Smashing WEP in A Passive Attack

Pouyan Sepehrdad[1], Petr Sušil[2] ⋆, Serge Vaudenay[2], and Martin Vuagnoux

[1] Intel CRI-SC at TU-Darmstadt, Darmstadt, Germany
[2] EPFL, Lausanne, Switzerland
pouyan.sepehrdad@trust.cased.de,
{petr.susil, serge.vaudenay}@epfl.ch,
martin@vuagnoux.com

Abstract. In this paper, we report extremely fast and optimised active and passive attacks against the old IEEE 802.11 wireless communication protocol WEP. This was achieved through a huge amount of theoretical and experimental analysis (capturing WiFi packets), refinement and optimisation of all the former known attacks and methodologies against RC4 stream cipher in WEP mode. We support all our claims by providing an implementation of this attack as a publicly available patch on Aircrack-ng. Our new attacks improve its success probability drastically. We adapt our theoretical analysis in Eurocrypt 2011 to real-world scenarios and we perform a slight adjustment to match the empirical observations. Our active attack, based on ARP injection, requires 22 500 packets to gain success probability of 50% against a 104-bit WEP key, using Aircrack-ng in non-interactive mode. It runs in less than 5 seconds on an off-the-shelf PC. Using the same number of packets, Aicrack-ng yields around 3% success rate. Furthermore, we describe very fast passive only attacks by just eavesdropping TCP/IPv4 packets in a WiFi communication. Our passive attack requires 27 500 packets. This is *much less than the number of packets* Aircrack-ng requires in *active mode* (around 37 500), which is a huge improvement. We believe that our analysis brings on further insight to the security of RC4.

1 Introduction

RC4 was designed by Rivest in 1987. It used to be a trade secret until it was anonymously posted on Cypherpunks mailing list in September 1994. Nowadays, due to its simplicity, RC4 is widely used in SSL/TLS, Microsoft Lotus, Oracle Secure SQL and Wi-Fi 802.11 wireless communications. The 802.11 [9] used to be protected by WEP (Wired Equivalent Privacy) which is now being replaced by WPA (Wi-Fi Protected Access) due to security weaknesses.

WEP uses RC4 with a pre-shared key. Each packet is encrypted by an XOR to a keystream generated by RC4. The RC4 key is a pre-shared key prepended with a 3-byte nonce initialisation vector IV. The IV is sent in clear for self-synchronisation. There have been several attempts to break the full RC4 algorithm, but it has only been devastating so far in this scenario. Indeed, the

⋆ Supported by a grant of the Swiss National Science Foundation, 200021_134860/1.

adversary knows that the key is constant except the IV, which is known. An active adversary can alter the IV. Nowadays, WEP is considered as being terribly weak, since passive attacks can recover the full key easily by assuming that the first bytes of every plaintext frame are known.

Structure of the paper. First, in Section 2, we refer to the motivation in this research area, then we present RC4, WEP and Aircrack-ng in Section 3. In Section 4, we go through all the existing well-known attacks on WEP. Next, we introduce some useful lemmas in Section 5. Then, we present all known biases for RC4 in Section 6. Subsequently, we elaborate on an optimised attack on WEP in Section 7 and, we compare our results with Aircrack-ng 1.1 in Section 8. Finally, we discuss some challenges and open problems in Section 9.

2 Motivation

For some people, attacking WEP is like beating a dead horse, but this horse is still running wildly in many countries all over the world. Also, some companies are selling hardware using modified versions of the WEP protocol, they claim to be secure [2]. Moreover, the new analysis and biases presented in this paper are related to RC4, which is the most popular stream cipher in the history of symmetric key cryptography. WEP is an example of a practical exploitation of these biases. The cryptanalysis of WEP is one of the most applied cryptographic attacks in practice. Indeed, tools such as Aircrack-ng are massively downloaded to provide a good example of weaknesses in cryptography. Finally, the TKIP protocol used by WPA is not much different from WEP (just a patch over WEP), so that attacks on WEP can affect the security of networks using TKIP, as seen in [2,26]. For instance in [26], the authors used exactly the same biases as in WEP to break WPA. Hence, gaining a better understanding of the behaviour of these biases may lead to a practical breach of WPA security in future.

3 Preliminaries

3.1 Description of RC4 and Notations

The RC4 stream cipher consists of two algorithms: the Key Scheduling Algorithm (KSA) and the Pseudo Random Generator Algorithm (PRGA). The RC4 engine has a state defined by two registers (words) i and j and one array (of N words) S defining a permutation over $\mathbf{Z}/N\mathbf{Z}$. The KSA generates an initial state for the PRGA from a random key K of L words as described in Fig. 1. It starts with an array $\{0, 1, \ldots, N-1\}$, where $N = 2^8$ and swaps N pairs, depending on the value of the secret key K. At the end, we obtain the initial state S'_0.

We define all the operators such as addition, subtraction and multiplication in the ring of integers modulo N represented as $\mathbf{Z}/N\mathbf{Z}$, or $\mathbf{Z}_N$, where $N = 256$ (i.e. *words* are *bytes*). Thus, $x + y$ should be read as $(x + y) \bmod N$.

KSA

```
for $i = 0$ to $N - 1$ do
    $S[i] \leftarrow i$
end for
$j \leftarrow 0$
for $i = 0$ to $N - 1$ do
    $j \leftarrow j + S[i] + K[i \bmod L]$
    swap($S[i]$,$S[j]$)
end for
```

PRGA

```
$i \leftarrow 0$
$j \leftarrow 0$
loop
    $i \leftarrow i + 1$
    $j \leftarrow j + S[i]$
    swap($S[i]$,$S[j]$)
    output $z_i = S[S[i] + S[j]]$
end loop
```

Fig. 1. The KSA and the PRGA algorithms of RC4.

Once the initial state S'_0 is created, it is used by the second algorithm of RC4, the PRGA. Its role is to generate a keystream of words of $\log_2 N$ bits, which will be XORed with the plaintext to obtain the ciphertext. Thus, RC4 computes the loop of the PRGA each time a new keystream word z_i is needed, according to the algorithm in Fig. 1. Note that each time a word of the keystream is generated, the internal state (i, j, S) of RC4 is updated.

Sometimes, we consider an idealised version RC4$^\star(t)$ of RC4 defined by a parameter t as shown in Fig. 2. Namely, after round t, index j is assigned randomly. This model has been already used in the literature such as in [17,22,20]. In fact, t is the index of the last known state. For instance, since we know $K[0], K[1], K[2]$ in WEP protocol, we can initially assume $t = 2$.

