Summer 8-10-2018

Fast and Accurate Machine Learning-based Malware Detection via RC4 Ciphertext Analysis

Euiseong Ko

Fast and Accurate Machine Learning-based Malware Detection via RC4 Ciphertext Analysis

A Thesis Presented to

The Faculty of the Computer Science Department

by

Euiseong Ko

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Kennesaw State University

July 2018

Fast and Accurate Machine Learning-based Malware Detection via RC4 Ciphertext Analysis

Approved:

_____

Dr. Donghyun Kim - Advisor

_____

Dr. Dan Chia-Tien Lo– Department Chair

_____

Dr. Jon Preston - Dean

In presenting this thesis as a partial fulfillment of the requirements for an advanced degree from Kennesaw State University, I agree that the university library shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish, this thesis may be granted by the professor under whose direction it was written, or, in his absence, by the dean of the appropriate school when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from or publication of, this thesis which involves potential financial gain will not be allowed without written permission.

_____

Euiseong Ko

Notice To Borrowers

Unpublished theses deposited in the Library of Kennesaw State University must be used only in accordance with the stipulations prescribed by the author in the preceding statement.

The author of this thesis is:

Euiseong
Ko

1100 S Marietta PKWY,
Marietta, GA 30060

The director of this thesis is:

Dr. Donghyun Kim

1100 S Marietta PKWY,
Marietta, GA 30060

Users of this thesis not regularly enrolled as students at Kennesaw State University are required to attest acceptance of the preceding stipulations by signing below. Libraries borrowing this thesis for the use of their patrons are required to see that each user records here the information requested.

Fast and Accurate Machine Learning-based Malware Detection via RC4 Ciphertext Analysis

An Abstract of
A Thesis Presented to

The Faculty of the Computer Science Department

by

Euiseong Ko
Bachelor of Science, Hanyang University, 2014

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Kennesaw State University

July 2018

ABSTRACT

Malware is dramatically increasing its viability while hiding its malicious intent and/or behavior by employing ciphers. So far, many efforts have been made to detect malware and prevent it from damaging users by monitoring network packets. However, conventional detection schemes analyzing network packets directly are hardly applicable to detect the advanced malware that encrypts the communication. Cryptoanalysis of each packet flowing over a network might be one feasible solution for the problem. However, the approach is computationally expensive and lacks accuracy, which is consequently not a practical solution. To tackle these problems, in this paper, we propose novel schemes that can accurately detect malware packets encrypted by RC4 without decryption in a timely manner. First, we discovered that a fixed encryption key generates unique statistical patterns on RC4 ciphertexts. Then, we detect malware packets of RC4 ciphertexts efficiently and accurately by utilizing the discovered statistical patterns of RC4 ciphertext given encryption key. Our proposed schemes directly analyze network packets without decrypting ciphertexts. Moreover, our analysis can be effectively executed with only a very small subset of the network packet. To the best of our knowledge, the unique signature has never been discussed in any previous research. Our intensive experimental results with both simulation data and actual malware show that our proposed schemes are extremely fast (23.06±1.52 milliseconds) and highly accurate (100%) on detecting a DarkComet malware with only a network packet of 36 bytes.

Fast and Accurate Machine Learning-based Malware Detection via RC4 Ciphertext Analysis

A Thesis Presented to

The Faculty of the Computer Science Department

by

Euiseong Ko

In Partial Fulfillment

of Requirements for the Degree

Master of Science, Computer Science

Advisor: Dr. Donghyun Kim

Kennesaw State University

July 2018

ACKNOWLEDGEMENTS

Above all, I would like to thank my advisor, Dr. Donghyun Kim. He gives me an opportunity to study here and support my overall master degree. Second, I would also like to thank Dr. Mingon Kang and Dr. Junggab Son. They make me find the right direction in which this research was stepped. Also, I want to express my appreciation for my lab (SPACL) members and Dr. Youngsoon Kim, who always encourage me to do my work. Next, I thank everyone who supported me throughout my journey. Finally, I really want to express my great gratitude to my family in South Korea.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter I

Introduction

Malware stands for malicious software or a program that is designed to damage or cause unexpected actions on computer systems or networks [1, 2]. Since the first report of malware, both malware and its detection schemes have continuously evolved while mutually affecting each other [3]; once a new malware is found, the malware is analyzed and a new detection scheme is developed to protect systems or networks from the malware. Then, another new malware that can circumvent the detection scheme is revealed, and so on. This is an ongoing conflict between the malware and the detection schemes. Recently, malware seems to enjoy a dominant position by leveraging encryption schemes [4–6]. The use of encryption brought a significant advantage to malware in terms of muddling detection schemes; malware utilizes ciphers to disguise as legal traffics and/or conceal themselves by encrypting either the whole program or just the malicious code. Hence, the problem of how to detect malware exploiting encryption algorithms becomes an urgent issue to retain the security, reliability, and dependability of networks.

Rivest Cipher 4 (RC4) is one of the most popular ciphers exploited by malware to disguise itself as it is highly efficient and simple to implement. Below are two representative cases of malware that uses RC4. It is noteworthy that in both cases, if identifying or distinguishing ciphertexts without decryption is possible, we can develop an efficient and safe detection scheme.

Disguising Communication. DarkComet [7] and NjRat in remote access trojan (RAT), ZEUS [8] and citadel [9, 10] in botnet, and Cryptowall [11] in ransomware are known to utilize RC4. Some of these also adopt secure network communication protocols, e.g., secure socket layer/transport layer security (SSL/TLS) which is used by most modern Internet applications to protect their communications [12–14]. As TLS is

becoming more popular and simplistic, hackers have been increasingly exploiting this security technology to conceal malicious actions and avoid detection. The recent advances in TLS also help malware easily establish a secure communication [15, 16]. According to the report issued by Cisco, about 12% of malware generated malicious traffic through the TLS in 2015, and this number is still growing [13].

Disguising Existence. Code packing is another technique that RC4 mainly utilizes to compress/encrypt program codes or executable files to hide malicious code and/or actions [17–19]. Malware widely utilizes the technique (over 80%), especially encrypting a part of executables, to bypass detection schemes [20, 21]. To check malicious indents of packed executables, they must be unpacked by using a built-in algorithm inside a memory. This process could infect the host directly as the execution of executables is a part of unpacking processes. Some advanced schemes such as Sandbox [22], which unpacks embedded executables inside an isolated environment to avoid infection, have been proposed. However, they still have limitations of that the solutions require high computational costs and resources to establish Sandbox's isolated virtual environment.

Cryptoanalysis of each packet flowing over a network might be one feasible solution for the problem. However, even with state-of-the-art approaches, its inaccuracy and time inefficiency make it impractical. The discrimination of ciphertexts, among normal (non-malware) network packets without decryption, is an alternative approach to deal with the problem. Likewise, identifying the ciphertexts that malware generates while monitoring network flows is an immense challenge due to the difficulty of extracting semantic information from ciphertexts. So far, only a handful of reports discuss this issue in their literatures [23, 24].

Many considerable weaknesses of RC4 have been reported. A flaw (e.g., biased bytes) in the RC4 key-scheduling algorithm was used to recover keys or plaintexts from ciphertexts [25, 26]. However, the recovery requires a complex process of about 2^{13} algorithm operations for 256-bit key [25]. Recently, a new statistical weakness of RC4, biases in RC4 keystreams, has been introduced [27–29]. This weakness could recover RC4 ciphertexts with a success rate of over 50% per byte when 2^{26} TLS sessions are used.

We discovered that RC4 generates statistical patterns on its ciphertexts when being encrypted using a fixed key. To the best of our knowledge, the unique signature has never been discussed before. Moreover, it is highly significant in terms of enabling a detection scheme to inspect packets without decryption. This brings us two major advantages of:

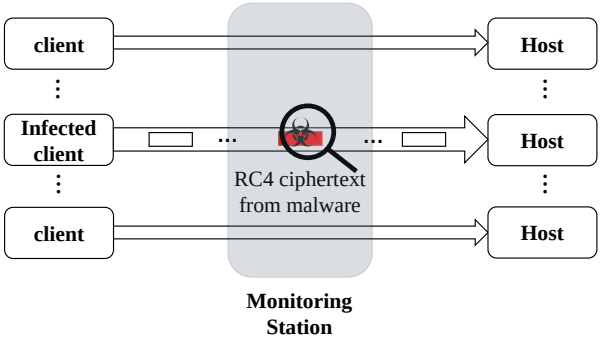


FIGURE 1: Conceptual overview.

(1) achieving a real-time monitoring by eliminating advanced inspection schemes toward encrypted packets and (2) preventing the detection scheme from being infected during inspection processes.

