INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2018, VOL. 64, NO. 2, PP. 137–145
Manuscript received September 30, 2017; revised March, 2018. DOI: 10.24425/119361

Tabu Search Against Permutation Based Stream Ciphers

Iwona Polak and Mariusz Boryczka

***Abstract*—Encryption is one of the most effective methods of securing data confidentiality, whether stored on hard drives or transferred (e.g. by e-mail or phone call). In this paper a new state recovery attack with tabu search is introduced. Based on research and theoretical approximation it is shown that the internal state can be recovered after checking 2^{52} internal states for RC4 and 2^{180} for VMPC.**

***Keywords*—tabu search, TS, cryptanalysis, RC4, VMPC, stream cipher, state recovery attack**

I. Introduction

PEOPLE each day communicate with each other. The formal, mathematical definition of this process was given by Shannon and Weaver [1]. Then on this basis Shannon in 1949 created a model of secret communication, where the content of the message is not known to the outsiders [2].

Encryption surrounds us even if we are not fully aware of it. It is implemented wherever the protection of transmitted or stored information is crucial. Everyone uses encryption when talking on the phone, logging in to the bank account through the Internet or shopping online.

To maintain high quality of cryptographic protection both already being in use and newly created ciphers need to be widely verified. In this paper a state recovery attack on two stream ciphers is presented. The ciphers are RC4 [3], [4] and VMPC [5]. The presented method is based on optimization technique called tabu search.

The goal is to generate such a keystream that is identical with the keystream being analysed, and thereby find the internal state of the cryptographic algorithm. For RC4 the correct internal state can be found after checking approximately 2^{52} internal states. The method is compared to another nature-inspired technique, namely Genetic Algorithm. Details of this technique can be found in [6].

Because there exist stronger attacks on RC4 [7], the presented method was also tested on another permutation-based cipher VMPC. For this stream cipher the correct internal state can be found after checking approximately 2^{180} internal states. While it is still a big number, the improvement over brute force and other known attacks is well seen.

In both cases only 256 (2^8) B of a keystream are analysed. Experiments are made for 512 iterations and using regression analysis and extrapolation it is predicted how many iterations are needed to completely reproduce the keystream and thereby find a proper internal state of the cipher.

Authors are with the Institute of Computer Science, University of Silesia, Poland (e-mail: {iwona.polak, mariusz.boryczka}@us.edu.pl).

In Section II some basic facts about stream ciphers are compiled. Also a detailed description of the RC4 and the VMPC is given here. For every cipher known attacks are summarized.

Section III contains a brief summary of tabu search method. It also provides a detailed exposition of a new state recovery attack on permutation-based ciphers. In order to explain the idea more clearly a simple example is described.

In Section IV we discuss a research that was made using the presented state recovery attack with tabu search. The section contains more closely look on the conditions of experiments and parameters values that were used. After this a detailed exposition of achieved results is provided, separately for RC4 and VMPC. Results are compared with known attacks and with the probability of random guess.

Section V contains a brief summary of the article and it discusses further work.

II. Stream ciphers

Modern cryptography is divided into symmetric-key cryptography and public-key cryptography. The first one is then divided into block and stream ciphers (Fig. 1).

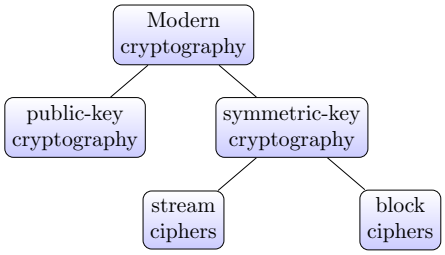

Fig. 1. Classification of modern ciphers

Encryption and decryption in symmetric-key cryptography in the simplest way can be defined as follows:

$$\text{Encryption: } E_K(P) = C \tag{1}$$

$$\text{Decryption: } D_K(C) = P \tag{2}$$

where:
E – encryption,
D – decryption,
K – key,

P – plaintext,
C – ciphertext.

In order to obtain a valid cryptographic algorithm, these functions must have the following property:

$$D_K(E_K(P)) = P \tag{3}$$

In stream ciphers usually the key is not used directly, but it is used to generate keystream κ of much longer period than the key itself. In most cases the keystream is added to the plaintext with the use of eXclusive OR (XOR) function. So in stream ciphers the decryption and encryption processes are defined as follows:

$$\text{ENCRYPTION: } P_n \oplus \kappa_n = C_n \tag{4}$$

$$\text{DECRYPTION: } C_n \oplus \kappa_n = P_n \tag{5}$$

where:
P_n – n-th bit of plaintext,
κ_n – n-th bit of keystream,
C_n – n-th bit of ciphertext,
$\oplus$ – eXclusive OR (XOR).

For example:

$$\begin{array}{rcl} P: & & 10011101011100011011011000110111... \\ \kappa: & \oplus & 11100111110101000011110100101100... \\ \hline C: & & 01111010101001011000101100011011... \end{array}$$

Of course decryption and encryption in stream ciphers satisfy property from Eq. 3, because:

$$C_n \oplus \kappa_n = (P_n \oplus \kappa_n) \oplus \kappa_n = P_n \oplus \kappa_n \oplus \kappa_n = P_n \oplus 0 = P_n \tag{6}$$

Most of the stream ciphers' algorithms are divided into two phases:

- Key Scheduling Algorithm, KSA – a phase in which the value of the secret key K and the apparent initialization vector (IV) is dissipated in the internal structure of the algorithm,
- Pseudo-Random Generation Algorithm, PRGA – the generation of the keystream κ; in this phase neither the key nor the IV are used any more.

Matching the internal state of the encryption algorithm immediately after the KSA will allow to generate a subsequent keystream without the need for a secret key. And then, without the need to know the key, one will be able to decipher the secret message. The attack, which aim is to guess or calculate the internal state, is called state recovery attack. Because of the negligibility of the KSA phase in these considerations, it will be omitted in the descriptions of cryptographic algorithms in subsequent paragraphs.