KSA$^\star(t)$

```
for $i = 0$ to $N - 1$ do
    $S[i] \leftarrow i$
end for
$j \leftarrow 0$
for $i = 0$ to $N - 1$ do
    if $i \leq t$ then
        $j \leftarrow j + S[i] + K[i \bmod L]$
    else
        $j \leftarrow$ random
    end if
    swap($S[i]$,$S[j]$)
end for
```

PRGA*

```
$i \leftarrow 0$
$j \leftarrow 0$
loop
    $i \leftarrow i + 1$
    $j \leftarrow$ random
    swap($S[i]$,$S[j]$)
    output $z_i = S[S[i] + S[j]]$
end loop
```

Fig. 2. The KSA$^\star(t)$ and the PRGA* algorithms of RC4$^\star(t)$.

Let $S_i[k]$ (resp. $S'_i[k]$) denote the value of the permutation defined by the array S at index k, after the round i in the KSA (resp. the PRGA). We also denote $S_{N-1} = S'_0$. Let j_i (resp. j'_i) be the value of j after round i of the KSA (resp. the PRGA), where the rounds are indexed with respect to i. Thus, the KSA has rounds $0, 1, \ldots, N-1$ and the PRGA has rounds $1, 2, \ldots$. The KSA and the PRGA are defined by

$$
\begin{array}{rl|rl}
\multicolumn{2}{c}{\mathsf{KSA}} & \multicolumn{2}{c}{\mathsf{PRGA}} \\
j_{-1} &= 0 & j'_0 &= 0 \\
j_i &= j_{i-1} + S_{i-1}[i] + K[i \bmod L] & j'_i &= j'_{i-1} + S'_{i-1}[i] \\
S_{-1}[k] &= k & S'_0[k] &= S_{N-1}[k] \\
S_i[k] &= \begin{cases} S_{i-1}[j_i] & \mathsf{if}\ k = i \\ S_{i-1}[i] & \mathsf{if}\ k = j_i \\ S_{i-1}[k] & \mathsf{otherwise} \end{cases} & S'_i[k] &= \begin{cases} S'_{i-1}[j'_i] & \mathsf{if}\ k = i \\ S'_{i-1}[i] & \mathsf{if}\ k = j'_i \\ S'_{i-1}[k] & \mathsf{otherwise} \end{cases} \\
 & & z_i &= S'_i[S'_i[i] + S'_i[j'_i]]
\end{array}
$$

Throughout this paper, we denote $\bar{K}[i] \overset{\mathsf{def}}{=} K[0] + \cdots + K[i]$. We let z denote the keystream derived from K using RC4. In the applications we are concerned, the first bytes of a plaintext frame are often known (see Fig. 6 in Appendix), as well as the IV, the first 3 bytes of K. That is, we assume that the adversary can use the keystream z and the IV in a known plaintext attack.

We let I_0 be a set of integers, which represents the key byte indices which are already known. We define a set clue which consists of all $\bar{K}$ bytes whose indices are in I_0. To begin with, we have $I_0 = \{0, 1, 2\}$ and $\mathsf{clue} = \mathsf{IV}$. Given a set of indices I_0 and an index i, we assume that we have a list $\mathsf{row}^{\mathsf{RC4}}_{i|I_0}$ of $d_{i|I_0}$ vectors $(\bar{f}_j, \bar{g}_j, p_j, q_j)$, $j = 1, \ldots, d_{i|I_0}$ with functions $\bar{f}_j$ and the corresponding predicates $\bar{g}_j$ such that

$$\mathsf{Pr}\left[\bar{K}[i] = \bar{f}_j(z, \mathsf{clue}) | \bar{g}_j(z, \mathsf{clue})\right] = p_j$$

for some probability $p_j \neq \frac{1}{N}$ and

$$\mathsf{Pr}\left[\bar{g}_j(z, \mathsf{clue})\right] = q_j$$

where q_j is called the *density* of the bias (for the list of such correlations, see Table 1 in Appendix).

For simplicity, we assume that for some given i, z, and clue, all suggested $\bar{f}_j(z, \mathsf{clue})$ for j's such that $\bar{g}_j(z, \mathsf{clue})$ holds, are pairwise distinct. We further assume that the events $\bar{K}[i] = \bar{f}_j(z, \mathsf{clue})$ with different i's are independent. We will also assume that $\bar{f}_j$ and $\bar{g}_j$ are of the form $\bar{f}_j(z, \mathsf{clue}) = f_j(h(z, \mathsf{clue}))$ and $\bar{g}_j(z, \mathsf{clue}) = g_j(h(z, \mathsf{clue}))$, where $\mu = h(z, \mathsf{clue})$ lies in a domain of size N_μ. In fact, h is just a function compressing the data to the minimum necessary to compute $\bar{f}_j$ and $\bar{g}_j$. The following prominent relation exists between the key bytes of RC4:

$$\bar{K}[i + 16j] = \bar{K}[i] + j\bar{K}[15] \tag{1}$$

for $0 \leq i \leq 15$ and $j = 0, 1, 2$. This relation reveals that if $\bar{K}[15]$ is known, the biases for $\bar{K}[i]$ and $\bar{K}[i + 16j]$ can be merged. This relation was initially used in [34] to derive a better success probability. For example, if we know $\bar{K}[15]$, we can use the biases for $\bar{K}[19]$ to vote for $\bar{K}[3]$. Similarly, the biases for $\bar{K}[15], \ldots, \bar{K}[18]$ and $\bar{K}[31], \bar{K}[32]$ can be merged to vote for $\bar{K}[15]$. Consequently, later, we recover $\bar{K}[15]$ before any other byte of the key.

Definition 1. *Let A, B and C be three random variables over $\mathbf{Z}_N$. We say that A is biased towards B with bias p conditioned on an event E and we represent it as $A \underset{E}{\overset{p}{=}} B$ if*

$$\mathsf{Pr}(A - B = x | E) = \begin{cases} p & \text{if } x = 0 \\ \frac{1-p}{N-1} & \text{otherwise} \end{cases}$$

When $\mathsf{Pr}[E] = 1$, it is denoted as $A \overset{p}{=} B$.

