known. In this way, the level of uncertainty decreases, since some parts of general conditions are instantiated and, therefore, they are ready for validation. Taking this into account, on the basis of the general conditions, we have derived the simplified, *modified conditions*. The modified conditions have the following structure:

- the precondition part, which has to be satisfied at a given time t to contribute to the solution,
- the condition that has to hold for $S_{t-1}(i_t)$,
- the condition that has to hold for $S_{t-1}(j_t)$.

In respect to this structure, we can represent the modified conditions in the following form:

- for time $t = 1$ (Table 5), the modified conditions are the same as the general ones because that is the initial state,
- for time $t > 1$, the modified conditions M_i , $i = 1, \dots, 5$ are listed in Table 6. The known values of parameters j_i are denoted as J_i , $0 < i < t$. As can be seen, the general conditions C1 and C5 are modified according to the algorithm steps in which they are examined.

Table 5
Modified conditions for $t = 1$

Modified condition	Precondition part	Conditions for $S_{t-1}(i_t)$	Conditions for $S_{t-1}(j_t)$
IS1	$Z_1 \neq 0$	$S_0(1) = 1 - Z_1$	$S_0(1 - Z_1) = Z_1$
IS2	$Z_1 \neq 0, 1$	$S_0(1) = Z_1$	$S_0(Z_1) = 0$
IS3		$S_0(1) \neq Z_1$	$S_0(j_1) \neq 0, Z_1$

Table 6
Modified conditions for $t > 1$

Modified condition	Precondition part	Conditions for $S_{t-1}(i_t)$	Conditions for $S_{t-1}(j_t)$
<i>M1</i> , Step 1			
(a) If Z_t assigned	$S_{t-1}(t - Z_t + J_{t-1}) = Z_t$	$S_{t-1}(t) = t - Z_t$	
(b) If Z_t not assigned		$S_{t-1}(t) = t - Z_t$	$S_{t-1}(t - Z_t + J_{t-1}) = Z_t$
<i>M1</i> , Step 3	$S_{t-1}(t) = t - Z_t$		$S_{t-1}(t - Z_t + J_{t-1}) = Z_t$
<i>M2</i>	$J_{t-1} \neq Z_t, t - Z_t$	$S_{t-1}(t) = Z_t$	$S_{t-1}(Z_t + J_{t-1}) = J_{t-1}$
<i>M3</i> (p)	$Z_t = J_{t-p+1} - J_{t-p}$ $S_{t-p+1}(J_{t-p+1}) = Z_t$ $J_{t-p+1} \neq t, t - 1, \dots, t - p + 1,$ $J_{t-1}, \dots, J_{t-p+2}$	$S_{t-1}(t) \neq J_{t-p+1} - J_{t-1}$	$S_{t-1}(j_t) \neq J_{t-1}$
<i>M4</i> (p)	$Z_t = L_{t-p+1} - J_{t-p+1} + J_{t-p}$ $S_{t-p+1}(t - p + 1) = Z_t$ $t - p + 1 \neq J_{t-1}, \dots, J_{t-p+2}$	$S_{t-1}(t) \neq t - p + 1 - J_{t-1}$	$S_{t-1}(j_t) \neq J_{t-1}$
<i>M5</i> , Step 1			
(a) If Z_t assigned	$e(Z_t) \neq t, t - 1, \dots, 1,$ J_{t-1}, \dots, J_1	$S_{t-1}(t) \neq e(Z_t) - J_{t-1}$	$S_{t-1}(j_t) \neq J_{t-1}$
(b) If Z_t not assigned			
If $v \in \{1, \dots, t - 1, t, J_1, \dots, J_{t-1}\}$		$S_{t-1}(t) \neq Z_t$	$S_{t-1}(j_t) \neq J_{t-1}$
If $v \notin \{1, \dots, t - 1, t, J_1, \dots, J_{t-1}\}$		$S_{t-1}(t) \neq e(Z_t) - J_{t-1}$	$S_{t-1}(j_t) \neq J_{t-1}$
<i>M5</i> , Step 3			$S_{t-1}(J_t) \neq J_{t-1}$

We have already mentioned that our search algorithm checks the modified conditions in its first and third step. The same modified condition has to be checked differently for each step, because the states which precede these steps are different. Before checking the conditions in the first step, the value of $S_{t-1}(i_t)$ is unknown, while in the state that precedes the third step, this value is known, and Z_t was not assigned before, and the value of $S_{t-1}(j_t)$ is unknown.