Stream ciphers presented below share the same internal structure, namely a permutation of 256 integer values, from 0 to 255. Both of them produce one byte of keystream every clock. Then a byte of keystream is XORed with a byte of plaintext and as a result it produces one byte of a ciphertext. Furthermore internal state changes with every clock. However, KSA and PRGA phases are different in RC4 and in VMPC.

A. RC4

RC4 is a stream cipher designed by Ron Rivest in RSA Security in 1987. Initially it was a trade secret, but after several years the algorithm was anonymously sent to the Cypherpunks mailing list [3]. The RC4 cipher has variable key length (up to 2048 bits), which sets the initial permutation of numbers in the 256-element array S. At a given time it can be in one of $256! \cdot 256 \cdot 256 \approx 2^{1700}$ possible states. The keystream is the same length as the plaintext and is independent of it. The algorithm has been implemented e.g. in SSL/TLS (Secure Socket Layer/Transport Layer Security) to encrypt traffic on the Internet and in WEP (Wired Equivalent Privacy) and WPA (WiFi Protected Access) to secure wireless networks [4]. In its time it was probably the most widely used stream cipher in the world. Its popularity is due to the ease of implementation in both software and hardware. PRGA of the RC4 is presented in Fig. 2 and 3, where:
i, j – 8-bit variables,
S – 256-byte array with a permutation of unsigned integer numbers {0, 1, ..., 255},
L – the length of the plaintext (in bytes),
$+$ – addition mod 256.

```
1: i ← 0
2: j ← 0
3: for b ← 1 to L do
4:     i ← i + 1
5:     j ← j + S[i]
6:     swap (S[i], S[j])
7:     output ← S[ S[i] + S[j] ]
8: end for
```

Fig. 2. RC4 PSEUDO-RANDOM GENERATION ALGORITHM

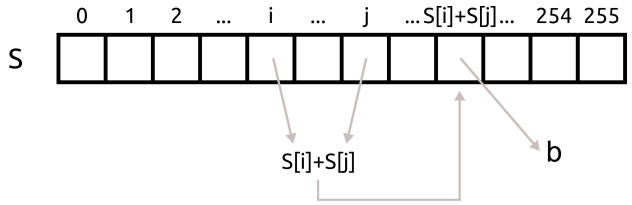

Fig. 3. RC4 encryption scheme

1) Known attacks: Exhaustive key space search requires checking 2^{2048} possibilities. Checking all possible initial internal states will reduce the number of opportunities to 2^{1684} (256!).

There is an attack with the use of another metaheuristic, namely Genetic Algorithm (GA), by B. Ferriman and C. Obimbo [6]. This technique is based on the phenomenon of evolution and its main components are: selection, crossover and mutation. There were two versions considered: non-adaptive and adaptive. The attack was a state recovery attack. GA allowed for 10-25% fit for 256-byte keystream. Authors estimate that finding a proper internal state will take 2^{122} generations (with 5 solutions in each generation).

RC4 has been subject to wide cryptanalysis for last 30 years. The most important attacks are presented by Mathy Vanhoef

and Frank Piessens [7]. They demonstrated an attack on WPA-TKIP (Wi-Fi Protected Access – Temporal Key Integrity Protocol), which can be executed within an hour. The same authors have also designed a practical approach to recovery the cookie of TLS protocol used in HTTPS, with success rate of 94%, using 2^{30} ciphertexts.

B. VMPC

VMPC is a function and stream cipher published by Bartosz Zoltak in 2004 [5]. The name is an acronym for Variably Modified Permutation Composition. VMPCrypt software, written by the same author, is used for internal purposes by Polish companies, institutions and individuals.

The definition of the VMPC function for the n-element permutation f is as follows:

$$g(x) = VMPC(f(x)) = f(f(f(x)) + 1) \ mod \ n \quad (7)$$

According to the author's declarations, the VMPC algorithm offers much better statistical properties than the RC4. The internal state of the VMPC may take $256! \cdot 256 \cdot 256 \approx 2^{1700}$ values (256-element permutation and two 8-bit variables). The initial internal state is calculated on the basis of a secret key and an explicit initialization vector. The key length, like in RC4, is variable (here from 128 to 512 bits). With each generated byte of a keystream, the internal state of the cipher changes. The security of the cipher is based on the assumption that the $g(x)$ is a one-way function. PRGA of the VMPC is presented in Fig. 4, where:
i, j – 8-bit variables,
S – 256-byte array with a permutation of unsigned integer numbers {0, 1, ..., 255},
L – the length of the plaintext (in bytes),
$+$ – addition mod 256.

```
1: i ← 0
2: for b ← 1 to L do
3:     j ← S[ j + S[i] ]
4:     output ← S[S[S[j]]+1]
5:     swap (S[i], S[j])
6:     i ← i + 1
7: end for
```

Fig. 4. VMPC Pseudo-Random Generation Algorithm

1) Known attacks: Inverting the base function, which would break the VMPC stream cipher, requires an estimated average of 2^{260} computational operations [5]. Alexander Maximov published distinguishing attack on this cipher with $O(2^{40})$ computational complexity [8]. From other distinguishing attacks [9]–[11] the best one requires 2^{24} B of keystream. A distinguishing attack allows an attacker only to distinguish the keystream from random bitstream, but does not reveal any information about the key or the secret message.

III. Tabu search

The tabu search technique (TS) was introduced by F. Glover in 1986 [12], [13]. The main idea of TS is to build upon previous experiences to make better decisions in the future. An important element of this technique is the memory of previous movements, which in some way leads searching in the space of possible solutions. This memory is represented as a tabu list (hence the name) of the most recently executed movements, which are simultaneously prohibited movements in the nearest future. Amount of iterations in which a movement is forbidden is horizon parameter.