3.2 Description of WEP

WEP [8] uses a 3-byte IV concatenated to a secret key of 40 or 104 bits (5 or 13 bytes) as an RC4 key. Thus, the RC4 key size is either 64 or 128 bits. Since the RC4 design does not accept an IV by default, WEP generates a per packet key for each packet. A devastating problem of WEP is that the 13 bytes of the key do not change for each packet encryption, while the first 3 bytes of the key are changing. Thus, the attacker can run a statistical attack on the key. This was avoided in WPA. In this paper, we do not consider the 40-bit key variant, but a very similar approach can be leveraged to break the 40-bit key version. So, $L = 16$. In fact, we have

$$K = K[0] \| K[1] \| K[2] \| K[3] \| \cdots \| K[15] = \mathsf{IV}_0 \| \mathsf{IV}_1 \| \mathsf{IV}_2 \| K[3] \| \cdots \| K[15]$$

where IV_i represents the $(i+1)$-th byte of the IV and $K[3] \| ... \| K[15]$ represents the fixed secret part of the key. In theory, the value of the IV should be random but in practice, it is a counter, mostly in little-endian and it is incremented by one each time a new 802.11b frame is encrypted. Sometimes, some particular values of the IV are skipped to thwart the specific attacks based on "weak IV's". Thus, each packet uses a slightly different key.

To protect the integrity of the data, a 32-bit long $\mathsf{CRC32}$ check sum called ICV is appended to the data. Similar to other stream ciphers, the resulting stream is XORed with the RC4 keystream and it is sent through the communication channel together with the IV in clear. On the receiver's end, the ciphertext is again XORed with the shared key and the plaintext is recovered. The receiver checks the linear error correcting code and it either accepts the data or declines it.

It is well known [21,31,34] that a relevant portion of the plaintext is practically constant and that some other bytes can be predicted. They correspond to the LLC header and the SNAP header and some bytes of the TCP/IPv4 and ARP encapsulated frames. For example, by XORing the first byte of the ciphertext with the constant value 0xAA, we obtain the first byte of the keystream. Thus, even if these attacks are called known plaintext attacks, they are ciphertext only in practice (see the Appendix for the structure of ARP and TCP/IPv4 packets).

We consider both passive and active adversaries in this paper. For an active attack, the attacker eavesdrops the ARP packets and since the plaintext bytes are known up to the 32-nd byte, she can compute $z_1, \ldots, z_{32}$ values using the ciphertext. It is also possible to inject data into the network. Because the ARP replies expire quickly (resetting the ARP cache), it usually takes only a few seconds or minutes until an attacker can capture an ARP request and start re-injecting it [31]. On the other hand, active attacks are detectable by Intrusion Detection systems (IDS) and also some network cards require extra driver patches to be able to inject data into the traffic, which is not available for all network cards. This is not the case for a passive attack. The attacker can eavesdrop the wireless communication channel for TCP/IPv4 packets, but some of the data frames are not known in this case (see the Appendix). As represented in Table 1, the Klein and the Maitra-Paul attacks require z_i and z_{i+1} to recover $\bar{K}[i]$ respectively. Hence in reality, we are not able to use those attacks to recover some bytes of the key. This is not the case for the Korek attacks, since they only require z_1 and z_2. To summarise, we need more packets in a passive attack compared to an active attack. We are going to elaborate on both types of attacks later.

3.3 Aircrack-ng

Aircrack-ng [5] is a WEP and WPA-PSK keys cracking program that can recover keys once enough data packets have been captured. It is the most widely downloaded cracking software in the world. It implements the standard Fluhrer, Mantin and Shamir's (FMS) attack [7] along with some optimisations like the Korek attacks [13,14], as well as the Physkin, Tews and Weinmann (PTW) attack [31]. In fact, it currently has the implementation of state of the art attacks on WEP and WPA. We applied a patch on Aircrack-ng 1.1 in our implementation.

4 State of the Art Attacks on WEP

WEP key recovery process is harder in practice than in theory. Indeed, some bytes of the keystream are unknown, depending on which type of packets are captured. Moreover, theoretical success probability has often been miscalculated and conditions to recover the secret key are not the same depending on the paper. For example, [31,34,2,25] check 2×10^6 most probable keys instead of the first one as in [7,14,13,11,27,28]. Additionally, IEEE 802.11 standard does not specify how the IV's should be chosen. Thus, some attacks consider randomly picked

IV's or incremental IV's (both little-endian and big-endian encoded). Some implementations specifically avoid some classes of IV's which are weak with respect to some attacks.

To unify the results, we consider recovering a random 104-bit long secret key with random IV's. This corresponds to the default IV behaviour of the 802.11 GNU/Linux stack. We compare the previous and the new results using both theoretical and practical analysis:

- In [7], Fluhrer, Mantin and Shamir's (FMS) attack is only theoretically described. The authors postulate that 4 million packets would be sufficient to recover the secret key of WEP with success probability of 50% with incremental IV's. A practical implementation of this attack has been realised by Stubblefield, Ioannidis and Rubin [27,28]. They showed that indeed between 5 million to 6 million packets are required to recover the secret key using the FMS attack. Note that in 2001, almost all wireless cards were using incremental IV's in big-endian.
- There is no theoretical analysis of the Korek [13,14] key recovery attacks. Only practical implementations such as Aircrack-ng [5] are available. Additionally, Aircrack-ng classifies the most probable secret keys and does a brute-force attack on this list. The success probability of 50% is obtained when about 100 000 packets are captured with random IV's. Note that the amount of the brute-forced keys depends on the values of the secret key and the *"Fudge"* factor [5] (the highest vote counter is divided by the Fudge factor and all values with votes higher than this value is brute-forced), a parameter chosen by the attacker (often 1, 2 or 3). By default, around one thousand to one million keys are brute-forced.
- The ChopChop attack was introduced in [12,30], which allows an attacker to interactively decrypt the last m bytes of an encrypted packet by sending $128 \times m$ packets in average to the network. The attack does not reveal the main key and is not based on any special property of the RC4 stream cipher.
- In [11], Klein showed theoretically that his new attack needs about 25 000 packets with random IV's to recover the secret key with probability 50%. Note that, there is no practical implementation of the Klein attack alone, but both PTW [31] and VV07 [34] attacks (using Klein attack by default), which theoretically improve the key recovery process, need more than 25 000 packets. So, the theoretical success probability of the Klein attack was over estimated. We implemented this attack and we obtained the success probability of 50% with about 60 000 packets (random IV's).
- Physkin, Tews and Weinmann (PTW) showed in [31] that the secret key can be recovered with only 40 000 packets for the same success probability (random IV's). However, this attack brute-forces the 2×10^6 most probable secret keys. Thus, the comparison with previous attacks is less obvious. Moreover, there is no theoretical analysis of this attack, only practical results are provided by the authors. We confirmed this practical result.