We propose fast and accurate machine learning-based schemes to detect RC4 ciphertexts when the encryption key is known, by incorporating the statistical patterns of RC4 ciphertexts. Figure 1 depicts a conceptual overview of our detection scheme. Suppose there exists a monitoring station on a network. The monitoring station inspects all the incoming and outgoing packets using the proposed schemes and detects malware packets that are encrypted by RC4 and known fixed keys. Note that it is a practical assumption to consider a known fixed key, since it is common that malware uses an embedded key due to the difficulty of key exchange. For instance, DarkComet, Cryptowall, ZEUS, and Lazarus malware family are reported to use embedded keys, which can easily be obtained from their source code or using reverse engineering [7, 8, 11, 30].

Our experimental analyses with DarkComet packets demonstrate that the proposed scheme enables fast and accurate identification of RC4 ciphertexts. Although the proposed schemes can only detect RC4 ciphertext, it will greatly improve the performance of detection schemes compared to others which rely on a set of trivial information such as port numbers, HTTP headers, packets from/to domain name server (DNS), and so on [4, 13]. Therefore, it will contribute to improve the security, reliability, and dependability of networks.

The rest of this paper is organized as follows: Chapter 2 introduces the discovery of the unique signature of RC4 ciphertexts, Chapter 3 proposes RC4 ciphertexts detection schemes using the signature, Chapter 4 evaluates the schemes with real DarkComet

packets, Chapter 5 discusses why the statistical patterns are generated in RC4 ciphertexts, Chapter 6 provides surveys on related work, and finally, Chapter 7 concludes this paper.

Chapter 2

Distinguishability of RC4 ciphertexts

Ciphertexts have been typically considered to generate no distinguishable patterns, since ciphers are developed to provide indistinguishability and semantic security to their ciphertexts. Contrary to these known characteristics, in our research, we found that a fixed encryption key generates unique statistical patterns on RC4 ciphertexts.

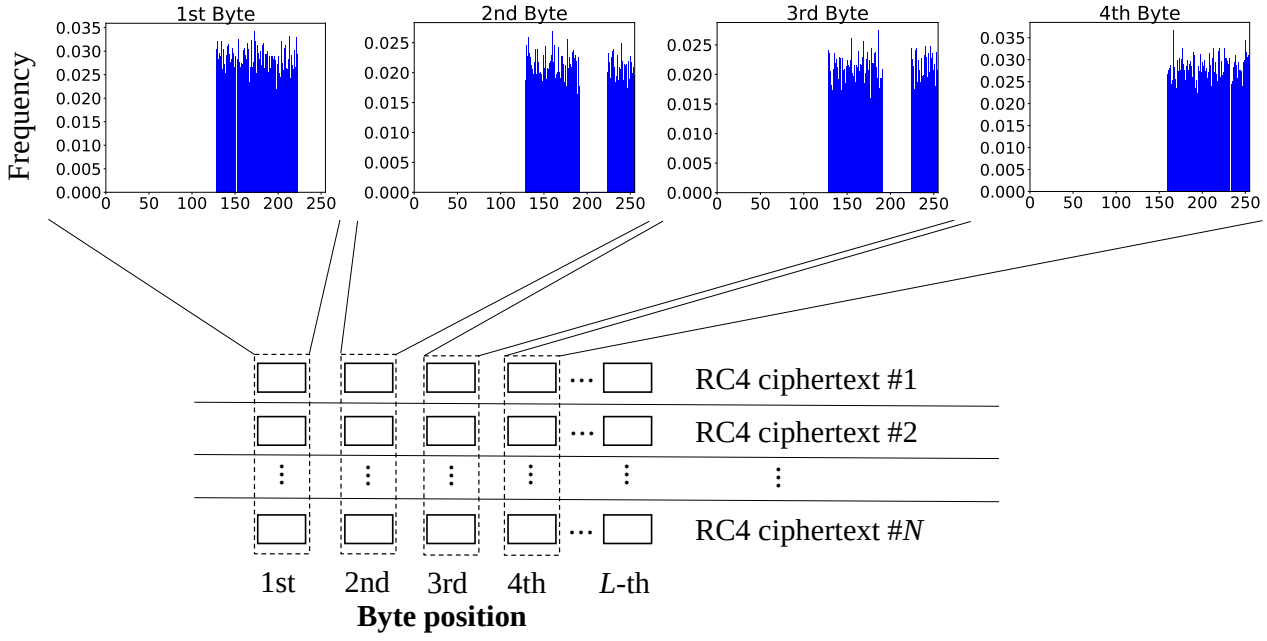


Figure 2: The distributions of the first four bytes in RC4 ciphertexts with LKEY.

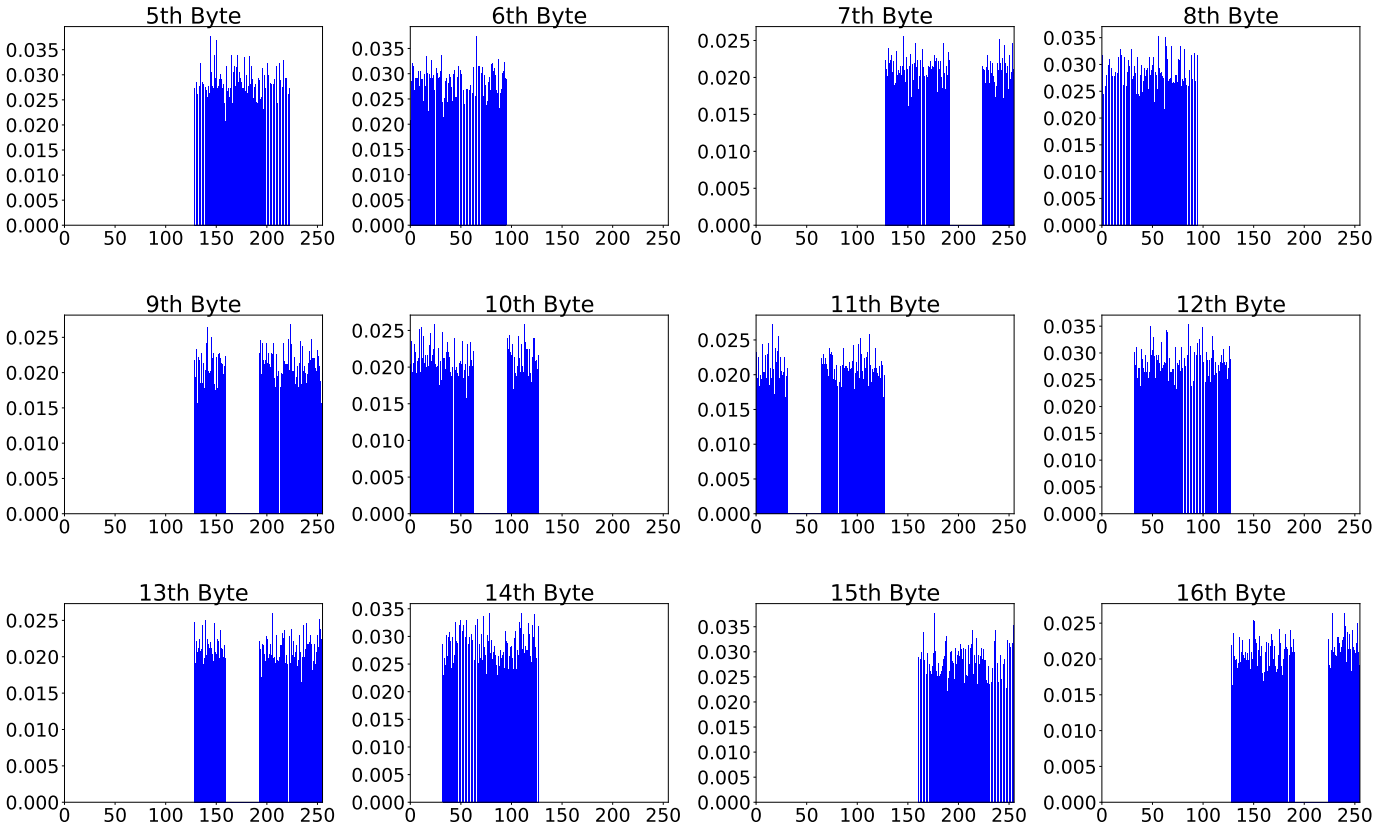


FIGURE 3: The distributions of the 5–16th bytes in RC4 ciphertexts with LKEY.