The TS is a metaheuristic search method based on local search and is used for mathematical optimization. In every step neighbouring solutions of current solution are considered and the best one from the current neighbouring set is chosen. Now this best solution becomes current solution. A movement that was made is placed on tabu list. As the next current solution, one that is not on tabu list is selected, regardless of whether the new solution has a greater value of the fitness function than the current solution or not. This should break the solution out of the local optimum. Presented in Subsection III-A technique is based on the basic version of TS.

The tabu search is an iterative search of the neighbourhood in the solutions space. However, in order not to run infinitely, there is a certain limitation on the duration of the algorithm. This limitation is called a termination criterion and can be defined in different ways depending on ones needs. The most common conditions of termination are:

- after a predetermined number of iterations or a certain amount of processor time,
- after a number of consecutive iterations, during which no improvement in the quality of the solution occurred,
- after finding a solution with a predetermined quality (usually defined as a specific value of the fitness function).

In spite of its simplicity, TS gives very good results where exact algorithms are not known or intractable in practice. For this reason, the current applications of TS cover many areas, e.g. bioinformatics, pattern classification, energy distribution, resource and space planning, molecular engineering, telecommunication, financial analysis or waste management [13].

A. State recovery attack

The main goal is to find a permutation that generates the analysed byte stream, in other words, to find the cipher's internal state from the beginning of the communication. Discovery of the internal state in this moment is synonymous with key knowledge – it allows one to decrypt the rest of the secret message.

Based on the tabu search, a cryptanalysis method of permutation-based stream ciphers was developed. The proposed algorithm for cryptanalysis is shown in Fig. 5. This attack is a plaintext attack, which means that an attacker has access to a plaintext and a ciphertext associated with it. From those two a keystream can be calculated. The attack is also a state recovery attack, because the main goal is to recover the internal state of the cipher, right after KSA and before the start of the communication.

The cryptanalytic algorithm starts with generating a random solution r (random internal state, random permutation) (Fig. 5, line 1). In the presented work, the candidate solution (internal

1: current permutation $r \leftarrow$ random permutation
2: TABU $\leftarrow \{\}$
3: **while** termination criterion not satisfied **do**
4: identify set $R(r)$ of possible 2-element swap on r
5: $r \leftarrow$ best permutation $\in R(r) \wedge \notin$ TABU
6: update TABU
7: **end while**
8: **return** best permutation found

Fig. 5. CRYPTANALYSIS WITH TABU SEARCH

state) is any permutation of non-negative integers from 0 to 255. The tabu list is empty at the beginning (Fig. 5, line 2).

Then a set of neighbouring solutions of r is identified $R(r)$ (Fig. 5, line 4). The neighbouring solution is also a permutation, almost identical to the considered one. They differ by two elements that have been swapped with each other. So the size of the neighbourhood $|R(r)|$ in this case is 32 640 ($\approx 2^{15}$).

From this neighbourhood $R(r)$ the best solution is selected, i.e. the solution with the highest value of fitness function (see Sec. III-A1) (Fig. 5, line 5). Solutions, which emerged by making swaps that are in the tabu list (TABU), are not taken into consideration, independently of their quality. The chosen solution becomes a new current solution. In every iteration only one solution is maintained and this solution is modified in each subsequent iteration by 2-element swap.

The procedure for updating the tabu list (Fig. 5, line 6) consists of listing the currently executed movement (permutation elements swap) and deleting the oldest entry if necessary (i.e. an entry that is in the list for more than *horizon* iterations).

Changing the current solution and updating the tabu list is made until the termination criterion is reached (Fig. 5, line 3).

1) The fitness function: The fitness function is a measure of quality of solutions. In this case the fitness function is the percentage of correctly guessed bytes in the keystream generated by the analysed solution (eq. 8). In the research (Sec. IV) the length of the analysed keystream is arbitrarily set to 256 bytes.

$$f_{fit} = \frac{B}{256} \tag{8}$$

where:
f_{fit} – fitness function,
B – number of bytes correctly guessed in the keystream.

The best possible result is 1, which means that the analysed byte stream and the keystream generated by the candidate solution are 100% consistent. We assume that at this point the internal state of the attacked cipher was recovered. The internal state of presented ciphers change with every generated byte, so it seems highly unlikely that incorrect initial internal state would produce correct 256 B of a keystream and after this would desynchronize (see Sec. IV-C).

The f_{fit} may be represented as a number from range $\langle 0, 1\rangle$ or the percentage value from range $\langle 0\%, 100\%\rangle$. For clarity, in this paper only the second convention will be used.

2) Example: Let us assume that the cryptosystem's internal state (permutation) is of size 5 and the size of the tabu list (horizon) is |TABU| = 3. Random initial permutation (internal state) was generated as follows:
r = 4, 2, 3, 0, 1
At the beginning the tabu list is empty:
TABU = {}.

There are 10 neighbouring solutions (elements that were swapped are underlined), so the set $R(r)$ consists of:
2, 4, 3, 0, 1
3, 2, 4, 0, 1
0, 2, 3, 4, 1
1, 2, 3, 0, 4
4, 3, 2, 0, 1
4, 0, 3, 2, 1
4, 1, 3, 0, 2
4, 2, 0, 3, 1
4, 2, 1, 0, 3
4, 2, 3, 1, 0

Every solution has a value of the fitness function. Let us assume that those values are as follows:
2, 4, 3, 0, 1 : $f_{fit} = 9.4\%$
3, 2, 4, 0, 1 : $f_{fit} = 8.2\%$
0, 2, 3, 4, 1 : $f_{fit} = 12.1\%$
1, 2, 3, 0, 4 : $f_{fit} = 10.2\%$
4, 3, 2, 0, 1 : $f_{fit} = 10.5\%$
4, 0, 3, 2, 1 : $f_{fit} = 11.7\%$
4, 1, 3, 0, 2 : $f_{fit} = 7.8\%$
4, 2, 0, 3, 1 : $f_{fit} = 13.3\%$
4, 2, 1, 0, 3 : $f_{fit} = 9.8\%$
4, 2, 3, 1, 0 : $f_{fit} = 7.4\%$
The solution with the best fitness was marked with bold text.

