A NEW ANALYSIS OF RC4
A Data Mining Approach (J48)

Mohsen HajSalehi Sichani and Ali Movaghar
School of Science and Engineering, Sharif University of Technology
International Campus, Persian Gulf, Kish Island, Iran

Keywords: Data mining, Information theory, J48 (C4.5), RC4, WEP.

Abstract: This paper combines the cryptanalysis of RC4 and Data mining algorithm. It analyzes RC4 by Data mining algorithm (J48) for the first time and discloses more vulnerabilities of RC4. The motivation for this paper is combining Artificial Intelligence and Machine learning with cryptography to decrypt cyphertext in the shortest possible time. This analysis shows that lots of numbers in RC4 during different permutations and substitutions do not change their positions and are fixed in their places. This means KSA and PRGA are bad shuffle algorithms. In this method, the information theory and Decision trees are used which are very powerful for solving hard problems and extracting information from data. The results of this Data mining approach could be used to improve the existing methods of breaking WEP (or other encryption algorithms) in less time with fewer packets.

1 INTRODUCTION

WEP is the most popular security method in wireless ad hoc networks. Tews et al. (E. Tews and Pyshkin, 2007) showed that in spite of its weakness, WEP is still popular among all users. The main part of WEP is RC4 which is responsible for Encryption. RC4 was designed by Ron Rivest in 1987 and was kept secret for 7 years until somebody sent it to a mailbox. From 1994 afterwards, many papers have been written based on statistics and mathematics. This paper; however, introduces a new approach which uses the data mining algorithms.

In recent years, data mining has been widely used in various areas of science and engineering and solved many serious problems in different areas of science such as electrical power engineering, genetics, medicine and bioinformatics. Data Mining is used to extract information from data. Data mining uses AI and Statistics in its algorithms.Information refers to patterns underlying data, and data refers to recorded facts. However, the captured data need to be converted into information and knowledge to become useful. Data mining is the entire process of applying computer-based methodology, including new techniques for knowledge conversion into data. Decision trees and information theory are chosen for this analysis based on personal experiments of authors for solving hard problems but other data mining algorithms such as MLP or Genetic Algorithms could also be used.

This paper introduces a new analysis of RC4 by data mining algorithms, a new method innovated for cracking WEP. The software used for implementing our method is free software named WEKA (Computer Science Department of University of Waikato, http://www.cs.waikato.ac.nz/ml/weka/).
We generated RC4 keys by the random function of Visual Studio 2008 software which is shown in Figure 2, based on FMS attack.
The generated data are given to WEKA for analysis. The outputs of data mining software showed many numbers are fixed during RC4 permutations. This analysis predicted more than 39% of 512000 instances correctly for module 64, and more than 38% of 512 instances (for training and test we used 10 fold-cross validation that cause instances repeated several times) correctly for module 256 which are great achievements in predicting keys. This new vulnerability can be used to crack WEP in less time with fewer packets. The result of this analysis again proves that KSA and PRGA are bad shuffle algorithms. The similar work can be done to analyze other encryption algorithms such as WEPplus, WEP512, AES, or DES to get new vulnerabilities. This method and results are explained in the next sections.

In section two, we describe RC4 and give a brief account of RC4. The phases of WEP are reviewed. Some attacks are mentioned and the FMS (Fluhrer, Mantin and Shamir 2001) attack (S. R. Fluhrer and Shamir, 2001) is studied.
In section three, we give an account of some important concepts of decision trees, and mention the Information Theory and its relation with C4.5 algorithm.
In section 4, we present the new analysis of RC4 based on FMS attack and Information Theory. We used WEKA for implementing C4.5 (J48) algorithm on the prepared data based on FMS attack.

2 RC4 ALGORITHM

RC4 is an algorithm which is responsible for encryption in WEP (Wu and Tseng, 2007). RC4 is composed of two main parts: KSA and PRGA. KSA is the acronym of Key Schedule Algorithm and PRGA is the acronym of Pseudo-Random Generator Algorithm.