- Vaudenay and Vuagnoux [34] showed an improved attack, where the same success probability can be reached with an average of 32 700 packets with random IV's. This attack also tests the 2×10^6 most probable secret keys. Moreover, only practical results are provided by the authors. We confirmed this practical result.

- According to [2], Beck and Tews re-implemented the [34] attack in 2009, obtaining the same success probability with only 24 200 packets using Aircrack-ng in "interactive mode", i.e., the success probability is fixed in this approach and the goal is to derive the least average number of packets for a successful attack. Obviously, this approach requires less packets than the case where we fix the number of packets and compute the success rate. We focus on the latter approach, since this is done often in the literature as a measure of comparison. Since Beck and Tews's attack was implemented on Aircrack-ng, we ran it in non-interactive mode. We observed that 24 200 packets brings about only less that 8% success rate in non-interactive mode. In fact, it needs more than 36 000 packets to yield the success probability of 50%. Therefore, it seems this attack does not yield any more success rate than the [34] attack.

- Sepehrdad, Vaudenay and Vuagnoux [25], showed that only 9 800 packets is enough to break WEP with success probability of 50%, while they used a *class of weak* IV's for their attack. We show in the following that reaching 9 800 packets to break WEP with *random* IV's is extremely ambitious by the currently available biases for RC4.

- In Eurocrypt 2011 [26], we presented an attack on WEP by optimising all the previous known attacks in the literature and by introducing a few new correlations. As a result, we claimed *theoretically* that using 4 000 packets, our analysis provides a success probability of 50% to break WEP. We did not implement the attack at that time. Only theoretical results were presented. In this paper, we show that some parts of that evaluation is not precise enough and need modification. In fact, we show that our theory needs more than 4 000 packets, due to the imprecise approximation of the variance of the rank of the correct key and an improper estimation of the probability distribution of this random variable.

- In this paper, in an optimised attack, we drop the number of packets to 22 500 for the same success probability by modifying the [26] attack and patching Aircrack-ng in non-interactive mode. It requires only 19 800 packets using Aircrack-ng in interactive mode. In our approach, the 2×10^6 most probable secret keys are brute-forced and we use random IV's.

We are going to construct a precise theory behind the WEP attack in the subsequent sections. All our analysis has been checked precisely through extensive amount of experiments. We show that we can recover a 104-bit long WEP key using 22 500 packets in less than 5 seconds using an off-the-shelf PC. With less number of packets, the attack will run for a longer period.

5 Some Useful Lemmas

Lemma 1. *Let A, B and C be random variables in $\mathbf{Z}_N$ such that*

$$A \overset{p_1}{=} B \qquad B \overset{p_2}{=} C$$

then we assume that $A - B$ and $B - C$ are independent. We have $A \overset{P}{=} C$, where

$$P = \frac{1}{N} + \left(\frac{N}{N-1}\right)\left(p_1 - \frac{1}{N}\right)\left(p_2 - \frac{1}{N}\right) \overset{\mathsf{def}}{=} p_1 \otimes p_2$$

Proof. See Chapter 3 of [24] for the proof. □

Corollary 1. *Let A, B, C, D and E be random variables in $\mathbf{Z}_N$ such that*

$$A \overset{p_1}{=} B \qquad B \overset{p_2}{=} C \qquad C \overset{p_3}{=} D \qquad D \overset{p_4}{=} E$$

then we assume that $A-B$, $B-C$, $C-D$ and $D-E$ are independent. We have $A \overset{P}{=} E$, where

$$P = p_1 \otimes p_2 \otimes p_3 \otimes p_4 = \frac{1}{N} + \left(\frac{N}{N-1}\right)^3 \cdot \prod_{i=1}^{4}\left(p_i - \frac{1}{N}\right)$$

For $p_4 = 1$, we obtain

$$P = p_1 \otimes p_2 \otimes p_3 = \frac{1}{N} + \left(\frac{N}{N-1}\right)^2 \cdot \prod_{i=1}^{3}\left(p_i - \frac{1}{N}\right)$$

Proof. The $\otimes$ operation is commutative and associative over $[0, 1]$ and 1 is the neutral element. The above statements should be trivial using these properties. □

We can extend the above Corollary by adding new conditions.

Lemma 2. *Let A, B, C, D and E be random variables in $\mathbf{Z}_N$ and* Cond *and* Cond′ *be two events such that*

$$A \overset{p_1}{=} B \qquad B \overset{p_2}{=} C \qquad C \underset{\mathsf{Cond}'}{\overset{p_3}{=}} S[D] \qquad D \overset{p_4}{=} E$$

We assume that $A - B$, $B - C$, $C - S[D]$, $D - E$ and Cond′ *are independent; Furthermore, we assume*

$$(A = S[D] \wedge \mathsf{Cond}) \Leftrightarrow (A = S[D] \wedge \mathsf{Cond}') \qquad \textit{and} \qquad \mathsf{Pr}[\mathsf{Cond}] = \mathsf{Pr}[\mathsf{Cond}']$$

Assuming that

$$\mathsf{Pr}[A = S[E] | A \neq S[D], D \neq E, \mathsf{Cond}] = \frac{1}{N-1}$$

we have

$$\mathsf{Pr}[A = S[E] | \mathsf{Cond}] = p_1 \otimes p_2 \otimes p_3 \otimes p_4$$