The internal state with the best fitness is chosen for the next iteration as current solution. So now:
r = 4, 2, 0, 3, 1
and the swap between 3^{rd} and 4^{th} elements is placed on the tabu list:
TABU = {(3,4)}.

The next step (next iteration) is to consider all neighbouring solutions of the new current solution. They are:
$R(r)$ =
2, 4, 0, 3, 1 : $f_{fit} = 12.5\%$
0, 2, 4, 3, 1 : $f_{fit} = 8.6\%$
3, 2, 0, 4, 1 : $f_{fit} = 9.8\%$
1, 2, 0, 3, 4 : $f_{fit} = 10.2\%$
4, 0, 2, 3, 1 : $f_{fit} = 7.4\%$
4, 3, 0, 2, 1 : $f_{fit} = 12.9\%$
4, 1, 0, 3, 2 : $f_{fit} = 11.3\%$
~~4, 2, 3, 0, 1 : $f_{fit} = 10.9\%$~~
4, 2, 1, 3, 0 : $f_{fit} = 13.3\%$
4, 2, 0, 1, 3 : $f_{fit} = 10.5\%$
The internal state that is achieved through the swap that is in the tabu list is struck out. This internal state is not taken into account.

The best solution from this iteration is now the current solution:

r = 4, 2, 1, 3, 0
and the swap between 3^{rd} and 5^{th} element is placed on the tabu list:
TABU = {(3,4), (3,5)}.

The set of all neighbouring solutions of the current one looks as follows, $R(r)$:
2, 4, 1, 3, 0 : $f_{fit} = 20.3\%$
1, 2, 4, 3, 0 : $f_{fit} = 18.4\%$
3, 2, 1, 4, 0 : $f_{fit} = 19.9\%$
0, 2, 1, 3, 4 : $f_{fit} = 19.1\%$
4, 1, 2, 3, 0 : $f_{fit} = 16.0\%$
4, 3, 1, 2, 0 : $f_{fit} = 17.6\%$
4, 0, 1, 3, 2 : $f_{fit} = 14.5\%$
~~4, 2, 3, 1, 0 : $f_{fit} = 7.4\%$~~
~~4, 2, 0, 3, 1 : $f_{fit} = 13.3\%$~~
4, 2, 1, 0, 3 : $f_{fit} = 9.8\%$

The new current solution for the next iteration is:
r = 2, 4, 1, 3, 0
and
TABU = {(3,4), (3,5), (1,2)}.

The set $R(r)$ consists of:
~~4, 2, 1, 3, 0 : $f_{fit} = 13.3\%$~~
1, 4, 2, 3, 0 : $f_{fit} = 16.0\%$
3, 4, 1, 2, 0 : $f_{fit} = 13.2\%$
0, 4, 1, 3, 2 : $f_{fit} = 14.8\%$
2, 1, 4, 3, 0 : $f_{fit} = 13.6\%$
2, 3, 1, 4, 0 : $f_{fit} = 13.4\%$
2, 0, 1, 3, 4 : $f_{fit} = 15.2\%$
~~2, 4, 3, 1, 0 : $f_{fit} = 18.0\%$~~
~~2, 4, 0, 3, 1 : $f_{fit} = 12.5\%$~~
2, 4, 1, 0, 3 : $f_{fit} = 12.9\%$

As it can be seen the solution with the best fit is on tabu list, so it will not be chosen to the next iteration. For the next iteration will be chosen a solution that has the best fit among permitted swaps. Not only the best possible swap is not made, but also a solution with worse fitness value than the previous one is accepted. Thus the new current solution is:
r = 1, 4, 2, 3, 0.

The swap, which led to this permutation, is placed in the tabu list. But at the same time the oldest movement is deleted, because list's length is 3. Thus:
TABU = {(3,5), (1,2), (1,3)}.

And so on, until the termination criterion is reached. After exiting the main `while` loop the procedure will return the internal state (permutation) with the highest value of the fitness function found during the whole algorithm run. This returned value is not necessarily from the last iteration. In the above example, this will be a permutation $(2, 4, 1, 3, 0)$ with $f_{fit} = 20,3\%$ from iteration 3.

3) Two-element swap: Let us assume that all possible permutations of m numbers form a graph. Two vertices are connected if and only if permutations are different only by two elements that are swapped with each other. An example for permutations of size $m = 3$ is given in Fig. 6.

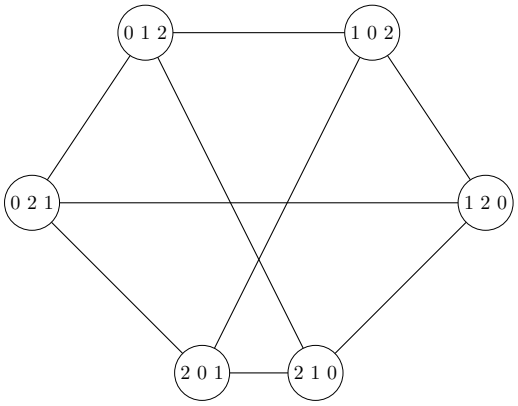

Fig. 6. Graph of permutations of size 3

The question is: what is the length of the longest path (without cycles) in such a graph? In other words: having some permutation, what is the maximum number of 2-element swaps to get any other permutation? Let us assume that from some (random) permutation we want to get permutation in which elements are sorted in ascending order. This can be achieved with the selection sort. From m elements choose the maximum one and swap it with the last one (2-element swap). In the next step choose from $m-1$ elements (as the last one is already sorted) and swap it with the one before last. And so on, until the whole set is sorted. To sort m elements there are at most $m-1$ swaps necessary.