2.1 KSA and PRGA

The 40-bit WEP key (K) concatenated with a 24-bit initialization vector (IV) forms the RC4 key (seed). The IV is generated automatically by a wireless adaptor and its length is 24 bits, and is shown here in three parts: first, 8 bits of IV(A), second, 8 bits of IV (B) and third, 8 bits of IV(X) which form (A,B,X). Each part can be from 0 to 255.

```
 KSA
1) for i=0 to N-1
2) S[i]=i
3) End
4) j=0
5) For i=0 to N-1
6) j=(j+S[i]+K[i mod l) mod N
7) Swap(S[i],S[j])
 PRGA
8) i=0
9) j=0
Output generator loop
10) i=(i+1) mod N
11) j=(j+ S[i] mod N
12) Swap(S[i], S[j])
13) Output= S[(S[i]+S[j]) mod N]
14) Cyphertext=Output XOR
    Plaintext
```

Figure 1: KSA and PRGA (RC4).

Maximum unique IV for the packets are limited to 2^{24}. Therefore, IV values are frequently reused in a busy network. One of the main flaws in RC4 is the repetition of the IV. The KSA and PRGA have been shown in Figure 1.

As shown in the Figure 1, there is an array with the length of 256 which is fulfilled with 0 to 255. In the first part of RC4 which is KSA, it makes a permutation with the help of the key entered by the user. In the second part, PRGA, it tries to make a new random array based on the output of KSA.

The main part of the algorithm is a pseudo random generator that produces one byte of output in each step. The encryption will be an XOR of the pseudo random sequence with the message (Plaintext) as usual for stream ciphers. For further information see (Wu and Tseng, 2007) and (Erickson, 2008).

2.2 Previous Attacks on RC4

In 1994, Finny (Finney, 1994) showed the class of states that RC4 never enters. These states are detemined by j=i+1 and S[j] =1. In 2001, Fluhrer et al. (S. R. Fluhrer and Shamir, 2001) introduced the FMS attack which uses a weakness in the key scheduling phase. The main idea is that RC4 uses the key with combination seed (session key)= IV||key. Thus for the number of packets greater than 2^{24}, the IVs will be repeated. If the IVs are chosen properly, then the first byte of the pseudo random sequence is the same as a predefined byte of the main key with probability ($\approx$ 1/e) (Klein, 2007), and (S. R. Fluhrer and Shamir, 2001). The idea behind FMS attack is used in data preparation for our experiment in data mining. The main idea of FMS attack is the following:

FMS showed that the weak IVs are in the form of (A + 3, B - 1, X), where A is the byte of the key to be attacked , B is 256, and X can be any value. So, for attacking, the form of the weak IV (3, 255, X) will be chosen by the hacker, where X is from 0 to 255. The bytes of the keystream must be attacked regularly. The first byte cannot be attacked until the zeroth byte is known and the second byte cannot be attacked until the first byte is known, and so on. The IVs can be attained from the packet because they are not encrypted.

If the attacker does not know the key because of this property which is found by FMS, he can perform A + 3 steps of KSA. If the zeroth byte of the key is known and A equals 1, then 1+3 equals 4 and the algorithm can be continued to find the second byte of the key in the fourth step.

Recently, A. Klein(Klein, 2007) showed a new method of attacking RC4 which uses the related key with a significantly reduced number of frames. He described a very general attack against RC4 when something about the distribution of plaintext is known but the Plaintext itself is unknown.

Tews et al.(E. Tews and Pyshkin, 2007) showed a special case of attacking WEP when a byte of plaintext is unknown, but could be restricted to some possible values.

3 C4.5 OR J48

C4.5 is an algorithm which is classified in decision tree algorithms. Decision trees and decision rules are data mining methodologies which are used in many applications for classification. Decision trees are supervised learning. Classification means mapping the input data (training and testing data) to one of the predefined classes. The optimal attribute is called the target or class. Classification is the process of assigning an attribute to the target, and a classifier is a model and the result of classification that predicts one attribute of a sample from the other attributes (Kantardzic, 2003).