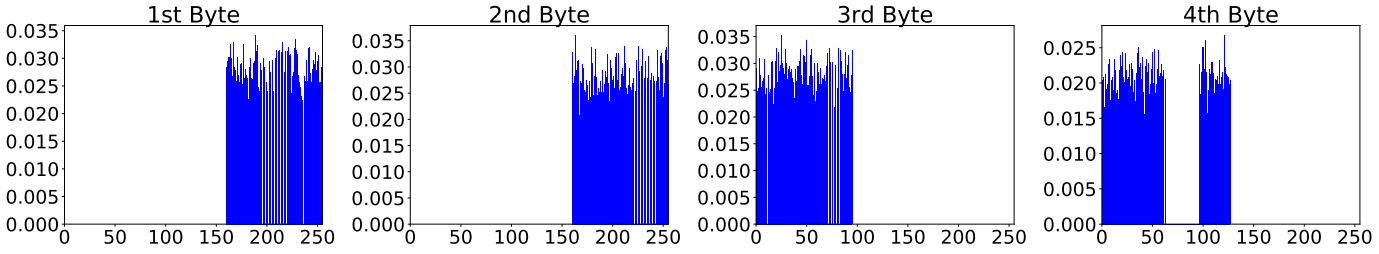


FIGURE 4: The distributions of the first four bytes in RC4 ciphertexts with DKEY.

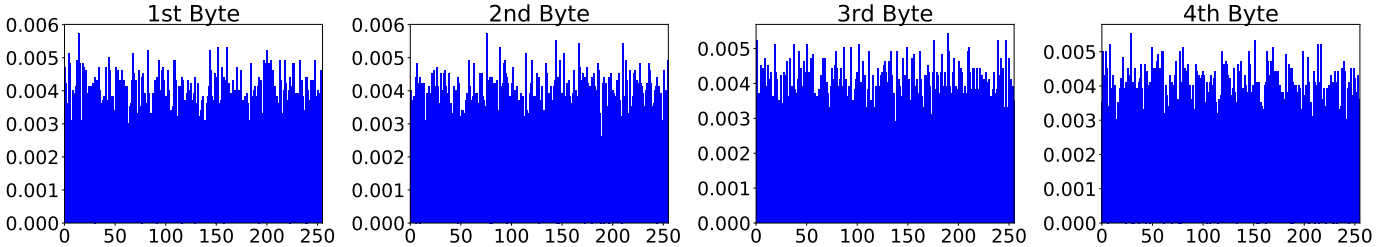


FIGURE 5: The distributions of the first four bytes in RC4 ciphertexts with random keys.

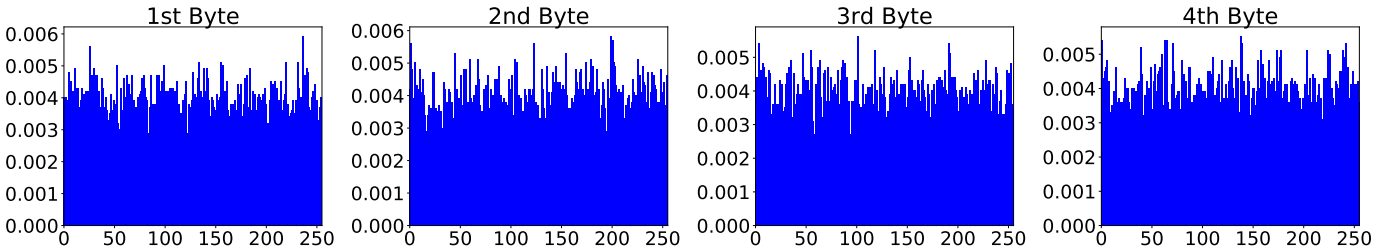


FIGURE 6: The distributions of the first four bytes in AES ciphertexts with LKEY.

In order to examine the unique statistical patterns on RC4 ciphertexts, we created four ciphertext datasets by the following procedures:

- Procedure 1: Generates L bytes of random plaintext, where each byte value is randomly between 32–126 in American Standard Code for Information Interchange (ASCII). The byte value contains decimals (0 to 9), alphabets (a to z and A to Z), and special symbols on the keyboards. Then, repeat to generate N numbers of plaintexts per each dataset.
- Procedure 2: Repeat Procedure 1 to generate four datasets and encrypt them by: (1) RC4 cipher with LKEY, (2) RC4 cipher with DKEY, (3) RC4 cipher with a random key on each plaintext, and (4) AES Cipher Block Chaining (CBC) mode with LKEY for the four datasets (refer to DATA1–DATA4) respectively,
- Procedure 3: Convert each byte of the ciphertexts into a decimal number.

where LKEY ("abcdefghijklmnopqrstuvwxyz012345\0\0\0\0\0") was extracted from Lazarus's collection of malware [30], while DKEY ("#KCMDDC51#-8900123456789") was obtained from DarkComet version 5.3.1 [7]. Note that the ASCII subset (32–126) can express most of the Internet packets, since the packets seldom contain special characters out of the ASCII subset.

The unique signature of RC4 ciphertexts is elucidated in Figure 2. The values on each byte of the RC4 ciphertexts, generated by the same encryption key, are shown only in a certain range of values. For instance, the first byte values of DATA1 (i.e., RC4 ciphertexts with LKEY) are observed only between 125–225 as shown in Figure 2. The similar patterns of the distribution are also shown in the other bytes. Figure 3 shows the distributions of the byte values in the 5–16th byte of the RC4 ciphertexts.

We compared the distributions of the byte values in DATA1 with those of DATA2 (i.e., RC4 ciphertext with DKEY). As shown in Figure 4, DATA2 presented different distribution from those of DATA1. However, the byte values in DATA2 still tends to be biased into certain ranges.

Furthermore, we examined the distribution of the byte values in DATA3 (RC4 ciphertexts with random keys on each plaintext) and DATA4 (AES with LKEY). For DATA3, each plaintext was encrypted by RC4 but with a randomly generated key. In contrast to RC4 ciphertexts with a fixed key, the distributions of the byte values of DATA3 and DATA4 revealed well-distributed across the range of 0 and 255, as illustrated in Figure 5 and Figure 6 respectively.

The observations indicate that RC4 ciphertexts produce unique statistical patterns with a given key. In other words, distinguishable statistical distributions of byte values in RC4 ciphertexts can be predicted if the encryption key is known.

Chapter 3

Machine-Learning based RC4 Ciphertext Analysis schemes

In this paper, we propose machine learning-based schemes that can detect RC4 ciphertexts, when the encryption key is known. We consider the following two settings:

- When an entire RC4 ciphertext is available
- When only a subset of RC4 ciphertext is available.

3.1 Notation

We investigated a sequence of network packets in the captured traffic generated during a communication session. A network packet is denoted by $\mathbf{x} = \{x_i | 0 \leq x_i \leq 255, 1 \leq i \leq L\}$. We consider a network packet of length L bytes, where each byte, x_i, is represented by an integer between 0 and 255. An encryption algorithm is denoted by $\mathbf{E}_k$, where $\mathbf{E}$ and the subscript k indicate the cipher and an encryption key respectively. For instance, $\mathbf{RC4}_{LKEY}$ and $\mathbf{AES}_{LKEY}$ indicate RC4 and AES encryptions with LKEY respectively.

Let $P(x_i)$ be a probability that a byte value x_i is observed in the i-th position ($1 \leq i \leq L$) of the network packet $\mathbf{x}$. A conditional probability that a network packet, $\mathbf{x}$, is a ciphertext encrypted by $\mathbf{E}_k$ is denoted by $P(\mathbf{x}|\mathbf{E}_k)$. For instance, $P(x_2|\mathbf{RC4}_{\mathrm{LKEY}})$ represents a chance that the byte value x_2 would be observed in the second byte of the ciphertext encrypted by $\mathbf{RC4}_{LKEY}$. Assuming that the bytes of the ciphertext are conditionally independent, the conditional probability can be computed by $P(\mathbf{x}|\mathbf{E}_k) = \prod_{i=1}^{L} P(x_i|\mathbf{E}_k)$.

3.2 Detection of RC4 Ciphertexts with a Known Key

First, we consider a classification problem that detects a ciphertext encrypted by $\mathrm{RC4}_k$, assuming that an entire network packet is available for the analysis when monitoring the network. To tackle the problem, we developed a machine learning-based approach. Let $P(\mathbf{RC4}_k|\mathbf{x})$ be a posterior probability that represents a network packet encrypted by $\mathbf{RC4}_k$. Thus, a RC4 ciphertext can be detected by the discriminant function:

$$P(\mathbf{RC4}_k|\mathbf{x}) \underset{\neg\mathbf{RC4}_k}{\overset{\mathbf{RC4}_k}{\gtrless}} \theta, \tag{3.1}$$

where θ is a threshold, and $\neg\mathbf{RC4}_k$ shows that the network packet is not encrypted by $\mathrm{RC4}_k$. If the posterior probability is greater than the threshold (i.e., $P(\mathbf{RC4}_k|\mathbf{x}) > \theta$), the network packet is classified as a ciphertext encrypted by $\mathbf{RC4}_k$, which indicates a malware packet.