The same reasoning can be applied to get any permutation from some other permutation. Thus for m numbers to get one permutation from some other one by 2-element swaps there are $m-1$ swaps needed at most. Therefore in presented in this paper ciphers at most 255 ($\approx 2^8$) 2-element swaps are needed to get any permutation from random one. And in particular, a finite number of 2-element swaps is enough to get wanted permutation from a random permutation.

IV. Research

Main aim of the research is to find internal state of the cipher. This will lead to revealing the successive keystream. The attack is a known plaintext attack – it is assumed that 256 bytes of plaintext is known, which consequently gives exact knowledge about the same amount of keystream. In presented research there are 2 ciphers cryptanalysed: RC4 (see Sec. II-A) and VMPC (see Sec. II-B) as they share similar structure, namely permutation of 256 values from 0 to 255. There are 10 different keys used for each cipher – values of those can be found in Tables II and III.

A. Conditions of Experiments

In this research termination condition was set as an arbitrarily chosen number of iterations in order to see, how the internal state and its f_{fit} would change in time. In this case the number of iterations was chosen as 512 ($= 2^9$). From the method's behaviour in this limited time, an approximation of further development of the function is made.

The initial internal state (permutation) is constructed randomly. In every iteration swap of two numbers is evaluated – the best swap passes to the next iteration (and is put on tabu list for some amount of consecutive iterations). There were four types of horizon considered:

- 1,
- $\log_2(|R(r)|)$,
- $\sqrt{|R(r)|}$,
- $\frac{|R(r)|}{2}$

where $|R(r)|$ is the size of the neighbourhood, i.e. the amount of possible 2-element swaps, here: $|R(r)| = 32\ 640\ (\approx 2^{15})$.

The fitness value is the percentage of keystream bytes that are the same in keystream produced by the candidate solution and cipher being attacked (see Sec. III-A1, eq. 8).

For every configuration set there were 32 tests executed. At the end of every test the best permutation found during the whole test was returned, regardless of iteration number in which it was found.

Comparatively random solutions (internal states) were generated in the number of 16 711 680 for every key. 16 711 680 is equal to 512 · 32 640, which is the number of solutions evaluated in a single cryptanalysis test using TS.

B. Results

As the presented technique contains a random component conclusions were drawn on the basis of average of returned values from those 32 tests. Also tables and figures contain average of returned values, unless it is clearly stated otherwise.

1) RC4: Results for RC4 are presented in Table IV. Every result for TS is a pair of obtained value of f_{fit} (above) and iteration number, in which the best solution was found (below). In columns there are results for different types of horizons. For every of them minimum, average and maximum value of 32 runs is showed. Also random generated solutions are presented at the end. A minimum, average and maximum value of 16 711 680 tries is showed there. In rows there are results presented for every key (see Table II) separately and also total results for all the keys altogether at the bottom.

When there is only one prohibited movement on the tabu list, the best score is 19.4%. The best result is found in up to 60 (of 512) iterations and later no improvement occurs. Probably the algorithm falls into a local optimum and it is unable to get out of it. It is therefore apparent that the tabu list has a significant impact on the quality of the solutions obtained.

The best results were obtained for horizon $\sqrt{|R(r)|}$. Fitness function value in consecutive iterations for this horizon is presented in Fig. 7. There is also a logarithmic regression chart marked as red line to approximate the trend. The equation of the line of the regression is:

$$f(x) = 0.0377018742 \ln(x) + 0.028359542 \qquad (9)$$

Using this extrapolation, it can be predicted that the keystream will be recreated after approximately 2^{37} iterations.

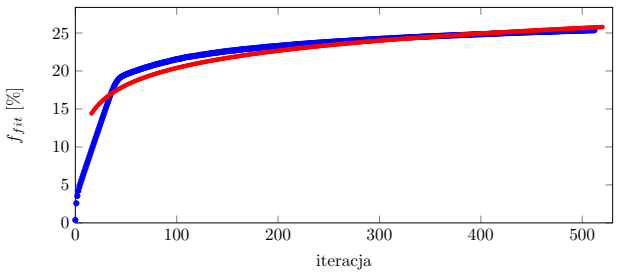

Fig. 7. Fitness function value in consecutive iterations for RC4 (horizon $\sqrt{|R(r)|}$) – blue dots; logarithmic regression chart is marked as red line

In every iteration $\approx 2^{15}$ potential solutions are being checked. Thereby the correct internal state will be guessed after checking 2^{52} internal states on average.

Results for RC4 were compared to those achieved by genetic algorithm (GA) [6] in Table V. Results from GA are an average from 25 runs as it also contains a random component. In 1 000 000 iterations GA on average found internal state that covered 20.7% of bytes correct (adaptive). Population of every iteration consisted of 5 candidates. In 512 iterations TS on average found internal state that covered 25.4% of bytes (65 of 256 B) correct (horizon $\sqrt{|R(r)|}$). A neighbourhood of size 32 640 ($\approx 2^{15}$) was checked in every iteration. Improvements of the GA and TS through generations and iterations are logarithmic functions. Using regression analysis and extrapolation it can be predicted that on average the keystream could be completely reproduced by GA after $\approx 2^{122}$ generations of $5 \approx 2^2$ candidates (details in [6]), which gives $\approx 2^{124}$ internal states being checked. Whereas for TS it can be predicted that on average the keystream could be reproduced after $\approx 2^{37}$ iterations of $\approx 2^{15}$ candidates each, which gives $\approx 2^{52}$ internal states checks. While it is still a big number, the improvement is well seen.