Proof. See Chapter 3 of [24] for the proof. □

Lemma 3. *To avoid the key byte dependency, the following equation can be extracted to have a better key recovery attack.*

$$\bar{K}[i] = j_i - \sum_{x=1}^{i} S_{x-1}[x]$$

Proof. We prove it by induction on i by using

$$j_i = j_{i-1} + S_{i-1}[i] + K[i]$$

□

Lemma 4. *For $0 \leq t < i$, the following five relations hold on RC4$^\star(t)$ for any set $(m_1, \ldots, m_b)$ of pairwise different m_j's such that $m_j \leq t$ or $m_j > i-1$.*

$$P_A^b(i,t) \overset{\mathsf{def}}{=} \mathsf{Pr}\left[\bigwedge_{j=1}^{b} S_{i-1}[m_j] = \cdots = S_{t+1}[m_j] = S_t[m_j]\right] = \left(\tfrac{N-b}{N}\right)^{i-t-1}$$

$$S_{i-1}[m_j] \overset{P_A^1}{=} S_t[m_j]$$

$$\sum_{x=1}^{i} S_{x-1}[x] \overset{P_B(i,t)}{=} \sigma_i(t) \quad \textit{where}$$

$$P_B(i,t) \overset{\mathsf{def}}{=} \prod_{k=0}^{i-t-1}\left(\frac{N-k}{N}\right) + \frac{1}{N}\left(1 - \prod_{k=0}^{i-t-1}\left(\frac{N-k}{N}\right)\right)$$

$$P_0 \overset{\mathsf{def}}{=} \mathsf{Pr}[S'_{i-1}[i] = \cdots = S'_1[i] = S_{N-1}[i] = \cdots = S_i[i]] = \left(\tfrac{N-1}{N}\right)^{N-2}$$

$$S'_{i-1}[i] \overset{P_0}{=} S_i[i]$$

where m_j's are distinct and

$$\sigma_i(t) = \sum_{j=0}^{t} S_{j-1}[j] + \sum_{j=t+1}^{i} S_t[j]$$

Proof. See Chapter 3 of [24] for the proof. □

6 The List of Biases for RC4

In this section, we only report RC4 correlations which are exploitable against WEP application. All such biases are listed in Table 1 in Appendix, following the notations in Section 3.1. This list includes the improved version of the Klein attack in [34] and the improved version of the Maitra-Paul attack in [15]. Furthermore, it includes an improved version of 19 biases by Korek [14,13] and SVV_10, the improved bias of Sepehrdad, Vaudenay and Vuagnoux in [25]. All the probabilities are new. We have proved all the correlations listed in Table 1, but, we have omitted the proofs due to the lack of space [3]. Biases were computed using the formulas represented after Table 1.

As an example, we are going to elaborate and provide a proof for the Klein-Improved attack, since it is fundamental in our WEP attack. The proof of all the other correlations are similar. The interested reader can also look at [24,26,4] for more details.

6.1 The Klein-Improved Attack

Andreas Klein combined the Jenkins correlation for the PRGA and weaknesses of the KSA and derived a correlation between the key bytes and the keystream. This bias was further improved in [34] by recovering $\bar{K}[i]$'s instead of $K[i]$ to reduce the secret key bytes dependency.

Theorem 1 (Jenkins correlation [10], Sec. 2.3 in [16]). *Assume that the initial permutation $S'_0 = S_{N-1}$ is randomly chosen from the set of all the possible permutations over $\{0, \dots, N-1\}$. Then,*

$$\mathsf{Pr}[S'_i[j'_i] = i - z_i] \approx \frac{2}{N} \qquad \mathsf{Pr}[S'_i[i] = j'_i - z_i] \approx \frac{2}{N}$$

Proof.

$$\begin{aligned}\mathsf{Pr}[S'_i[j'_i] = i - z_i] &= \mathsf{Pr}[S'_i[j'_i] = i - z_i | S'_i[i] + S'_i[j'_i] = i] \ . \ \mathsf{Pr}[S'_i[i] + S'_i[j'_i] = i] \\ &\quad + \mathsf{Pr}[S'_i[j'_i] = i - z_i | S'_i[i] + S'_i[j'_i] \neq i] \ . \ \mathsf{Pr}[S'_i[i] + S'_i[j'_i] \neq i] \\ &= \frac{1}{N} + \frac{1}{N}\left(1 - \frac{1}{N}\right) \approx \frac{2}{N}\end{aligned}$$

By symmetry, the other equation can be proved similarly. □

We use the theorem by Jenkins and explain how it can be merged with the weaknesses of the KSA (see Algorithm 1). In fact, the attacker checks the conditions. If they all hold, she votes for $\bar{K}[i]$ using the key recovery relation. We are only using the assumptions in Algorithm 1 to compute the Klein-Improved attack success probability. More clearly, we do not assume these relations always hold. They are all probabilistic.
Exploiting the Jenkins correlation and the relations in the KSA and the PRGA, we obtain

1. $S'_i[j'_i] \overset{P_J}{=} i - z_i$ (Lemma 1)
2. $S'_i[j'_i] = S'_{i-1}[i]$

[3] See [23] for the proof of SVV_10 bias and for all the others, see Chapter 6 of [24].

```
Algorithm 1 The Klein-Improved Attack
Success Probability: $P_{\mathrm{KI}}(i,t)$
Assumptions: (see Fig. 3)
1: $S_t[j_i] = \cdots = S_{i-1}[j_i] = S_i[i] = S'_{i-1}[i] = S'_i[j'_i] = i - z_i$
Conditions: $(i - z_i) \notin \{S_t[t+1], \ldots, S_t[i-1]\}$ $(\mathsf{Cond})$
Key recovery relation: $\bar{K}[i] = S_t^{-1}[i - z_i] - \sigma_i(t)$
```

3. $S'_{i-1}[i] \overset{P_0}{=} S_i[i]$ (Lemma 4)
4. $S_i[i] = S_{i-1}[j_i]$
5. $S_{i-1}[j_i] \underset{\mathsf{Cond}'}{\overset{P_A^1}{=}} S_t[j_i]$ (where Cond' is the event that $j_i \leq t$ or $j_i > i - 1$.)
6. $j_i = \bar{K}[i] + \sum_{x=1}^{i} S_{x-1}[x]$ (Lemma 3)
7. $\sum_{x=1}^{i} S_{x-1}[x] \overset{P_B}{=} \sigma_i$ (Lemma 4)

We make the same heuristic assumptions of independence as in Lemma 2, then using Lemma 2 and Lemma 4, we derive

$$P_{\mathrm{KI}}(i,t) = P_J \otimes P_0 \otimes P_A^1(i,t) \otimes P_B(i,t)$$

conditioned to Cond (see Algorithm 1). Hence, the key recovery relation becomes

$$\bar{K}[i] \underset{\mathsf{Cond}}{\overset{P_{\mathrm{KI}}}{=}} S_t^{-1}[i - z_i] - \sigma_i(t)$$

Next, we are going to describe our modifications on Sepehrdad, Vaudenay and Vuagnoux attack [26] to mount a very fast key recovery attack on WEP.