The posterior probability can be estimated by Bayes' theorem:

$$P(\mathbf{RC4}_k|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{RC4}_k)P(\mathbf{RC4}_k)}{P(\mathbf{x})}. \tag{3.2}$$

Since $P(\mathbf{RC4}_k)$ and $P(\mathbf{x})$ are constants in the discriminant function in (3.1), the posterior probability is propositional to:

$$P(\mathbf{RC4}_k|\mathbf{x}) \propto P(\mathbf{x}|\mathbf{RC4}_k). \tag{3.3}$$

The conditional probability, $P(\mathbf{x}|\mathbf{RC4}_k)$, can be computed by:

$$P(\mathbf{x}|\mathbf{RC4}_k) = \prod_{i=1}^{L} P(x_i|\mathbf{RC4}_k), \tag{3.4}$$

where $P(x_i|\mathbf{RC4}_k)$ is a probability that a byte value x_i is observed in the i-th position ($1 \leq i \leq L$) of the ciphertext encrypted by $\mathbf{RC4}_k$. Then, we take the log-likelihood function for efficient computation:

$$\ln P(\mathbf{x}|\mathbf{RC4}_k) = \sum_{i=1}^{L} \ln P(x_i|\mathbf{RC4}_k). \tag{3.5}$$

Therefore, the final discriminant function, $\mathcal{D}(\mathbf{x}, k)$, is constructed for the RC4 ciphertext

Algorithm 1: Detection of RC4 ciphertexts with an encryption key k

1. Training model:
Generate N random ASCII texts and encrypt them by $\mathrm{RC4}_k$,
$X = \{\{x_{11}, \ldots, x_{1L}\}, \ldots, \{x_{N1}, \ldots, x_{NL}\}\}$.
Compute a posterior probability on each byte:
for $i = 1$ *to* L **do**
  $P(x_i|\mathbf{RC4}_k) = \frac{c_i(x_i)+\alpha}{\sum_{m=0}^{255} c_i(m)+\alpha}$
end
2. Detection:
input : $\mathbf{x} = \{x_1, x_2, \ldots, x_L\}$
output: $\mathbf{RC4}_k$ or $\neg\mathbf{RC4}_k$
begin
  for $i = 1$ *to* L **do**
    $p = p + \ln P(x_i|\mathbf{RC4}_k)$
  end
  if $(p > \theta)$ **then**
    return $\mathbf{RC4}_k$
  else
    return $\neg\mathbf{RC4}_k$
  end
end

detection as:

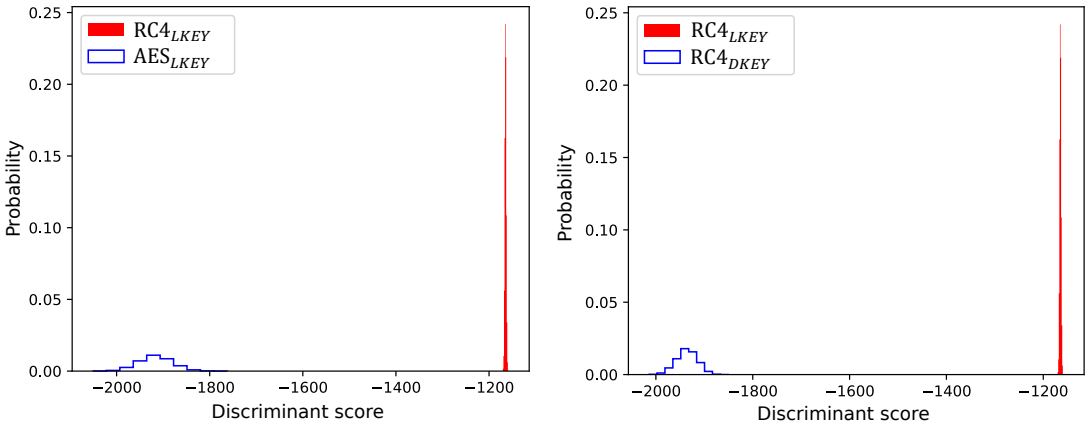
$$\mathcal{D}(\mathbf{x}, k) = \sum_{i=1}^{L} \ln P(x_i|\mathbf{RC4}_k) \underset{\neg\mathbf{RC4}_k}{\overset{\mathbf{RC4}_k}{\gtrless}} \theta. \tag{3.6}$$

When the available training data is insufficient, estimating accurate parameters in Bayesian approaches is challenging. However, the conditional probability on each byte $P(x_i|\mathbf{RC4}_k)$ can be efficiently approximated by synthetic ciphertexts in this study. A large number of synthetic ciphertexts can be generated with random ASCII texts encrypted by $\mathbf{RC4}_k$. Then, the conditional probability can be empirically estimated by the occurrence of the synthetic ciphertexts on each byte:

$$P(x_i|\mathbf{RC4}_k) = \frac{c_i(x_i) + \alpha}{\sum_{m=0}^{255} c_i(m) + \alpha}, \tag{3.7}$$

where $c_i(x_i)$ is the number of the occurrences of the byte value x_i $(0 \leq x_i \leq 255)$ in the i-th position of the ciphertexts, and α is a pseudo-count to prevent a zero probability. The denominator normalizes the occurrences of the byte values. The details of the scheme are described in Algorithm 1.

We carried out two simulation experiments that detect RC4 ciphertexts with a fixed key from (a) AES ciphertexts with the same key and (b) RC4 ciphertexts with a different



((A)) ((B))

FIGURE 7: Comparison of the distributions of the discriminant scores. (a) $\mathcal{D}(\boldsymbol{X}_{RC4}, \text{LKEY})$ vs. $\mathcal{D}(\boldsymbol{X}_{AES}, \text{LKEY})$ and (b) $\mathcal{D}(\boldsymbol{X}_{RC4}, \text{LKEY})$ vs. $\mathcal{D}(\boldsymbol{X}_{RC4}, \text{DKEY})$

key in order to assess the performance.

In the first experiment, we generated 30,000 random text messages where each one includes ASCII data of 256 bytes ($L = 256$). Additionally, the messages were evenly divided into three sets (10,000 each). The first two sets were encrypted by RC4 with LKEY, and the last set was encrypted by AES with LKEY which is the same key with RC4's. We used the first set (denoted by $\mathbf{T}_{RC4}$) as the training data for building the detection model and used both the second and third sets for testing (denoted by $\boldsymbol{X}_{RC4}$ and $\boldsymbol{X}_{AES}$ respectively). The model parameters $P(x_i|\mathbf{RC4}_k)$ were estimated by Eq. (3.7) with the training data $\mathbf{T}_{RC4}$. We set the pseudo-count to one ($\alpha = 1$). Then, we evaluated the performance of the proposed scheme with the two datasets, $\boldsymbol{X}_{RC4}$ and $\boldsymbol{X}_{AES}$. The scores of the discriminant functions, $\mathcal{D}(\boldsymbol{X}_{RC4}, \text{LKEY})$ and $\mathcal{D}(\boldsymbol{X}_{AES}, \text{LKEY})$, with the test data are illustrated in Figure 3.7(a), where $\mathcal{D}(\boldsymbol{X}_{RC4}, \text{LKEY})$ and $\mathcal{D}(\boldsymbol{X}_{AES}, \text{LKEY})$ are colored red and blue respectively. Interestingly, the figure shows that the distributions of the discriminant scores are perfectly discriminative, which consequently could detect the RC4 ciphertexts with 100% accuracy against the AES ciphertexts when $\theta =$ -1,200.

In the second experiment, we considered a similar experimental setting as the first experiment but encrypted the third data set by RC4 with DKEY. Thus, we evaluated whether or not the proposed scheme can detect a RC4 ciphertext of the known key from the ciphertexts encrypted by the same encryption algorithm but with the different encryption key. The distribution of the discriminant scores, $\mathcal{D}(\boldsymbol{X}_{RC4}, \text{LKEY})$ and

Algorithm 2: Detection of a partial RC4 ciphertext

1. Training model:
Use the training model in Algorithm 1
2. Detection:
input : $\mathbf{s} = \{s_1, s_2, \ldots, s_M\}$
output: $\mathbf{RC4}_k$ or $\neg\mathbf{RC4}_k$
begin
  for $i = 1$ *to* $(L - M + 1)$ **do**
    p_i=0
    for $j = 1$ *to* M **do**
      $p_i = p_i + \ln P(x_{i+j-1} = s_j|\mathbf{RC4}_k)$
    end
    p=argmax p_i
    if $(p > \zeta)$ **then**
      return $\mathbf{RC4}_k$
    end
  end
  return $\neg\mathbf{RC4}_k$
end

$\mathcal{D}(\boldsymbol{X}_{RC4}, \text{DKEY})$, are shown in Figure 3.7(b). Similarly, the proposed scheme perfectly detected the RC4 ciphertexts with LKEY from the RC4 ciphertexts with DKEY, where the cutoff threshold was also set to -1,200. We have repeated the experiments multiple times with various settings of different keys, and they have consistently shown 100% accuracy in detecting RC4 ciphertexts of a known key.