By analysing the modified conditions, we have observed certain dependencies among them. For example, it can be proved that at time t only one of the *M3* conditions can be satisfied. A similar conclusion is valid for the *M4* conditions, too. These relations between the modified conditions are important for the checking process, because the number of examined conditions is reduced, and, therefore, the execution of the algorithm is faster.

4.3. Effects of the examination of the modified conditions

The primary aim of the modified conditions examination is to get new information which can help to extract a set of possible values for actual assignment. In this subsection we explain, for each modified condition, how these values can be chosen among the values which have not already been assigned. Besides, the other effects important for the algorithm are presented.

At given time t , let a denote the number of the assigned entries in the table. The set of S_d values which have not been assigned is S . As a consequence of the modified condition checking, we expect to get the set of possible values P , which has P_d elements.

For time $t = 1$, the initial table S_0 is empty, therefore $a = 0$.

The initial states IS1 and IS2 are checked only in Step 1(b) of the algorithm that produces the following effects:

- (1) set $P_d = 1$, since $S_0(1)$ can have only one value, $P = \{1 - Z_1\}$ for IS1, or $P = \{Z_1\}$ for IS2;
- (2) increment a , because besides $S_0(1)$, the value $S_0(1 - Z_1)$ for IS1, or the value $S_0(Z_1)$ for IS2 has been found, too;
- (3) denote the transition to Step1, since it is known that the condition in Step 2 is satisfied, thus it should not be checked separately.

The initial state IS3 is also examined only in Step 1(b). These are the effects of this checking:

- (1) set $P_d = S_d - 1$, since $S_0(1)$ must be different from Z_1 ;
- (2) save the information that $S_0(j_1) \neq 0$, Z_1 , because it can be used in Step 3(a);
- (3) denote the transition to Step 2.

For time $t > 1$, the modified conditions $M1$ to $M5$ are relevant.

The modified condition $M1$ can be examined in the first or third step of the algorithm. If it is checked in Step 1, the following effects occur:

- (1) set $P_d = 1$, since $S_{t-1}(t)$ can have only one value, $t - Z_t$;
- (2) set $a = a + (1 - (a + 1)/2^n)^2$, because besides the value $S_{t-1}(t)$, the value $S_{t-1}(J_t)$ can be found if the position J_t was not already used, and if Z_t has not been already assigned. The probability of each of these events is $1 - (a + 1)/2^n$ where term $(a + 1)$ stands because besides the value a which was assigned earlier, the value $S_{t-1}(t)$ is also known now;
- (3) denote the transition to Step 1, since it is known that the condition in Step 2 is already satisfied, and it should not be separately checked.

As the effect of the checking of modified condition $M1$ in Step 3, we have

- (1) set $P_d = 1$, since $S_{t-1}(J_t)$ can have only one value, Z_t ;

The modified condition $M2$ can be examined only in Step 1, and effects are the same as for $M1$ in step 1, except for the fact that the sole value that $S_{t-1}(t)$ can have is Z_t .

The effects of the $M3(p \in [2, t])$ and $M4(p \in [2, t])$ modified conditions checking are the same:

- (1) set $P_d = S_d - (1 - a/2^n)$, because the value from which $S_{t-1}(t)$ must differ belongs to the set S with the probability $1 - a/2^n$;
- (2) save the information that $S_{t-1}(j_t) \neq J_{t-1}$, because it can be used in Step 2(a);
- (3) denote the transition to Step 2.

If the modified condition $M5$ is examined in the first step, the effects are the same as for $M3$ or $M4$. If it is checked in Step 3, it follows that:

- (1) set $P_d = S_d - (1 - a/2^n)$, since the probability that value J_{t-1} belongs to set S is $(1 - a/2^n)$.

4.4. Efficiency of the attack

In order to estimate the complexity of our attack, the modified search algorithm can be presented in three steps, analogously to the formulation in Ref. [6]. Let us assume that the initial table S_0 has 2^n elements. At some given time t , let a denote the number of the known entries in the table, and V is an average number of the possible values that can be assigned. The value V is calculated taking into account the effects obtained by the modified conditions examination (see Section 4.2) and related probabilities.

1. Check if $S_{t-1}(i_t)$ is already known:
 - (a) If yes, go to Step 2.
 - (b) If not, check the modified conditions at level t . If this does not lead to a contradiction, assign the V values to $S_{t-1}(i_t)$, increment a , and then, depending on the results of checking, proceed to time $t + 1$ and go to Step 1, or go to Step 2.