7 An Optimised Attack on WEP

We define an statistical attack using the following mapping:

$$z^m, \mathsf{IV}^m \xrightarrow{\quad h_i \quad} \mu_i \xrightarrow[\text{if } g_{\ell_i}(\mu_i)]{f_{\ell_i}(\mu_i)} x_i$$

Our goal is to recover the values of $\bar{K}[i]$'s for $i = \{3, \ldots, 15\}$. For each key candidate value x_i (corresponding to $\bar{K}[i]$), each packet m, and each $\ell_i = 1, \ldots, d_i$ (corresponding to each bias), if the agglomerated condition $g_{\ell_i}(h_i(z^m, \mathsf{IV}^m))$ holds, we define $x_i = f_{\ell_i}(h_i(z^m, \mathsf{IV}^m))$ as the value of the RC4 key byte suggested by the bias ℓ_i on packet m, which is correct with probability p_{ℓ_i}. We let X_{x_i,m,ℓ_i} be some magic coefficient a_{ℓ_i} (to be optimised later) if $f_{\ell_i}(h_i(z^m, \mathsf{IV}^m)) = x_i$ and 0 otherwise. We let

$$Y_{x_i} = \sum_{m=1}^{n} \sum_{\ell_i=1}^{d_i} X_{x_i,m,\ell_i}$$

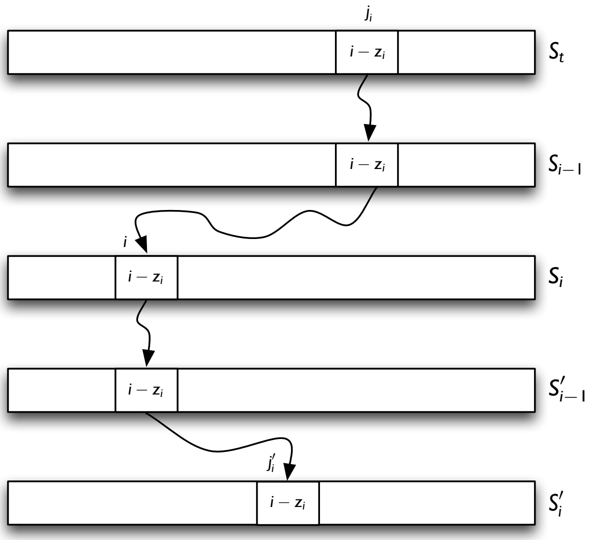

Fig. 3. The RC4 state update in the Klein-Improved attack

where n_i is the total number of packets to be used in attacking $\bar{K}[i]$. Clearly, the correct value for x_i is suggested with probability p_{ℓ_i} and others are obtained randomly. We assume the incorrect ones are suggested with the same probability $\frac{1-p_{\ell_i}}{N_{x_i}-1}$. So, the X_{x_i,m,ℓ_i} for the incorrect x_i's are random variables with the expected values $a_{\ell_i} q_{\ell_i} \frac{1-p_{\ell_i}}{N_{x_i}-1}$ if x_i is not the correct value. For the correct x_i, then X_{x_i,m,ℓ_i} are random variables with the expected value $a_{\ell_i} q_{\ell_i} p_{\ell_i}$. The difference between these two expected values is important. This is also the case for the difference of the variances. As every x_i is suggested with probability roughly $\frac{q_{\ell_i}}{N_{x_i}}$, we assume that the variance of a bad X_{x_i,m,ℓ_i} can be approximated by $\frac{q_{\ell_i}}{N_{x_i}} \left(1 - \frac{q_{\ell_i}}{N_{x_i}}\right) a_{\ell_i}^2$. In [26], it was assumed that the variance of a good and a bad counter Y_{x_i} is the same. Our experiments revealed that they are actually very different. Let Δ be the operator making the difference between the distributions of a good x_i and a bad one. We have

$$\begin{aligned}
E(Y_{x_i\ \mathsf{bad}}) &= \frac{n_i}{N_{x_i}-1}\sum_{\ell_i} a_{\ell_i} q_{\ell_i}(1-p_{\ell_i}) \\
E(Y_{x_i\ \mathsf{good}}) &= E(Y_{x_i\ \mathsf{bad}}) + \mathit{\Delta} E(Y_i) \\
\mathit{\Delta} E(Y_i) &= \frac{n_i}{1-\frac{1}{N_{x_i}}}\sum_{\ell_i} a_{\ell_i} q_{\ell_i}\left(p_{\ell_i}-\frac{1}{N_{x_i}}\right) \\
V(Y_{x_i\ \mathsf{bad}}) &\approx n_i \sum_{\ell_i} a_{\ell_i}^2 \frac{q_{\ell_i}}{N_{x_i}}\left(1-\frac{q_{\ell_i}}{N_{x_i}}\right) \\
V(Y_{x_i\ \mathsf{good}}) &= V(Y_{x_i\ \mathsf{bad}}) + \mathit{\Delta} V(Y_i) \\
\mathit{\Delta} V(Y_i) &\approx \frac{n_i}{1-\frac{1}{N_{x_i}}}\sum_{\ell_i} a_{\ell_i}^2 q_{\ell_i}\left(p_{\ell_i}-\frac{1}{N_{x_i}}\right)
\end{aligned}$$

where $E(Y_{x_i\ \mathsf{bad}})$ and $V(Y_{x_i\ \mathsf{bad}})$ denote the expected value and the variance of a Y_{x_i} variable for any bad x_i respectively. Here, we removed the subscript x_i of Y_{x_i} in $\mathit{\Delta} E(Y_i)$ as this does not depend on a specific value for x_i. Let λ_i be such that $\mathit{\Delta} E(Y_i) = \lambda_i\sqrt{V(Y_{x_i\ \mathsf{bad}}) + V(Y_{x_i\ \mathsf{good}})}$. The probability that the correct Y_{x_i} is lower than an arbitrary wrong Y_{x_i} is $\rho_i = \varphi(-\lambda_i)$. That is, the expected number of wrong x_i's with larger Y_{x_i} is

$$r_i = (N_{x_i}-1)\varphi(-\lambda_i) \tag{2}$$

So,

$$n_i = \frac{\lambda_i^2 \sum_{\ell_i} a_{\ell_i}^2\left[2\left(\frac{q_{\ell_i}}{N_{x_i}}\right)\left(1-\frac{q_{\ell_i}}{N_{x_i}}\right)\left(1-\frac{1}{N_{x_i}}\right)^2 + q_{\ell_i}\left(p_{\ell_i}-\frac{1}{N_{x_i}}\right)\left(1-\frac{1}{N_{x_i}}\right)\right]}{\left(\sum_{\ell_i} a_{\ell_i} q_{\ell_i}\left(p_{\ell_i}-\frac{1}{N_{x_i}}\right)\right)^2}$$