3.3 Detection of RC4 Ciphertexts When a Network Packet is Partially Available

We also examined RC4 ciphertext detection when the network packet was somehow partially available during network monitoring. We demonstrated that our proposed scheme can detect RC4 ciphertexts accurately when the positions of the ciphertexts and their encryption keys are known in the previous section. However, it is often difficult to know the exact position of a ciphertext in the network, and some network protocols (e.g., UDP) may have data loss. Since the observed patterns of RC4 ciphertexts are shown as a sequence, our proposed scheme may not function if we have only partial packets of RC4 ciphertexts or if some packets are missing.

Therefore, we extended the proposed scheme to detect a RC4 ciphertext even if a network packet is partially available or some bytes are missing. Suppose that we analyze a subset of a network packet, $\mathbf{s} = \{s_j | 1 \leq j \leq M\}$ and $\mathbf{s} \subset \mathbf{x}$, where the size of the

partial packet is much shorter than that of regular RC4 ciphertexts, i.e., $M \ll L$. Specifically, we aim to determine whether or not $\mathbf{s}$ is a part of ciphertexts encrypted by $\mathbf{RC4}_k$.

Since a given partial packet is a subset of the original RC4 ciphertext that the malware generated, we infer the most probable position of the original RC4 ciphertext, in which the partial packet was located. We define a likelihood function, $P(i|\mathbf{s}, \mathbf{RC4}_k)$, which shows how likely the partial packet $\mathbf{s}$ is a subset of the RC4 ciphertext starting at i $(1 \leq i \leq L)$. The log-likelihood, $\ln P(i|\mathbf{s}, \mathbf{RC4}_k)$, can be computed by:

$$\ln P(i|\mathbf{s}, \mathbf{RC4}_k) = \sum_{j=1}^{M} \ln P(x_{i+j-1} = s_j|\mathbf{RC4}_k), \tag{3.8}$$

where $P(x_{i+j-1} = s_j|\mathbf{RC4}_k)$ represents the probability that the byte value s_j is observed in the $(i + j - 1)$-th of the RC4 ciphertext $\mathbf{x}$. Thus, $P(x_{i+j-1} = s_j|\mathbf{RC4}_k)$ can be estimated by Eq. (3.7).

Then, the maximum likelihood function shows the most probable position where the statistical patterns of the partial packet are matched. The most probable position of the partial packet in the original RC4 ciphertext can be obtained by:

$$i^* = \underset{i}{\operatorname{argmax}} \ln P(i|\mathbf{s}, \mathbf{RC4}_k), \tag{3.9}$$

where $1 \leq i \leq L - M + 1$. An example of the log-likelihood of a partial packet is illustrated in Figure 8. The partial packet of 18 bytes was extracted from a RC4 ciphertext of 256 bytes encrypted with LKEY, where the partial packet data was located in between 219 to 236 of the original RC4 ciphertext. Then, the likelihood scores in Eq. (3.8) were computed to find the position of the partial packet in the original RC4 ciphertext (shown in Figure 8). In the figure, the distribution of the log-likelihood shows the highest score (around -82) at 219, which means that the partial packet is the subset of the RC4 ciphertext starting at the position.

In order to determine whether or not the partial packet $\mathbf{s}$ is a subset of a RC4 ciphertext, the discriminant function, $\mathcal{L}(\mathbf{s}, k)$, is defined by a log-posterior probability,

$\ln P(\mathbf{RC4}_k|\mathbf{s})$, which can be estimated by the log-likelihood:

$$\begin{aligned}\mathcal{L}(\mathbf{s},k) &= \ln P(\mathbf{RC4}_k|\mathbf{s}) \\ &\propto \ln P(i^*|\mathbf{s},\mathbf{RC4}_k) \\ &= \sum_{j=1}^{M} \ln P(x_{i^*+j-1}=s_j|\mathbf{RC4}_k). \end{aligned} \tag{3.10}$$

Finally, RC4 ciphertext can be detected by comparing the discriminant function with a threshold (ζ):

$$\mathcal{L}(\mathbf{s},k) = \sum_{j=1}^{M} \ln P(x_{i^*+j-1}=s_j|\mathbf{RC4}_k) \underset{\neg\mathbf{RC4}_k}{\overset{\mathbf{RC4}_k}{\gtrless}} \zeta. \tag{3.11}$$

The details of the scheme are described in Algorithm 2.

We empirically estimated the optimal threshold (ζ^*) and the minimum size (M) of the partial packet for RC4 ciphertext detection. We compared the distributions of log-likelihood scores of the two groups with synthetic data. The first group (denoted by G_C) includes only the log-likelihood scores computed in the correct position, and the second (denoted by G_I) contains the scores in the other positions. A cut-off value that discriminates the two distributions is considered the optimal threshold. In this study, the optimal threshold is simply determined by the middle point of the range, $[\max(G_I), \min(G_C)]$. Specifically, we generated 10,000 RC4 ciphertexts with LKEY and selected a partial packet of length M in a random position on each ciphertext. We considered various lengths of the partial packets: $M \in \{16, 18, 20, 26, 32, 34, 36, 40\}$. In Figure 9, the distributions of the log-likelihood scores in the two groups are depicted with partial packets of different lengths, where the distribution of G_C is shown in a solid line (blue), while that of G_I is in a dash-dot line (red). The empirically optimal thresholds on the various lengths of partial packets are listed in Table 4. When the length of partial packets (M) is 16 bytes, the two distributions are overlapped between -72.61 and -71.48, which causes misclassification (Figure 3.9(a) and Table 4). However, the experiments show that the overlap decreases as the length of partial packets increase, because longer partial packets provide more information about the statistical patterns of RC4 ciphertexts. Interestingly, when examining partial packets of longer than 20 bytes, the distributions of the two groups are distinctly separated, which means partial packets of RC4 ciphertext can be detected 100% if the partial packet is longer than 20 bytes.

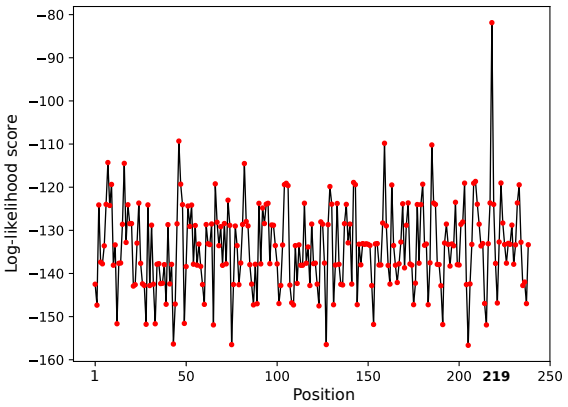


FIGURE 8: Log-likelihood of a partial packet across a network packet.