C4.5 is a famous algorithm introduced by J. R. Quinlan. It is based on the Information Theory and Entropy. Information Theory was introduced by (Shannon, 1984) and (Shannon and Weaver, 1949). Further information on C4.5 is available in (J. R. Quainlan home page, http://de.scientificcommons.org/j_r_quinlan) and (Witten and Frank, 2005).

If *Samp* is any set of samples, $Numos(C_i, Samp)$ stands for the number of samples in the *Samp* that belongs to class C_i (there are several(K) classes), $|samp|$ denotes the number of samples in the set *Samp*.

Information value and entropy can be calculated by:

$$Info(Samp) = -\sum_{i=1}^{k}((Numos(C_i, Samp)/|Samp|)* \\ Log_2(Numos(C_i, Samp)/|Samp|)) \quad (1)$$

and

$$entropy(P_1, ..., P_n) = -P_1 Log P_1 - ... - P_n Log P_n \quad (2)$$

The arguments p_1, p_2 of the entropy formula are expressed as fractions that add up to one.

The task of selecting a possible test with n outcomes (n values for a given feature) that partitions set ToT of training samples into subsets ToT_1, ToT_2,..., ToT_n. The only information available for guidance is the distribution of classes in ToT and its subsets ToT_i.

ToT is the total number of rows in our data that belong to the target. ToT_i is the number of samples in one attribute which are mapped to C_i.

WEKA performs these equations for all attributes. Each attribute is shown by AttbX. The expected information requirement is the weighted sum of entropies over the subsets.

$$Info_{AtbX}(ToT) = -\sum_{i=1}^{n}((|ToT_i|/|ToT|)*Info(ToT_i)) \quad (3)$$

and then

$$Gain(AtbX) = Info(ToT) - Info_{AtbX}(ToT) \quad (4)$$

Gain (AttbX) should be maximum to be selected as the root or upper node in each step. Pruning the tree is also important. Good examples could be found at (Kantardzic, 2003) or (Witten and Frank, 2005).

```
1) for (int f = 0; f < 4000; f++)
2) {
3) a=Randnumber(32,126);
//passing time(by a loop)to keys be more random
4) b=Randnumber(32,126);
 //passing time(by a loop)
5) c=Randnumber(32,126);
  //passing time(by a loop)
6) d=Randnumber(32,126);
  //passing time(by a loop)
7) e=Randnumber(32,126);
  //passing time(by a loop)
8) seed= (3,63,1,a,b,c,d,e);//IV+User key
 save (seed);
9) }
10//Start of KSA
11) For  i = 0 to 63
12) S[i] = i;
13) j = 0;
14) For  i = 0 to 63
15) {
16)   j = (j + S[i] + K[i mod 8]) mod 64
17)   Swap(S[i], S[j])
18)   if (i == 0) //We did this saving
19)           //for permutations 0 to 3
20)       {
21)         Save S[0..63]
22)       }
23) }
24) //End of  KSA
25) //Start of PRGA
26) i = j = 0;
27) for (int x = 0; x < 63; x++)
28)  {
29)     i = (i + 1) % 64;
30)     j = (j + S[i]) % 64;
31)    Swap(S[i], S[j]) ;
32)    PrgaOutput[x] = S[(S[i] + S[j]) % 64];
33)    Save(PrgaOutput[0..63]);
34)  }
35)//End of PRGA
```

Figure 2: Pseudo-Code of our program for data generation.

4 NEW ANALYSIS

In this part, we combine the data mining algorithm with cryptanalysis of RC4. We implemented this idea via one of the data mining tools; namely, WEKA, which could be found in (Computer Science Department of University of Waikato, http://www.cs.waikato.ac.nz/ml/weka/). For data preparation we gathered our data based on the weakness of IVs in RC4 (S. R. Fluhrer and Shamir, 2001). To implement this, we make little changes in RC4 algorithm. These changes are mentioned in the next section.

4.1 Data Preparation

In the real world the key entered by the user is converted into ASCII code. Because of this property, we chose our keys from 32 to 126 of ASCII codes which are more normal in the real world and could be found at (http://www.asciitable.com).