By derivating both terms of the fraction with respect to a_{ℓ_i} and equaling them, we obtain that the optimal value is reached for

$$a_{\ell_i} = a_{\mathsf{opt}_i} \overset{\mathsf{def}}{=} \frac{\left(p_{\ell_i}-\frac{1}{N_{x_i}}\right)}{\left(p_{\ell_i}-\frac{1}{N_{x_i}}\right) + \frac{2}{N_{x_i}}\left(1-\frac{1}{N_{x_i}}\right)\left(1-\frac{q_{\ell_i}}{N_{x_i}}\right)}$$

The above a_{opt_i} is very different from the one derived in [26]. In fact, a_{opt_i} is the most crucial value to be optimised in the WEP attack. Using the old value of a_{opt_i} in [26], the success probability would be much lower. Hence, we obtain

$$n_i = n_{\mathsf{opt}} \overset{\mathsf{def}}{=} \frac{\lambda_i^2\left(1-\frac{1}{N_{x_i}}\right)}{\sum_{\ell_i} a_{\ell_i} q_{\ell_i}\left(p_{\ell_i}-\frac{1}{N_{x_i}}\right)} \tag{3}$$

The attack works as in Algorithm 2, where Step 9 is computed using the below algorithm:

```
Set $I = (3, 4, \ldots, 15)$ and $I_0 = \{0, 1, 2\}$.
Initialize the $Y_{x_i}$ counters to 0.
for $m = 1$ to $n_i$ do
    for $\ell_i = 1$ to $d_i$ do
        if $g_{\ell_i}(h_i(z^m, \mathsf{IV}^m))$ holds then
            Compute $x_i = f_{\ell_i}(h_i(z^m, \mathsf{IV}^m))$, the suggested value for $\bar{K}[i]$.
            Increment $Y_{x_i}$ by $a_{\ell_i}$.
        end if
    end for
end for
Output $x_i = \arg\max_{x_i} Y_{x_i}$.
```

This attack produces a ranking of possible x_i's (possible $\bar{K}[i]$) in the form of a list $\mathcal{L}_i$ by decreasing order of likelihood. The complexity of voting for each $\bar{K}[i]$ is represented as c_i, where

$$c_i = n_i d_i \tag{4}$$

Algorithm 2 An optimised attack against the WEP protocol

```
compute the ranking $\mathcal{L}_{15}$ for $I = (15)$ and $I_0 = \{0, 1, 2\}$
truncate $\mathcal{L}_{15}$ to its first $\rho_{15}$ terms
for each $\bar{k}_{15}$ in $\mathcal{L}_{15}$ do
    run recursive attack on input $\bar{k}_{15}$
end for
stop: attack failed
recursive attack with input $(\bar{k}_{15}, \bar{k}_3, \ldots, \bar{k}_{i-1})$:
If input is only $\bar{k}_{15}$, set $i = 3$.
if $i \leq i_{\max}$ then
    compute the ranking $\mathcal{L}_i$ for $I = (i)$ and $I_0 = \{0, \ldots, i-1, 15\}$
    truncate $\mathcal{L}_i$ to its first $\rho_i$ terms
    for each $\bar{k}_i$ in $\mathcal{L}_i$ do
        run recursive attack on input $(\bar{k}_{15}, \bar{k}_3, \ldots, \bar{k}_{i-1}, \bar{k}_i)$
    end for
else
    for each $\bar{k}_{i_{\max}+1}, \ldots, \bar{k}_{14}$ do
        test key $(\bar{k}_3, \ldots, \bar{k}_{14}, \bar{k}_{15})$ and stop if correct
    end for
end if
```

Let $N_{x_i} = N$ for all i and r_i, c_i be their parameters following Eq. (2), Eq. (4). Let R_i be the rank of the correct $\bar{k}_i$ value in $\mathcal{L}_i$. Let's define a random variable $U_{ij} = 1_{(Y_{x_i \text{ good}} < Y_{x_i \text{ bad}_j})}$, where $Y_{x_i \text{ bad}_j}$ is the j-th bad counter in attacking $\bar{K}[i]$. Hence, we have

$$R_i = \sum_{j=1}^{N_{x_i}-1} U_{ij}$$

The expected value and the variance of this random variable can be computed as follows:

$$r_i = E(R_i) = (N_{x_i} - 1)\varphi(-\lambda_i)$$

$$\text{and} \tag{5}$$

$$E(R_i^2) = E(R_i) + (N_{x_i} - 1)(N_{x_i} - 2) \ \cdot \ E(U_{i1}.U_{i2})$$

where

$$E(U_{i1}.U_{i2}) = \frac{1}{\sqrt{2\pi V(Y_{x_i \ \mathsf{good}})}} \int_{-\infty}^{\infty} e^{-\frac{\left(Y - E(Y_{x_i \ \mathsf{good}})\right)^2}{2V(Y_{x_i \ \mathsf{good}})}} \left(1 - \varphi\left(\frac{Y - E(Y_{x_i \ \mathsf{bad}})}{\sqrt{V(Y_{x_i \ \mathsf{bad}})}}\right)^2\right) dY$$

This finally yields

$$V(R_i) = (N_{x_i} - 1)\varphi(-\lambda_i) + (N_{x_i} - 1)(N_{x_i} - 2) \ . \ E(U_{i1}.U_{i2}) - (N_{x_i} - 1)^2\varphi(-\lambda_i)^2 \tag{6}$$

In [26], U_{i1} and U_{i2} were incorrectly assumed to be independent, leading to

$$V(R_i) \approx (N_{x_i} - 1)\varphi(-\lambda_i)(1 - \varphi(\lambda_i)) \approx r_i$$

which did not match our experiment. Now, the fundamental question is what would be the distribution of R_i. This is discussed in the next section.

7.1 Analysis Based on Pólya Distribution

In [26], it was assumed that the distribution of R_i is normal. Running a few experiments, we noticed that in fact it is not normal and it is following a distribution very close to the Poisson distribution. A crucial observation was that the variance of the distribution was much higher than the expected value. A number of distributions have been devised for series in which the variance is significantly larger than the mean [1,6,18], frequently on the basis of more or less complex biological models [3]. The first of these was the negative binomial, which arose in deriving the Poisson series from the point binomial [29,35]. We use a generalised version of negative binomial distribution called the Pólya distribution.