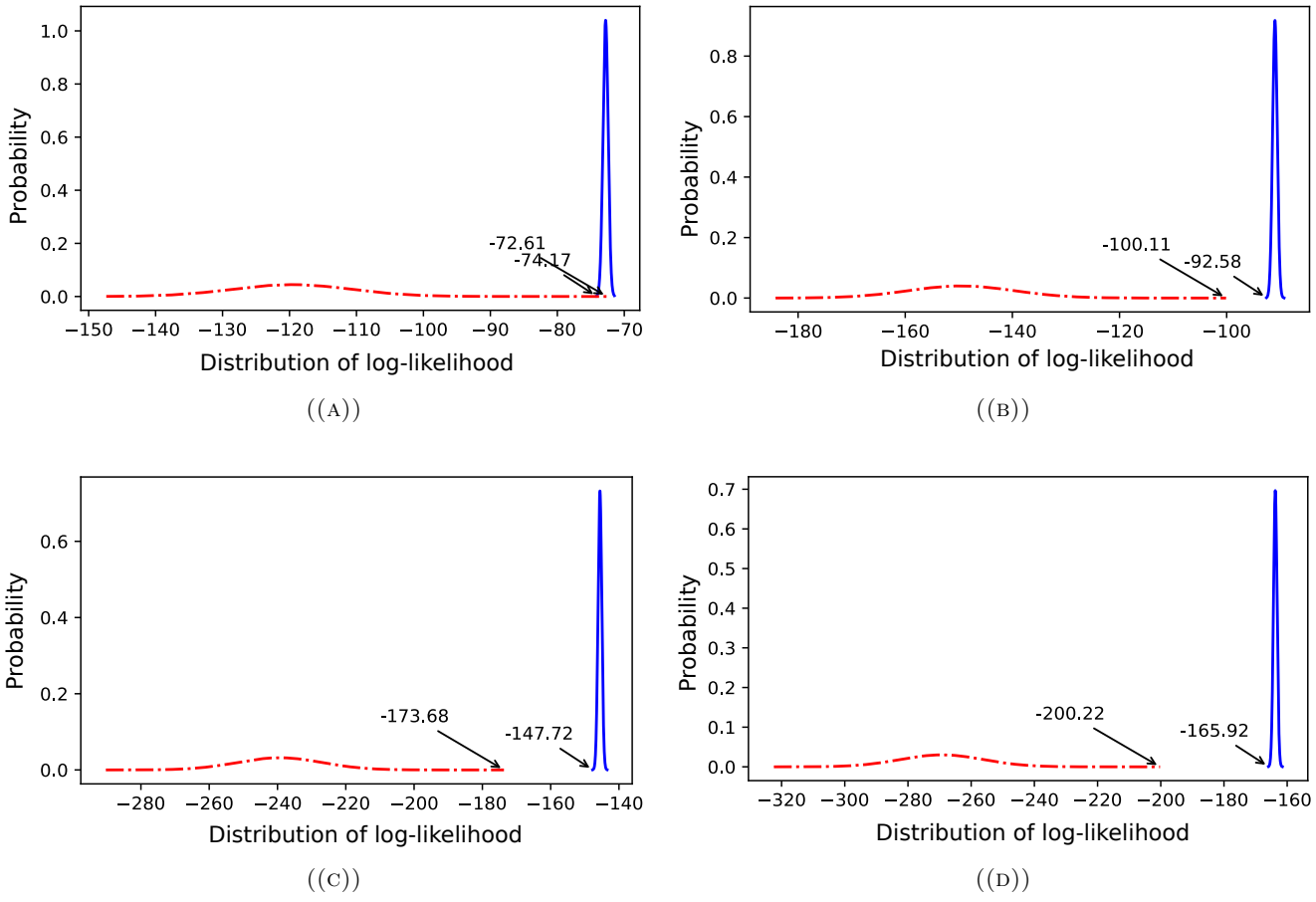


FIGURE 9: The distributions of log-likelihood in G_C and G_I with various lengths: (a) N=16, (b) N=20, (c) N=32, and (d) N=36.

TABLE 1: The distributions of log-likelihood in G_C and G_I with various lengths.

		G_C	G_I	Threshold value
16 bytes	Min	-74.17	-147.36	-73.39
	Max	-71.48	-72.61	
	Avg	-72.78	-119.65	
18 bytes	Min	-83.48	-165.78	-84.72
	Max	-80.41	-85.97	
	Avg	-81.88	-134.61	
20 bytes	Min	-92.57	-184.20	-96.34
	Max	-89.28	-100.11	
	Avg	-90.98	-149.55	
26 bytes	Min	-120.25	-239.46	-130.66
	Max	-116.57	-141.06	
	Avg	-118.28	-194.44	
32 bytes	Min	-147.72	-290.26	-160.70
	Max	-143.46	-173.68	
	Avg	-145.57	-239.37	
34 bytes	Min	-156.83	-304.27	-171.74
	Max	-152.51	-186.65	
	Avg	-154.66	-254.32	
36 bytes	Min	-165.92	-322.43	-183.07
	Max	-161.51	-200.22	
	Avg	-163.77	-269.31	
40 bytes	Min	-184.09	-354.63	-206.18
	Max	-179.57	-228.27	
	Avg	-181.97	-299.27	

Chapter 4

Detection of RC4 ciphertexts from Malware

We evaluated our schemes with real malware as well as synthetic ciphertext data. For the assessment, we used DarkComet Remote Administration Tool (RAT) version 5.3.1 [7]. DarkComet has many functions, such as keylogger, webcam capture, and remote chat. Moreover, DarkComet securely transfers data using TCP communication with RC4 cipher to avoid being detected. There have been a number of reports that DarkComet uses a fixed encryption key (i.e., LKEY [31, 32]), which is embedded in the software for the secure communications.

We analyzed network packets that DarkComet transfers to a victim computer via *Remote Chat* of DarkComet. We captured the TCP network packets using Wireshark [33]. The packets are illustrated in Figure 10. The data in red shows a header, while payloads are in blue. We considered only the payloads for testing. We generated 100 test sets by executing DarkComet 100 times. On each execution, we attempted to detect RC4 ciphertexts examining partial packets of various sizes ($N \in \{16, 18, 20, 26, 32, 34, 36, 40\}$). The experimental results are shown in Table 5, where it shows (1) lowest, highest, and average of the distribution of the log-likelihood scores of the packets; and (2) the detection accuracy by our scheme. The results show that RC4 ciphertexts can be detected with 100% accuracy by monitoring partial packets of longer than 36 bytes (see Table 5). In this experiment, we considered the optimal thresholds obtained in Table 4.

We also examined the computational cost of the proposed scheme in the DarkComet experiments. Table 6 shows the execution times of the proposed scheme that detect RC4 ciphertexts that DarkComet generates. The proposed scheme detected RC4 ciphertexts

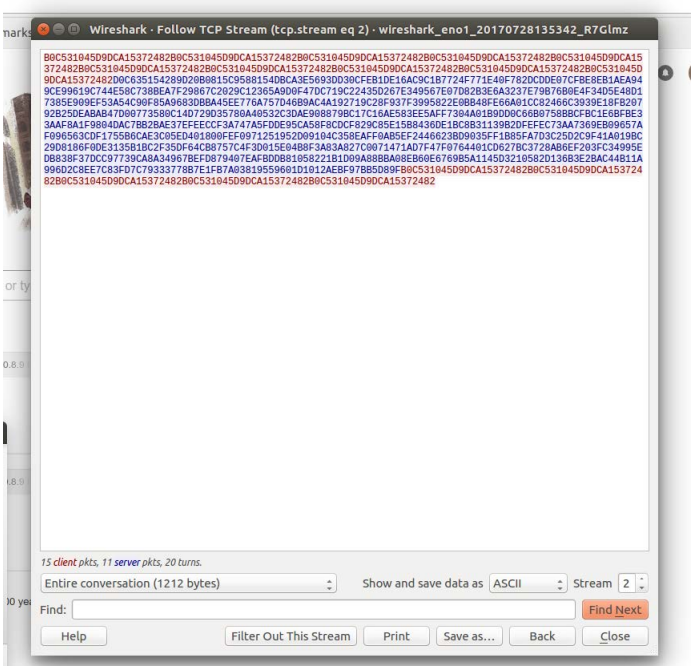


FIGURE 10: The network packets that DarkComet generates.

TABLE 2: Accuracy with DarkComet packets.

Length	Min	Max	Avg	# of samples	Detected	Undected	Accuracy
16	-82.19	-72.15	-73.46	100	74	26	74 %
18	-101.67	-81.08	-83.88	100	86	14	86 %
20	-100.50	-90.38	-92.15	100	90	10	90 %
26	-137.33	-117.40	-121.91	100	96	4	96 %
32	-165.28	-145.37	-148.85	100	98	2	98 %
34	-173.81	-153.72	-158.28	100	99	1	99 %
36	-182.97	-163.35	-167.89	100	100	0	100 %
40	-202.24	-181.17	-186.28	100	100	0	100 %

in 23.06±1.52 millisecond with 100% accuracy, when analyzing partial packets of 36 bytes.

DarkComet required at least 36 bytes for the 100% detection accuracy, which is longer than the experimental results in the simulation study (100% accuracy with 20 byte packets in Figure 9). This discrepancy between them may be caused by special characters in the plaintext, which are out of the ASCII subset. We evaluated our scheme only with DarkComet due to the lack of available malware and difficulty to establish test environments.

The experiments were implemented on an Intel core (TM) i5 processor running at 1.6

TABLE 3: Execution times.

# of Lengths (bytes)	Execution Time (msec)
16	15.62 ± 0.48
18	15.62 ± 0.48
20	15.86 ± 1.40
26	18.06 ± 1.31
32	21.77 ± 1.64
34	22.27 ± 1.26
36	23.06 ± 1.52
40	25.99 ± 2.01

GHz speed, 4.00 GB of RAM, and an SSD Serial ATA 3.0 Gbit/s drive with an 8 MB buffer. All algorithms in this paper were implemented using Python (ver. 2.7.13) in a 64-bit operating system.