2. Check if the value Z_t is already assigned:
 - (a) If yes, calculate the value of $S_{t-1}(j_t)$ on the basis of the equation for Z_t in the RC4 description. If a contradiction does not occur, increment t and proceed to Step 1.
 - (b) If not, go to Step 3.
3. Check if $S_{t-1}(j_t)$ is already known:
 - (a) If not, check the modified conditions at level t if they have not been checked before, or update the information obtained in the case when the modified conditions have already been checked (in Step 1). After that, assign the V values to $S_{t-1}(j_t)$ and increment a . Check Eq. (7) from the RC4 description, and then, if it does not lead to a contradiction, increment t and go to Step 1.

The complexity of the attack given in Ref. [6] is measured by the total number of assignments made for all entries of the initial table until the solution is found. In order to calculate the complexity, three functions $c_i(a)$, $i = 1, 2, 3$ are defined. Each of them represents the complexity of a related individual step of the algorithm. Similarly, we can determine these functions according to our search algorithm given above. First of all, we separate the initial time $t = 1$, $a = 0$, $S_d = 2^n$, in which only initial conditions IS1, IS2 and IS3 are examined. To find the function $c_1(0)$, we check IS1 first. The probability that it will succeed, $v(\text{IS1})$, is given in Table 2 (corresponds to C1). The effects obtained in this case (see Section 4.3) show that two entries of the initial table are known, and the transition to step 1 is set. In a similar way the initial condition IS2 is checked. Finally, we check the IS3 condition whose probability corresponds to C5 in Table 2. According to the effects described in Section 4.3 for this case, we do one assignment for every of $P_d = 2^n - 1$ possible values of $S_{t-1}(i_t)$, and go to Step 2. So, we have

$$c_1(0) = v(\text{IS1})c_1(2) + v(\text{IS2})c_1(2) + v(\text{IS3})(2^n - 1)c_2(1).$$

Using a similar approach, we can derive the equations for $c_2(1)$ and $c_3(1)$:

$$c_2(1) = c_3(1).$$

$$c_3(1) = (1 - 1/2^n)[(1 - 2/2^n)(2^n - 3) + 2/2^n(2^n - 2)](1/(2^n - 2) + (1 - 1/(2^n - 2))c_1(3)).$$

For time $t > 1$, the table contains a assigned values, so that $S_d = 2^n - a$. Now, we will explain how the equation for $c_1(a)$ is derived. The answer to test in Step 1 is positive with the probability $a/2^n$. If the answer is negative (probability $1 - a/2^n$), we will examine the modified conditions. If we check the conditions M1 or M2, only one value for assignment is allowed, while for the other ones, M3 to M5, there are on average $2^n - a - (1 - a/2^n)$ possible values. If we approximate $a + (1 - (a + 1)/2^n)^2$ with a (we can do it because it increases the complexity), there follows:

$$c_1(a) = (a/2^n)c_2(a) + (1 - a/2^n) \left[(v(M1) + v(M2))c_1(a + 1) + (v(M5) + \sum_{p=2}^t (v(M3(p)) + v(M4(p))))(2^n - a - (1 - a/2^n))c_2(a + 1) \right].$$

Similarly, we have developed the expressions for $c_2(a)$ and $c_3(a)$:

$$c_2(a) = (1 - a/2^n)c_3(a) + (a/2^n)[(a/2^n)(1/2^n)c_1(a) + (1 - a/2^n)(a/2^n + (1 - a/2^n)c_1(a + 1))],$$

$$c_3(a) = (1 - a/2^n)[(1 - a/2^n)(2^n - a - (1 - a/2^n)) + (a/2^n)(v(M1) + v(M5)(2^n - a - (1 - a/2^n)))]((a + 1)/2^n + (1 - (a + 1)/2^n)c_1(a + 2)).$$

The probabilities of modified conditions correspond to the probabilities of general conditions C1–C5 (Table 2). That is justified because, due to the averaging in the previous time step, it can be assumed that in the current time step the system is in the state that leads to the solution.

Table 7
Complexity of the cryptanalytic attacks on n -bit RC4

Word size n	3	4	5	6	7	8
Knudsen et al. [6]	2^8	2^{21}	2^{53}	2^{132}	2^{324}	2^{779}
Our attack	2^5	2^{17}	2^{46}	2^{120}	2^{300}	2^{731}

The complexity of our attack can be estimated by the analytical calculation based on the given functions. We start with the known expressions $c_i(2^n)$. Then, going backwards, we calculate the total number of assignments given by $c_1(0)$.