2) VMPC: Results for VMPC are presented in Table VI. Every result for TS is a pair of obtained value of f_{fit} (above) and iteration number, in which the best solution was found (below). In columns there are results for different types of horizons. For every of them minimum, average and maximum value of 32 runs is showed. Also random generated solutions are presented at the end. A minimum, average and maximum value of 16 711 680 tries is showed there. In rows there are results presented for every key (see Table II) separately and also total results for all the keys altogether at the bottom. Horizon equals 1 was not considered as in the experiments with RC4 it was shown that the presented algorithm is ineffective without the tabu list.

The best results were obtained for horizon $\log_2(|R(r)|)$. It was 10.0% bytes (26 of 256 B) correct in 512 iterations. Fitness function value in consecutive iterations for this horizon is presented in Fig. 8. There is also a logarithmic regression chart marked as red line to approximate the trend. The equation of the line of the regression is:

$$f(x) = 0.0082806371 \ln(x) + 0.0553808841 \qquad (10)$$

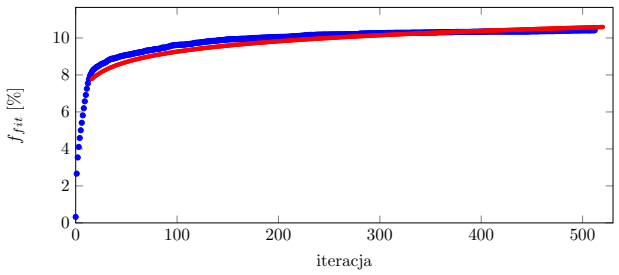

Fig. 8. Fitness function value in consecutive iterations for VMPC (horizon $\log_2(|R(r)|)$) – blue dots; logarithmic regression chart is marked as red line

Using this extrapolation, it can be predicted that the keystream will be recreated after approximately 2^{165} iterations. In every iteration $\approx 2^{15}$ potential solutions are being checked. Thereby the correct internal state will be guessed after checking 2^{180} internal states on average.

Although this number is outside the scope of the practical breaking the VMPC cipher, this is a much better result than the best known attack with complexity 2^{260}. Furthermore, the further research in this topic may lead to the development of an effective VMPC cryptanalysis algorithm. All other presented in Section II-B attacks are distinguishing attacks and as it was stated before – such an attack does not reveal any information neither about the key nor the plaintext. Therefore attack presented in this paper is stronger as it could help in decrypting the ciphertext.

C. Probability of random guess

Let us compare the results with random draw of 256 integer numbers from the range $\langle 0, 255\rangle$. Let p denote the probability of success, i.e drawn number is the same as the number in the byte stream (keystream). Then $1-p$ denotes the probability of failure, i.e. drawn number is different than the number in the byte stream (keystream). Probability of k successes in n tries can be counted from Bernoulli Scheme:

$$\binom{n}{k} p^k (1-p)^{n-k} \tag{11}$$

Probability of single success p in this case is equal to $\frac{1}{256}$. And number of tries $n = 256$. Thus the equation will be as follows:

$$\binom{256}{k} \left(\frac{1}{256}\right)^k \left(\frac{255}{256}\right)^{256-k} \tag{12}$$

Probabilities of selected amounts of correct numbers are given in Table I.

Cumulative probability of guessing less than 5 numbers is 0.9964. Cumulative probability of guessing up to 10 numbers is 0.999999992. Therefore the probability of guessing correctly more than 10 numbers is $0.000000008 \approx 2^{-27}$. The attack with presented in the paper technique was able to reconstruct 65 numbers for RC4 and 28 numbers for VMPC in 512 iterations. As it can be seen from Bernoulli Scheme guessing so many numbers by chance is very improbable.

TABLE I
PROBABILITY OF GUESSING NUMBERS CORRECTLY IN 256 NUMBERS STREAM

numbers correct k	probability	
0	0.3672	$\approx 2^{-1.4}$
1	0.3686	$\approx 2^{-1.4}$
2	0.1843	$\approx 2^{-2.4}$
3	0.0612	$\approx 2^{-4}$
4	0.0152	$\approx 2^{-6}$
5	0.0030	$\approx 2^{-8}$
6	0.0005	$\approx 2^{-11}$
7	$6.89 \cdot 10^{-5}$	$\approx 2^{-14}$
8	$8.41 \cdot 10^{-6}$	$\approx 2^{-17}$
9	$9.09 \cdot 10^{-7}$	$\approx 2^{-20}$
10	$8.81 \cdot 10^{-8}$	$\approx 2^{-23}$
⋮		
26	$2.71 \cdot 10^{-28}$	$\approx 2^{-92}$
⋮		
65	$7.76 \cdot 10^{-96}$	$\approx 2^{-316}$
⋮		
256	$3.09 \cdot 10^{-617}$	$\approx 2^{-2048}$

Moreover probability of guessing a series of 256 numbers equals $3.09 \cdot 10^{-617} = 2^{-2048}$. It confirms that internal state, which generates attacked byte keystream, is almost for sure the right one.

V. CONCLUSION

In this paper a new state recovery attack was presented. It was based on tabu search. It is constructed for permutation-based stream ciphers.

Based on research and theoretical approximation it was shown that the internal state can be recovered after checking 2^{52} internal states for RC4 and 2^{180} for VMPC.