Chapter 5

Discussion

We discuss why RC4 ciphertexts produce statistical patterns. To describe the signature, we denote the encryption algorithm of RC4 cipher as $c = \mathbf{RC4}_k\{m\}$, where m is a plaintext, k is a key, and c is a ciphertext as an output of the encryption. Given the two inputs k and m, a RC4 cipher creates a key stream k_s. Then, a RC4 ciphertext (c) is generated by computing $k_s \oplus m$, where $\oplus$ is exclusive OR (XOR). Intuitively, the key scheduling algorithm of RC4 always generates the same key stream k_s if the key k is the same. Thus, the range of c is determined by the range of m. For instance, if we create three RC4 ciphertexts using three characters ('a', 'b', and 'c') and a fixed key k, the RC4 cipher would produce only three characters (c_1, c_2, and c_3) for ciphertexts. In other words, 'a' is only mapped to c_1, and c_1 is only mapped to 'a' with a fixed key. Similarly, 95 characters (32–126 in ASCII) that we used in the experiments were mapped to only 95 values in the RC4 ciphertexts.

Chapter 6

Related Works

A number of weaknesses in RC4 have been discussed. Traces that represent correlation between an input key and keystream were found in the keystream when a small set of encryption keys (weak keys) were used [34]. Therefore, an attacker can easily recover the key by following the traces from the internal state of the Key Scheduling Algorithm (KSA) or the output stream [34]. It was reported that different keys often generate similar output keystreams (called key collisions) [35]. A new approach was developed to creates colliding key pairs [36]. The reversibility of the Pseudo Random Generation Algorithm (PRGA) is exploited to recover an internal state from any given state [37]. The internal state can also be exploited to recover secret keys. However, these weaknesses cannot be utilized for the packet monitoring, since the approach is applicable only if the internal state of KSA is accessible.

Biased bytes are one significant weakness of RC4, which interrelate to secret keys and/or internal states. Examples of the biased bytes are: (1) the first byte of an output of KSA is correlated to the first three bytes of an input key [38], (2) the second byte of the RC4 ciphertext is biased toward zero with probability of 1/128 (generally 1/256) [39], and (3) the first two bytes of the RC4 ciphertext are also biased in a different circumstance [40]. Later, short-term biases and long-term biases are defined where the former biases do not appear on the further rounds of KSA [41–43], while the latter biases are remained in the key stream even after removing the initial bytes [44–46]. Recently, a bias on the 128th byte of the permutation after KSA was discussed, and the bias are also found in RC4A and Variably Modified Permutation Composition (VMPC), which are the variants of RC4 [29]. A research showed that the biases are interrelated with the length of the secret key [47], and the biases cannot be removed even after the key length is simply

increased [48].

These biases can be utilized: (1) to recover an encryption key [49], (2) to distinguish the keystream of RC4 from a random stream [43], and (3) to recover keys and/or plaintexts [25, 26]. However, the recovery requires a complex process of about 2^{13} algorithm operations for 256-bit key [25]. AlFardan et al. introduced ciphertext-only plaintext recovery attacks against RC4-based TLS [27]. Their attack is based on short term biases on a single byte in RC4 keystreams.

General attacks have been also performed to cryptanalyzeRC4. Xue et al. proposed a GB-RC4 algorithm for brute force attacks on RC4 [50]. Their attack was performed using GPU to improve the performance. Nevertheless, it took 12.8 hours to search the whole key space of 40-bit input. Aviram et al. used a server supporting SSLv2 as a random Oracle and proposed a cross-protocol attack on TLS [51].

While there exist diverse research results in RC4 cryptanalysis, only a few researches have approached in detection of malware leveraging ciphers. Wressnegger et al. proposed a probable-plaintext attacks to deobfuscate enbedded malware in documents [52]. Specifically, their scheme could efficiently decrypt a ciphertext encrypted by Vigenere cipher, XOR, ADD, and ROL instructions. If the length of used key is less then 13 bytes, their scheme can decrypt it within a second. Anderson et al. proposed a new scheme that can identify encrypted malware traffic based on contextual flow data [4]. Specifically, they used featured data, such as TLS handshake metadata, domain name server (DNS) contextual flows, and HTTP headers for a supervised learning. They also extracted features of TLS by analyzing 18 malicious families and 5,623 samples. The extracted features include flow metadata, sequence of packet lengths and times, byte distribution, and unencrypted header information [53]. These features can be effectively used as machine learning classifiers. However, these approaches rely on a set of trivial information, and thus some limitations exist: (1) a sufficient amount of data must be collected, (2) detection is not highly accurate, and (3) malware will be able to easily bypass the detection scheme by simply tweaking its behaviors.

Furthermore, some research results have been introduced to especially deal with Botnets that employs ciphers. Zhang et al. proposed high-entropy classifiers to detect encrypted botnet traffic [54]. They used the characteristics that the ciphertext has higher entropy than its plaintext. However, entropy is not sufficient to be a discriminant feature for the ciphertext level detection. Rossow et al. identified a special type of Botnet which

uses encryption in their Command-and-Control (C&C) protocols [55]. After that, they proposed a probabilistic model that automatically infers a syntax of the C&C protocol. However, this approach can detect only a specific type of Botnet, which exhibits characteristic payload strings. The most recently, Carli et al. proposed an end-to-end system that can automatically discover an encryption scheme and a key used to encrypt C&C traffics [56]. However, it needs a pair of encrypted/decrypted network traffic, and thus their scheme has to perform a dynamic analysis with Sandbox, which makes real-time analysis impossible.

Chapter 7

Conclusion

The detection of encrypted malware packets is extremely challenging but essential for retaining the security, reliability, and dependability of networks. In this paper, we develop novel machine learning-based malware detection schemes to identify malware and/or malware packets encrypted by RC4 when the encryption is known. The schemes detect RC4 malware-based the unique signature of RC4 ciphertexts that we discovered. We found that RC4 ciphertexts encrypted with a fixed key generate unique statistical patterns. To our best knowledge, the unique signature of RC4 has never been reported before, although some weaknesses of RC4 cipher have been often discussed.

In the intensive simulation studies, the proposed schemes accurately identified RC4 ciphertexts with 100% accuracy. To demonstrate the efficiency and effectiveness of the proposed schemes, we performed the experiments using real malware packets. We used DarkComet version 5.3.1 for the assessment. The real malware packets of DarkComet were detected with 90% of accuracy within 15.86±1.40 milliseconds by our schemes, when the input network packet was of 20 byte length. Furthermore, the detection accuracy reached 100% when the length of the input network packet was longer than or equal to 36 bytes. The execution times to identify 36 bytes packet were only 23.06±1.52 milliseconds.

The proposed schemes are only applicable to detect RC4 ciphertexts of malware, and it is assumed that the encryption key is already known. However, a number of malware programs are still using simple ciphers such as RC4 for both efficient encryption and decryption, and the encryption key is often embedded in the program due to the difficulty of key exchange.

References

[1] M. Damshenas, A. Dehghantanha, K.-K. R. Choo, and R. Mahmud, "M0droid: An android behavioral-based malware detection model," *Journal of Information Privacy and Security*, vol. 11, no. 3, pp. 141–157, September 2015.

[2] Z. Zhu and T. Dumitras, "Featuresmith: Automatically engineering features for malware detection by mining the security literature," in *CCS'16 Conference Proceedings*. ACM SIGSAC, October 2016, pp. 767–778.

[3] M. F. A. Razak, N. B. Anuar, R. Salleh, and A. Firdaus, "The rise of malware: Bibliometric analysis of malware study," *Journal of Network and Computer Applications*, vol. 75, pp. 58–76, November 2016.

[4] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *AISec'16 Conference Proceedings*. ACM, October 2016, pp. 35–46.

[5] X. Jiang, X. Wang, and D. Xu, "Stealthy malware detection and monitoring through vmm-based out-of-the-box semantic view reconstruction," *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 2, pp. 12:1–28, February 2010.

[6] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *SPSM'11 Conference Proceedings*. ACM, October 2011, pp. 3–14.

[7] (2012, July) Darkcomet reomte administration tool. [Online]. Available: http://www.darkcomet-rat.com/

[8] C. Park, H. Park, and K. Kim, "Realtime c&c zeus packet detection based on rc4 decryption of packet length field," *Advanced Science and Technology Letters*, vol. 64, pp. 55–59, 2014.

[9] J. Milletary, "Citadel trojan malware analysis," in *Dell SecureWorks Counter Threat Unit Intelligence Services*. Dell, September 2012, pp. 10–18.

[10] A. Rahimian, R. Zirati, S. Preda, and M. Debbabi, "On the reverse engineering of the citadel botnet," in *FPS'13 Conference Proceedings*. Springer International Publishing, October 2013, pp. 408–425.

[11] (2014, July) Cryptowall ransomware built with rc4 bricks. [Online]. Available: https://securingtomorrow.mcafee.com/mcafee-labs/cryptowall-ransomware-built-with-rc4-bricks

[12] "Finding hidden threats by decrypting ssl," *SANS Institute InfoSec Reading Room*, pp. 1–17, November 2013.