As seen in Figure 2, we added the zeroth, first, second and third permutation to our data, based on the idea of FMS. Because of the need for more realistic results, we used the Dot Net (.Net) random function to generate more random keys. We concatenated IVs to the key as it happens in real transmission. These IVs can be obtained easily from the packet because, as we explained in previous sections, IVs do not encrypt in the header. We also chose the keys with a 5-byte length to be more realistic, as done in making ad hoc networks in Windows XP. Another change we made was to eliminate the XOR part from PRGA which is line number 14 in Figure 1. This is done for several reasons. First: the plaintext does not play any role except in the XOR part. The second and the most important reason is that many attacks are made based on the fixed or known content of some part of the cypher such as Snap Field, or known plaintext such as email content.

For the analysis of RC4, it is convenient to replace the original algorithm that works on bytes (Z/256Z) with Z/64Z. We did this because of hardware limitations. After all, the pseudo-code which is used for data generation is shown in Figure 2. On a large scale and as a complementary, we tested the idea on limited samples in module 256. As shown in Figure 3, we tested 4000 different seeds (line number 1). These seeds (line number 8) are composed of fixed IVs (3, 63, 1) based on FMS which are concatenated to ASCII codes of different keys. These keys are generated by random function of Dot Net (.NET). As shown in Figure 2, we produced five-byte keys through lines 3 to 7.

The word 'save' in Figure 2 shows that the data in front of 'save' are saved so that they are used later in data mining input file. As shown in Figure 2, line 8 saves the seeds, and the seed in Figure 3 is shown by key (@attribute key). And lines 11 to 35 are composed of two parts: KSA and PRGA. In KSA, we saved the first, second, third and 4th permutations (line 19 to 22) named atb0, atb1, atb2, and atb3, respectively. PRGA output, before XORing by plaintext, is saved in line 33 and is shown in Figure 3 by PRGAout (@attribute PRGAout).

The output of our program in Figure 2 is converted into the format expected by WEKA. The input data of WEKA is shown in Figure 3. The part under '@data' is our data produced by the program in Figure 2. As an example, we just showed two out of 256000 rows.

64 (our data in module 64) multiplied by 4000 (seeds) equals 256000. This is the total number of the rows of our data generated by program in Figure 2.

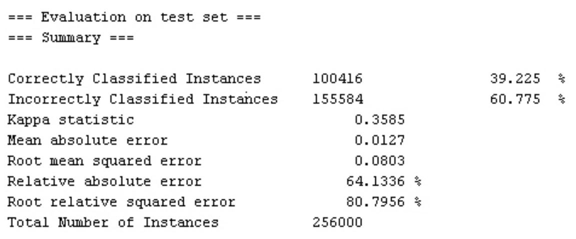
The data were given to WEKA and we used the supervised learning and classified the data. The Key attribute is our target which means the software and algorithm should predict the key value based on other attributes. We used algorithm J48 for our classification. The WEKA classifier package has its own version of C4.5 known as J48. We used 256000 rows of data for training and other 256000 rows (of data) for testing in WEKA. These two training and testing files had different data but in the same format. By adding up these two sets, we had 512000 rows of data composed of 8000 seeds.

```
@relation 'RC4-weka.filters.unsupervised.
attribute.Remove-R1,6-7-weka.filters.
unsupervised.attribute.
NumericToNominal-Rfirst-last'
@attribute atb0 {0,1,2,3,4,5,6,7,,61,62,63}
@attribute atb1 {0,1,2,3,4,5,6,7,,61,62,63}
@attribute atb2 {0,1,2,3,4,5,6,7,,61,62,63}
@attribute atb3 {0,1,2,3,4,5,6,7,,61,62,63}
@attribute key {1,3,32,33,34,...,123,124,125}
@attribute prgaout {0,1,2,3,4,5,6,7,. . .,60,
61,62,63}
@data
3,3,3,3,3,60
1,0,0,0,63,48
., ., ., ., .
```

Figure 3: Input data for WEKA.