Further work will focus on applying the attack to other permutation-based ciphers, e.g. RC4+ [14] or Spritz [15]. Interesting question is whether the method will be at least the same good for a cipher RC4A [16], which consists of two arrays of permutations.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, July 1948.
[2] C. Shannon, *Communication Theory and Secrecy Systems*. Bell Telephone Laboratories, 1949. [Online]. Available: https://books.google.pl/books?id=8IL3HAAACAAJ
[3] "RC4 Source Code," Cypherpunks, September 1994, http://cypherpunks.venona.com/archive/1994/09/msg00304.html.
[4] B. Harris, "Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer Protocol," January 2006. [Online]. Available: http://tools.ietf.org/html/rfc4345
[5] B. Zoltak, "VMPC One-Way Function and Stream Cipher," in *Fast Software Encryption*, ser. Lecture Notes in Computer Science, B. Roy and W. Meier, Eds. Springer Berlin Heidelberg, 2004, vol. 3017, pp. 210–225. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-25937-4_14
[6] B. Ferriman and C. Obimbo, "Solving for the RC4 stream cipher state register using a genetic algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 5, pp. 218–223, May 2014.

TABLE II
RC4 – KEYS USED FOR EXPERIMENTS (HEXADECIMAL); CHOSEN FROM OFFICIAL TEST VECTORS [17]

ID	key
1-0	0x0102030405
1-1	0x01020304050607
1-2	0x0102030405060708090a
1-3	0x0102030405060708090a0b0c0d0e0f10
1-4	0x0102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f20
1-5	0x833222772a
1-6	0x1910833222772a
1-7	0x8b37641910833222772a
1-8	0xebb46227c6cc8b37641910833222772a
1-9	0xc109163908ebe51debb46227c6cc8b37641910833222772a

TABLE III
VMPC – KEYS AND IV VECTORS USED FOR EXPERIMENTS (HEXADECIMAL); THE FIRST PAIR IS THE TEST VECTOR [5]

ID	key	IV
2-0	0x9661410ab797d8a9eb767c21172df6c7	0x4b5c2f003e67f39557a8d26f3da2b155
2-1	0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	0x4b5c2f003e67f39557a8d26f3da2b155
2-2	0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	0x55555555555555555555555555555555
2-3	0x00000000000000000000000000000000	0x00000000000000000000000000000000
2-4	0x80000000000000000000000000000000	0x00000000000000000000000000000000
2-5	0x09090909090909090909090909090909	0x00000000000000000000000000000000
2-6	0x000102030405060708090a0b0c0d0e0f	0x00000000000000000000000000000000
2-7	0x288ff65dc42b92f960c70f62b5085bae	0x00000000000000000000000000000000
2-8	0x00000000000000000000000000000000	0x00000010000000000000000000000000
2-9	0x00000000000000000000000000000000	0x288ff65dc42b92f960c70f62b5085bae

TABLE IV
RESULTS OBTAINED FOR RC4 (MIN – MINIMUM, AVG – AVERAGE, MAX – MAXIMUM);
IN THE FIRST ROW THERE IS f_{fit}, IN THE SECOND – ITERATION NUMBER

horizon (function) (numerically)	1 1			$\log_2(\lvert R(r)\rvert)$ 15			$\sqrt{\lvert R(r)\rvert}$ 181			$\lvert R(r)\rvert/2$ 16 320			random 16 711 680		
key ID	min	avg	max	min	avg	max	min	avg	max	min	avg	max	min	avg	max
1-0	15.6%	18.8%	23.0%	18.0%	23.0%	27.3%	22.3%	25.8%	29.3%	22.7%	25.7%	30.5%	0.0%	0.4%	3.5%
	33	40	47	39	367	510	186	432	512	307	436	512			
1-1	16.4%	20.1%	25.0%	19.1%	23.0%	27.7%	23.0%	25.9%	29.3%	21.1%	26.0%	29.7%	0.0%	0.4%	3.5%
	36	44	59	90	350	512	313	421	510	330	444	510			
1-2	16.0%	19.5%	23.8%	20.3%	23.2%	27.7%	22.3%	25.2%	30.9%	21.9%	26.2%	31.3%	0.0%	0.4%	3.9%
	35	43	55	123	352	509	249	436	506	192	425	507			
1-3	15.6%	18.8%	21.5%	18.0%	22.8%	27.3%	22.7%	25.0%	28.5%	21.9%	25.2%	28.1%	0.0%	0.4%	3.9%
	33	41	51	53	303	511	305	424	510	275	454	510			
1-4	16.8%	19.3%	21.5%	20.3%	23.4%	28.1%	21.9%	25.2%	29.3%	21.9%	25.6%	30.9%	0.0%	0.4%	3.9%
	35	42	51	44	333	512	210	430	502	312	435	507			
1-5	15.2%	19.2%	23.0%	18.0%	22.7%	26.6%	20.7%	25.6%	28.9%	21.5%	25.6%	29.3%	0.0%	0.4%	4.3%
	32	41	50	63	340	509	367	461	508	231	439	511			
1-6	15.2%	19.8%	23.4%	20.7%	23.5%	26.6%	22.3%	25.4%	28.1%	22.7%	25.4%	28.1%	0.0%	0.4%	3.9%
	31	43	60	93	341	508	245	440	508	263	445	512			
1-7	16.0%	19.7%	24.2%	18.8%	22.4%	28.1%	21.9%	25.6%	28.5%	23.0%	25.5%	29.7%	0.0%	0.4%	3.9%
	32	43	54	55	332	507	259	411	505	213	409	508			
1-8	16.0%	19.1%	23.0%	16.8%	22.4%	26.2%	19.9%	25.4%	30.9%	22.7%	25.3%	29.3%	0.0%	0.4%	3.9%
	32	41	53	93	305	497	239	415	510	269	429	510			
1-9	16.4%	19.8%	24.2%	18.4%	22.7%	25.4%	21.9%	25.4%	28.1%	21.1%	25.2%	29.3%	0.0%	0.4%	3.5%
	31	43	58	92	329	504	247	422	501	289	431	506			
altogether	15.2%	19.4%	25.0%	16.8%	22.9%	28.1%	19.9%	25.4%	30.9%	21.1%	25.6%	31.3%	0.0%	0.4%	0.0%
	31	42	60	39	335	512	186	429	512	192	435	512			
							see Fig. 7								