[13] B. Anderson, *Hiding in Plain Sight: Malware's Use of TLS and Encryption*. http://blogs.cisco.com/security/malwares-use-of-tls-and-encryption: Cisco Blogs: Security, January 2016.

[14] R. Ernst, *Encrypted Traffic Management*. Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE in Wachtberg and Bonn, January 2016.

[15] C. Brubaker, S. Jana, B. Ray, S. Khurshid, and V. Shmatikov, "Using frankencerts for automated adversairal testing of certificate valication in ssl/tls implementations," in *S&P'14 Conference Proceedings*. IEEE, November 2014, pp. 114–129.

[16] L. S. Huang, A. Rice, E. Ellingsen, and C. Jackson, "Analyzing forged ssl certificates in the wild," in *S&P'14 Conference Proceedings*. IEEE, November 2014, pp. 83–97.

[17] K. Pandey, *Adobe Flash Zero-Day Vulnerability-Operation Clandestine Wolf by FireEye*. SISA-Blog: Official Bolog site of SISA, June 2015. [Online]. Available: http://blog.sisainfosec.com/2015/06/adobe-flash-zero-day-vulnerability.html

[18] P. Porras, H. Saidi, and V. Yegneswaran, *An Analysis of Conficker's logic and Rendezvous Points*. SRI International Technical Reoprt, March 2009.

[19] A. Trivedi, *Future Malware Analysis*, July 2017. [Online]. Available: https://www.linkedin.com/pulse/future-malware-analysis-arjun-trivedi

[20] M. Bat-Erdene, H. Park, H. Li, H. Lee, and M.-S. Choi, "Entropy analysis to classify unknown packing algorithms for malware detection," *International Journal of Information Security*, vol. 16, no. 3, pp. 227–248, June 2017.

[21] C. Burgess, F. Kurugollu, S. Sezer, and K. McLaughlin, "Detecting packed executables using steganalysis," in *EUVIP'14 Conference Proceedings*, Dec 2014, pp. 1–5.

[22] L. Martignoni, M. Christodorescu, and S. Jha, "Omniunpack: Fast, generic, and safe unpacking of malware," in *ACSAC'07 Conference Proceedings*, December 2007, pp. 431–441.

[23] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *BWCCA'10 Conference Proceedings*. ACM, October 2011, pp. 297–300.

[24] P. O'Kane, S. Sezer, and K. McLaughlin, "Obfuscation: The hidden malware," *IEEE Security and Privacy*, vol. 9, no. 5, pp. 41–47, August 2011.

[25] M. A. Orumiehchiha, J. Pieprzyk, E. Shakour, and R. Steinfeld, "Cryptanalysis of rc4(n, m) stream cipher," in *SIN'13 Conference Proceedings*. ACM, 2013, pp. 165–172.

[26] S. R. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of rc4," in *SAC'01 Conference Proceedings*. Springer-Verlag, 2001, pp. 1–24.

[27] N. J. Alfardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. Schuldt, "On the security of rc4 in tls," in *USENIX Security Symposium Proceedings*. USENIX, August 2013, pp. 305–320.

[28] S. Sarkar, S. S. Gupta, G. Paul, and S. Maitra, "Providing tls-attack related open biases of rc4," *Designs, Codes, and Cryptography*, vol. 77, no. 1, pp. 231–253, August 2014.

[29] S. Sarkar, "Further non-randomness in rc4, rc4a, and vmpc," *Cryptography and Communications*, vol. 7, no. 3, pp. 317–330, December 2014.

[30] *Operation Blockbuster: Unraveling the Long Thread of the Sony Attack*. NOVETTA, February 2016.

[31] "Looking at the skey for a darkcomet," in *FIDELIS Cybersecurity*, August 2015, pp. 1–41.

[32] "Darkcomet analysis-understanding the trojan used in syrian uprising," in *Reverse Engineering*. INFOSEC Institute, March 2012.

[33] (2012, Version 2.2.8) Wireshark network protocol analyzer. [Online]. Available: https://www.wireshark.org/

[34] G. Paul, S. Rathi, and S. Maitra, "On non-negligible bias of the first output byte of rc4 towards the first three bytes of the secret key," *Des. Codes Cryptography*, vol. 49, no. 1-3, pp. 123–134, Dec. 2008.

[35] A. L. Grosul and D. S. Wallach. (2000, June) Variably modified permutation composition. [Online]. Available: https://scholarship.rice.edu/handle/1911/96275

[36] S. Maitra, G. Paul, S. Sarkar, M. Lehmann, and W. Meier, "New results on generalization of roos-type biases and related keystreams of rc4," in *AFRICACRYPT'13 Conference Proceedings*, 2013, pp. 222–239.

[37] G. Paul and S. Maitra, "Permutation after rc4 key scheduling reveals the secret key," in *SAC'14 Conference Proceedings*, 2007, pp. 360–377.

[38] A. L. Grosul and D. S. Wallach. (1995, September) A class of weak keys in the rc4 stream cipher. [Online]. Available: http://www.impic.org/papers/WeakKeys-report.pdf

[39] I. Mantin and A. Shamir, "A practical attack on broadcast rc4," in *FSE'01 Conference Proceedings*, ser. FSE'01, 2002, pp. 152–164.

[40] S. Paul and B. Preneel, "Analysis of non-fortuitous predictive states of the rc4 keystream generator," in *INDOCRYPT'03 Conference Proceedings*, 2003, pp. 52–67.

[41] T. Isobe, T. Ohigashi, Y. Watanabe, and M. Morii, "Full plaintext recovery attack on broadcast rc4," in *FSE'13 Conference Proceedings*, 2013, pp. 179–202.

[42] J. D. Golic, "Linear statistical weakness of alleged rc4 keystream generator," in *EUROCRYPT'97 Conference Proceedings*, 1997, pp. 226–238.

[43] S. R. Fluhrer and D. A. McGrew, "Statistical analysis of the alleged rc4 keystream generator," in *FSE'00 Conference Proceecdings*, 2001, pp. 19–30.

[44] S. Maitra, G. Paul, and S. S. Gupta, "Attack on broadcast rc4 revisited," in *FSE'11 Conference Proceedings*, 2011, pp. 199–217.

[45] I. Mantin, "Predicting and distinguishing attacks on rc4 keystream generator," in *EUROCRYPT'05 Conference Proceedings*, 2005, pp. 491–506.

[46] R. Basu, S. Ganguly, S. Maitra, and G. Paul, "A complete characterization of the evolution of rc4 pseudo random generation algorithm," *Journal of Mathematical Cryptology*, vol. 2, no. 3, pp. 257–289, October 2008.

[47] S. S. Gupta, S. Maitra, G. Paul, and S. Sarkar, "(non-)random sequences from (non-)random permutations-analysis of rc4 stream cipher," *Journal of Cryptology*, vol. 27, no. 1, pp. 67–108, January 2014.

[48] C. Liu, Y. Cai, and T. Wang, "Security evaluation of rc4 using big data analytics," in *ICSESS'16 Conference Proceedings*. IEEE, August 2016, pp. 316–320.

[49] G. Paul and S. Maitra, "Permutation after rc4 key scheduling reveals the secret key," in *SAC'07 Conference Proceedings*, 2007, pp. 360–377.

[50] P. Xue, T. Li, H. Dong, C. Liu, W. Ma, and S. Pei, "Gb-rc4: Effcetive brute force attacks on rc4 algorithm using gpu," in *IGSCO'16 Conference Proceedings*. IEEE, November 2016, pp. 1–6.

[51] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni, E. Ksper, S. Cohney, S. Engels, C. Paar, and Y. Shavitt1, "Drown: Breaking tls using sslv2," in *USENIX Security Symposium Proceedings*. USENIX, August 2016, pp. 1–18.

[52] C. Wressnegger, F. Boldewin, and K. Rieck, "Deobfuscating embedded malware using probable-plaintext attacks," in *RAID'13 Conference Proceedings*, 2013, pp. 164–183.

[53] B. Anderson, S. Paul, and D. McGrew, "Deciphering malware's use of tls (without decryption)," in *arXiv:1607.01639 [cs.CR]*. Cornell University Library, July 2016, pp. 1–15.

[54] H. Zhang, C. Papadopoulos, and D. Massey, "Detecting encrypted botnet traffic," in *Computer Communications Workshops Proceedings*. IEEE, April 2013, pp. 163–168.

[55] C. Rossow and C. J. Dietrich, "Provex: Detecting botnets with encrypted command and control channels," in *DIMVA'13 Conference Proceedings*, 2013, pp. 21–40.

[56] L. D. Carli, R. Torres, G. Modelo-Howard, A. Tongaonkar, and S. Jha, "Botnet protocol inference in the presence of encrypted traffic," in *INFOCOM'17 Conference Proceedings*, 2017, pp. 1–9.