The results of the calculation for the RC4 versions with different word size n are presented in Table 7. Compared to the estimated complexity of the original algorithm, an improvement has been made definitely. However, it is still not enough to practically menace the security of the RC4 stream cipher.

5. Conclusions

We have proposed the general approach to the cryptanalysis of a class of table-shuffling based stream ciphers, which promotes an idea of using all information available at a given time. The information is organized into the trees from which the general conditions are analytically extracted. After that, the general conditions are, according to their probabilities, ordered into the tree of general conditions, which is searched using the hill-climbing strategy in order to reconstruct the initial state. The algorithm for complexity estimation is developed.

The general approach is applied to the RC4 stream cipher and a new cryptanalytic algorithm is formulated. The estimated complexity of this algorithm is lower than for the exhaustive search. If a sufficient number of output words is known, the deciphering process is successful. Otherwise, the initial table will not be completed. Therefore, in this case, after the last known output word is examined, we have the set of information valid at this time. This information can be used to reduce the search space of some other cryptanalytic attack on the RC4 stream cipher. Thus, the general conditions have been incorporated in the existing backtracking algorithm in order to make a better choice for the assignment to unknown entries of the cipher's table. The complexity of backtracking algorithm has been further decreased below the best published result, but the RC4 remains a quite secure cipher for practical applications.

References

- [1] E. Biham, L. Granboulan, P. Nguyen, Impossible fault analysis of RC4 and differential fault analysis, Fast Software Encryption 2005, Lecture Notes in Computer Science 3557 (2005) 359–367.
- [2] S. Fluhrer, I. Mantin, A. Shamir, Weakness in the key scheduling algorithm of RC4, Selected Areas in Cryptography 2001, Lecture Notes in Computer Science 2259 (2001) 1–24.
- [3] S. Fluhrer, D. McGrew, Statistical analysis of the alleged RC4 Keystream Generator, Fast Software Encryption 2000, Lecture Notes in Computer Science 1978 (2000) 19–30.
- [4] J. Golić, Linear statistical weakness of alleged RC4 Keystream Generator, Advances in Cryptology – EUROCRYPT '97, Lecture Notes in Computer Science 1233 (1997) 226–238.
- [5] A. Grosul, D. Wallach, A related-key cryptanalysis of RC4, Rice University, Technical Report TR-00-358, June 2000.
- [6] L. Knudsen, W. Meier, B. Preneel, V. Rijmen, S. Verdoolaege, “Analysis methods for (alleged) RC4”, Advances in Cryptology – ASIACRYPT '98, Lecture Notes in Computer Science 1514 (1998).
- [7] I. Mantin, Predicting and distinguishing attacks on RC4 Keystream generator, Advances in Cryptology – EUROCRYPT 20 05, Lecture Notes in Computer Science 3494 (2005) 491–506.
- [8] I. Mantin, A. Shamir, “A practical attack on broadcast RC4”. Fast Software Encryption FSE 2001, Lecture Notes in Computer Science 2355 (2002) 152–164.
- [9] S. Mister, S. Tavares, Cryptanalysis of RC4-like ciphers, Selected Areas in Cryptography – SAC '98, Lecture Notes in Computer Science 1556 (1999) 131–143.
- [10] S. Paul, B. Preneel, A new weakness in the RC4 Keystream generator and an approach to improve the security of the cipher, Fast Software Encryption FSE, Lecture Notes in Computer Science 3017 (2004) 245–259.
- [11] S. Paul, B. Preneel, Analysis of non-fortuitous predictive states of the RC4 Keystream generator, INDOCRYPT, Lecture Notes in Computer Science 2904 (2003) 52–67.

- [12] J. Pearl, *Heuristics – Intelligent search strategies for computer problem solving*, Addison-Wesley Series in Artificial Intelligence, 1984.
- [13] A. Roos, A class of weak keys in the RC4 stream cipher, *Sci. Crypt.*, September 1995.
- [14] B. Schneier, *Applied Cryptography*, Wiley, New York, 1996.
- [15] V. Tomašević, S. Bojanić, O. Nieto-Taladriz, An approach for analysis of stream ciphers based on table-shuffling. *The State of the Art of Stream Ciphers*, Workshop Record, Brugge, October 2004, pp. 165–174.
- [16] R.R. Yager, OWA trees and their role in security modeling using attack trees, *Information Science* 176/20 (2006) 2933–2959.

Author's personal copy