TABLE V
COMPARISON OF RC4 CRYPTANALYSIS WITH GA AND TS

technique	number of iterations	·	solutions per iteration	= solutions checked		f_{fit}	bytes correct
GA non-adaptive	1 000 000	·	5	$\approx 2^{22}$	$\approx 10^7$	18.1%	46
GA adaptive	1 000 000	·	5	$\approx 2^{22}$		20.7%	53
TS $\log_2(\|R(r)\|)$	512	·	32 640	$\approx 2^{24}$		22.9%	59
TS $\sqrt{\|R(r)\|}$	512	·	32 640	$\approx 2^{24}$		25.4%	65
TS $\|R(r)\|/2$	512	·	32 640	$\approx 2^{24}$		25.6%	66
random	16 711 680			$\approx 2^{24}$		0.4%	1

TABLE VI
RESULTS OBTAINED FOR VMPC (MIN – MINIMUM, AVG – AVERAGE, MAX – MAXIMUM); IN THE FIRST ROW THERE IS f_{fit}, IN THE SECOND – ITERATION NUMBER

horizon (function) (numerically)	$\log_2(\|R(r)\|)$ 15			$\sqrt{\|R(r)\|}$ 181			$\|R(r)\|/2$ 16 320			random 16 711 680		
key-IV ID	min	avg	max	min	avg	max	min	avg	max	min	avg	max
2-0	6.3%	9.8%	12.5%	8.6%	10.8%	12.9%	8.6%	10.3%	13.3%	0.0%	0.4%	4.7%
	26	227	501	34	270	477	63	284	507			
2-1	7.4%	10.0%	14.5%	8.6%	10.4%	13.3%	8.2%	10.7%	13.7%	0.0%	0.4%	3.9%
	31	216	489	52	255	509	52	240	496			
2-2	5.9%	10.1%	12.1%	8.2%	11.0%	14.1%	8.6%	10.5%	12.9%	0.0%	0.4%	4.3%
	20	226	510	38	204	494	26	281	505			
2-3	6.6%	9.6%	12.9%	8.6%	11.0%	14.5%	8.6%	11.0%	13.3%	0.0%	0.4%	3.9%
	11	222	510	48	264	497	65	232	493			
2-4	7.8%	10.3%	14.8%	8.2%	11.1%	14.1%	8.2%	11.0%	13.3%	0.0%	0.4%	4.3%
	14	222	482	50	272	512	45	217	464			
2-5	5.9%	9.9%	12.5%	8.2%	11.0%	14.1%	8.2%	10.8%	12.5%	0.0%	0.4%	3.9%
	35	221	480	19	241	490	36	244	506			
2-6	7.0%	9.6%	13.7%	9.0%	11.0%	13.3%	7.0%	10.4%	12.9%	0.0%	0.4%	3.5%
	25	191	491	25	282	501	29	194	509			
2-7	7.0%	10.3%	13.7%	9.4%	11.1%	14.5%	8.6%	10.4%	12.1%	0.0%	0.4%	3.9%
	21	184	390	49	266	511	55	236	511			
2-8	7.4%	9.8%	13.3%	8.2%	10.6%	14.1%	7.4%	10.2%	13.7%	0.0%	0.4%	3.5%
	27	180	434	53	304	512	42	215	439			
2-9	7.4%	10.4%	14.8%	7.8%	10.8%	14.1%	8.6%	10.8%	13.7%	0.0%	0.4%	3.5%
	40	157	287	15	266	499	49	213	505			
altogether	5.9%	10.0%	14.8%	7.8%	10.9%	14.5%	7.0%	10.6%	13.7%	0.0%	0.4%	4.7%
	11	204	510	15	262	512	26	236	511			
	see Fig. 8											

[7] M. Vanhoef and F. Piessens, "All Your Biases Belong to Us: Breaking RC4 in WPA-TKIP and TLS," in *Proceedings of the 24th USENIX Conference on Security Symposium*, ser. SEC'15. Berkeley, CA, USA: USENIX Association, 2015, pp. 97–112. [Online]. Available: http://dl.acm.org/citation.cfm?id=2831143.2831150

[8] A. Maximov, *Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness of RC4 Family of Stream Ciphers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 342–358. [Online]. Available: https://doi.org/10.1007/11502760_23

[9] S. Li, Y. Hu, Y. Zhao, and Y. Wang, "Improved cryptanalysis of the VMPC stream cipher," *Journal of Computational Information Systems*, vol. 8, no. 2, pp. 831–838, 2012.

[10] S. Sarkar, "Further non-randomness in RC4, RC4A and VMPC," *Cryptography and Communications*, vol. 7, no. 3, pp. 317–330, 2015. [Online]. Available: https://doi.org/10.1007/s12095-014-0119-0

[11] Y. Tsunoo, T. Saito, H. Kubo, M. Shigeri, T. Suzaki, and T. Kawabata, "The Most Efficient Distinguishing Attack on VMPC and RC4A."

[12] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, May 1986. [Online]. Available: http://dx.doi.org/10.1016/0305-0548(86)90048-1

[13] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[14] S. Maitra and G. Paul, *Analysis of RC4 and Proposal of Additional Layers for Better Security Margin*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 27–39. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-89754-5_3

[15] R. L. Rivest and J. C. N. Schuldt, "Spritz—a spongy RC4-like stream cipher and hash function," August 19, 2014, presented at Charles River Crypto Day (2014-10-24).

[16] S. Paul and B. Preneel, *A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 245–259. [Online]. Available: https://doi.org/10.1007/978-3-540-25937-4_16

[17] J. Strombergson and S. Josefsson, "Test Vectors for the Stream Cipher RC4," May 2011. [Online]. Available: http://tools.ietf.org/html/rfc6229