4.2 Results of WEKA

These are the results (outputs) of WEKA:



```
=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances      100416              39.225  %
Incorrectly Classified Instances    155584              60.775  %
Kappa statistic                          0.3585
Mean absolute error                      0.0127
Root mean squared error                  0.0803
Relative absolute error                 64.1336 %
Root relative squared error             80.7956 %
Total Number of Instances           256000
```

Figure 4: Accuracy of the test.

4.2.1 Explanation of Results

Figure 4 shows that the algorithm could correctly predict more than 39 out of 100 keys, which is a great success in comparison to previous works (the exact number is 39.255).

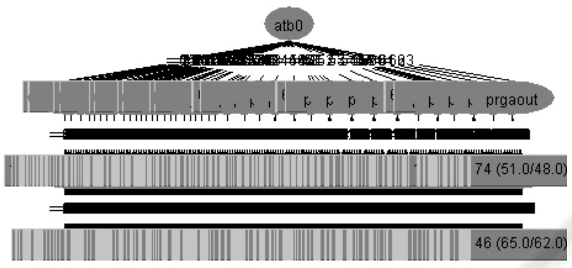


Figure 5: Visualized tree.

Figure 5 is the visualized tree of the output generated by WEKA. There is a lot of information in this part of output which is also similar to the non visualized tree of WEKA. This part will be discussed in more detail in the next lines.

```
| atb3=62:119(79.0/35.0)
| atb3=63:120(100.0/45.0)
atb0=1:63(4000.0)
atb0=2:1(4000.0)
atb0=3:3(4000.0)
atb0=4
| prgaout=0:88(58.0/55.0)
| prgaout=1:41(72.0/69.0)
```

Figure 6: Non visualized tree.

Figure 6 is the most important part of the output. This figure is just one part of the output result. The difference is that it is not visualized. The explanation of atb3=62:119 (79.0/35.0) will follow:
If atb3 which is the 4th permutation equals 62, then the target which is the seed value (attribute key) will be 119. Numbers in parentheses tell how many instances in the testing set are correctly classified by this node (79 instances in this case). The second number, if any, (if not, it is assumed to be 0.0), represents the number of instances incorrectly classified by the node (35 instances in this case) (J. Hakkila and Paciesas, 2009). Figure 6 also shows that in all 4000 seeds if the zeroth attribute which is the first permutation of KSA be equal to 1, then the key will be 63 in all 4000 seeds (atb0=1:63(4000.0)). The other results similar to Figure 6 are in Figure 7. These numbers are copied from the non visualized output of WEKA. Figure 7 shows that RC4 is not as strong in generating random outputs as it seems to be.

```
atb0=1:63(4000.0)
atb0=2:1(4000.0)
atb0=3:3(4000.0)
atb0=8:3(4000.0)
.
atb0=58:1(4000.0)
```

Figure 7: Important numbers of WEKA outputs.

4.3 Analysis on Module 256 as Complementary

So far this idea has been obtained that RC4 does not act good in permutations. Thus, to test this idea in real case, we changed the program in Figure 2 to generate 512 (instead of 4000) rows of data in module 256 and then these data were given to WEKA. Instead of using an extra file for the test, we used 10 fold cross validation for it (because of limited computer resources) and we also shuffled data by WEKA filters. This gave us the following results.

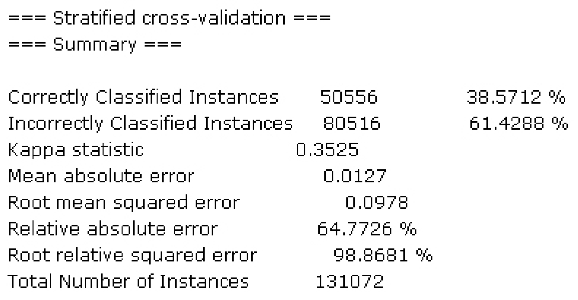
The similar result of Figure 7 as expected is seen and shown in Figure 8. 96 results were found and we showed some of them in Figure 8.

Figure 9 shows that more than 38% of instances were correctly classified. This is a great success in predicting the target values (key).

This result can be used in different ways for different encryption algorithm to find flaws and make them safer.

```
atb0 = 1: 255 (512.0)
atb0 = 2: 1 (512.0)
atb0 = 3: 3 (512.0)
atb0 = 4
|   prgaout = 0: 104 (0.0)
.
|   prgaout = 255: 94 (3.0/2.0)
atb0 = 8: 3 (512.0)
atb0 = 9: 255 (512.0)
atb0 = 10: 1 (512.0)
```

Figure 8: Numbers which did not move during permutations and other correctly calssified instances.



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      50556             38.5712 %
Incorrectly Classified Instances    80516             61.4288 %
Kappa statistic                  0.3525
Mean absolute error                  0.0127
Root mean squared error                 0.0978
Relative absolute error             64.7726 %
Root relative squared error            98.8681 %
Total Number of Instances          131072
```

Figure 9: Accuracy of the test.

CONCLUSIONS AND FUTURE WORKS

Using data mining to solve problems is like using Artificial Intelligence and Statistics to get the best results. As the analysis showed, the key predicted correctly more than 38% for module 64 and more than 38% for module 256. The Decision tree (J48) also found many states in our analysis which are fixed and do not change in RC4. These states can be developed and used to reduce the number of captured packets for cracking WEP in a shorter time.This analysis also proves the previous flaws of RC4 such as what was stated in (Mironov, 2002).

This analysis could be stronger and more accurate by considering the following:
We had very serious hardware restrictions in our analysis such as CPU Power and RAM. Because of these restrictions, we decreased the number of instances. This degradation probably causes some faults in the experiment. To get better results, we suggest increasing the number of instances for module 256 and generating the keys based on the most popular passwords. The authors believe in success of this approach on the other strong streams and other encryption algorithms such as AES, and can disclose more vulnerability of these algorithms. But they suggest that in order to get better results, it is better to join AI or data mining methodology for cryptanalysis with some known attacks or known vulnerabilities.

ACKNOWLEDGEMENTS

Very great thanks to Erik Tews and Andress Klein for their supports. A great thanks to M. A. Kharraz, M. Baghaee, and H. Ghominejad for their editorial helps.

REFERENCES

E. Tews, R. W. and Pyshkin, A. (2007). Breaking 104 bit wep in less than 60 seconds. *Cryptology ePrint*. Available at http://eprint.iacr.org/2007/120.pdf.

Erickson, J. (2008). *Hacking: The Art of Exploitation*. No Starch Publication, San Francisco, CA, 2nd edition.

Finney, H. (1994). An rc4 cycle that can't happen. *sci.crypt newsgroup*.

J. Hakkila, T. Giblin, D. J. H. R. J. R. and Paciesas, W. (2009). J48 decision trees. Retrieved February 4, 2009 from http://grb.mnsu.edu/grbts/doc/manual/J48_Decision_Trees.html.

Kantardzic, M. (2003). *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley & Sons.

Klein, A. (2007). Attacks on the rc4 stream cipher. *Springer Netherlands: Designs, Codes and Cryptography*, 48(3).

Mironov, I. (2002). Not (so) random shuffle of rc4. *Crypto 2002 (M. Yung, ed.)*, 2442 of LNCS:304–319.

S. R. Fluhrer, I. M. and Shamir, A. (2001). Weaknesses in the key scheduling algorithm of rc4. *Eighth Annual Workshop on Selected Areas in Cryptography (SAC)*. Available at http://www.securitytechnet.com/crypto/algorithm/block.html.

Shannon, C. (1984). A mathematical theory of communication. *Bell System Technical*, 27:379–423 and 623–656.

Shannon, C. and Weaver, W. (1949). *The Mathematical Theory of Communication*. IL: University of ILLinois Press.

Witten, I. H. and Frank, E. (2005). *DATA MINING : Practical machine learning tools and techniques*. Morgan Kaufmann series in data management systems, UNITED STATES OF AMERICA, 2nd edition.

Wu, S. L. and Tseng, Y., editors (2007). *Wireless ad hoc networking: personal area, local area, and the sensory area networks*. Auerbach Publications, New York.