To be more precise, if two events occur with Poisson distribution and their expected values are very low, then it can be assumed that those events are happening independently. On the other hand, for the Poisson events with high expected values (approximated as normal), the occurrence of the former event may increase the probability of the latter. In such cases, the overall distribution would be the Pólya [32,33]. Regarding the current problem, the events $(Y_{x_i \text{ good}} < Y_{x_i \text{ bad}_j})$ and $(Y_{x_i \text{ good}} < Y_{x_i \text{ bad}_{j'}})$ are not independent. Therefore, they tend to follow the Pólya distribution. As $E(R_i)$ and $V(R_i)$ are known from Eq. (5), Eq. (6), the values p_i and r_i for attacking $\bar{K}[i]$ can be simply computed by

$$p_i = \left(1 - \frac{E(R_i)}{V(R_i)}\right) \qquad \text{and} \qquad r_i = \left(\frac{E(R_i)^2}{V(R_i) - E(R_i)}\right)$$

As a proof of concept, we have sketched the probability distribution of R_3 for 5 000 packets. The corresponding parameters for the Pólya distribution would be $p = 0.9839$ and $r = 0.356$ (see Fig. 4). As can be observed, those two distributions are extremely close. Also,

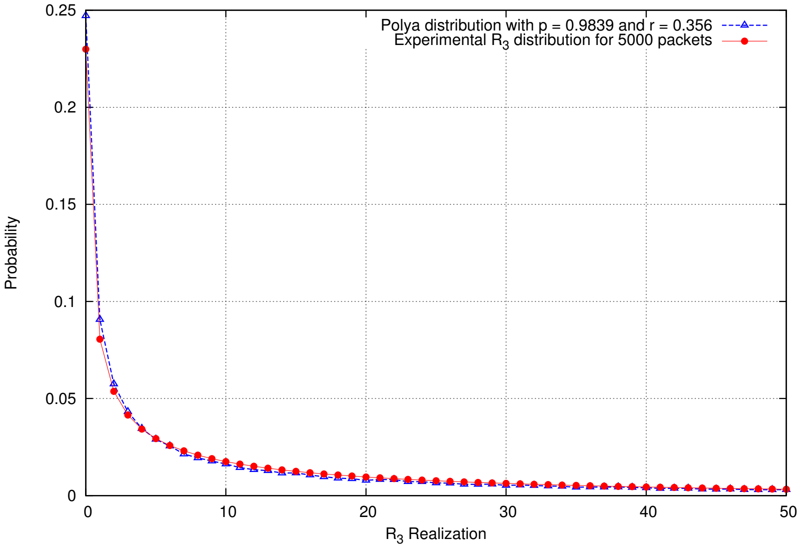

Fig. 4. R_3 distribution using 5 000 packets following the Pólya distribution

$$u_i \overset{\mathsf{def}}{=} \mathsf{Pr}[R_i \leq \rho_i - 1] = 1 - I_{p_i}(\rho_i, r_i)$$

where I is the regularised incomplete beta function. Overall, the success probability is

$$u = u_{15} \prod_{i=3}^{i_{\max}} u_i$$

and the complexity is

$$c = c_{15} + \rho_{15}\left(c_3 + \rho_3\left(c_4 + \rho_4\left(\cdots c_{i_{\max}} + \rho_{i_{\max}} N^{14-i_{\max}} \cdots\right)\right)\right)$$

To be able to compare our results with the state of the art, we set $u = 50\%$. To approximate the optimal choice of ρ's, let $i_{\max} = 14$. We have to deal with the following optimisation problem:

$$\boxed{\text{Minimize } c \text{ in terms of } \rho_i\text{'s, limiting } u = \prod_{i=3}^{15}\left(1 - I_{p_i}(\rho_i, r_i)\right) = \frac{1}{2}}$$

To solve this optimisation problem, we use Lagrange multipliers to find the optimal solution. We used the fmincon function in Matlab with the Sequential Quadratic Programming [19] (SQP) algorithm as the default algorithm to compute the local minimum. As this algorithm needs a starting point x0 for its computations, we used the GlobalSearch class which iterates the fmincon function multiple times using random vectors for x0. Simultaneously, it checks how the results merge towards the global minimum. One can also use Genetic algorithms to find the optimal values.

8 Comparison with Aircrack-ng

Fig. 5 represents a comparison between Aircrack-ng and our new attack. The reader can see that our passive attack outperforms Aircrack-ng running in active mode. This gives significant advantage to the attacker, since for some network cards, the driver has to be patched so that the network card can inject packets, and in some cases such patch is not available at all. Moreover, the active attacks are detectable by intrusion detection systems. Similarly, passive attacks can be performed from much large distance. Moreover, the TCP/IPv4 packets can be captured with much higher rate than ARP packets. As a rule of thumb, in a high traffic network, (for instance the user is downloading a movie), if we consider TCP/IPv4 packets with maximum size around 1500 bytes, in a 20 Mbit/sec wireless network, it takes almost 10 seconds to capture 22 500 packets. This amount is already enough to find a key with our improved Aircrack-ng in less than 5 seconds.

9 Challenges and Open Problems

WEP key recovery process is harder in practice than in theory. This is because the biases in RC4 are not independent, and several bytes of the keystream are unknown in ARP and TCP/IP packets. Therefore, the theoretical analysis is more complex if the dependencies are considered. Also, some bytes of the keystream have to be guessed, and the proportion of TCP/IP packets to ARP packets is distinct for every network and attack (passive vs. active). The a priori probability of guessing those bytes correctly can not be precisely determined, and we had to leverage some heuristics to deal with this problem; Since this proportion also depends on the traffic itself, finding the ρ which is optimised for every network is not feasible. We leveraged some heuristics to set the ρ to obtain a high success rate in practice. Moreover, the Aircrack-ng is not an interactive software. The interaction with the user may allow to tweak the ρ and/or wait for more packets to capture. This trade-off should also be considered in real life applications.

The Algorithm 2 is recursive. This recursion is very expensive in practice, since with a wrong guess on a key byte, all the subsequent key bytes with higher indices are recovered incorrectly (in theory), so we need to recompute the vote for each of them again. In practice, we observed that a wrong guess of a key byte *does not* influence the next key bytes recovery significantly. For instance, even with

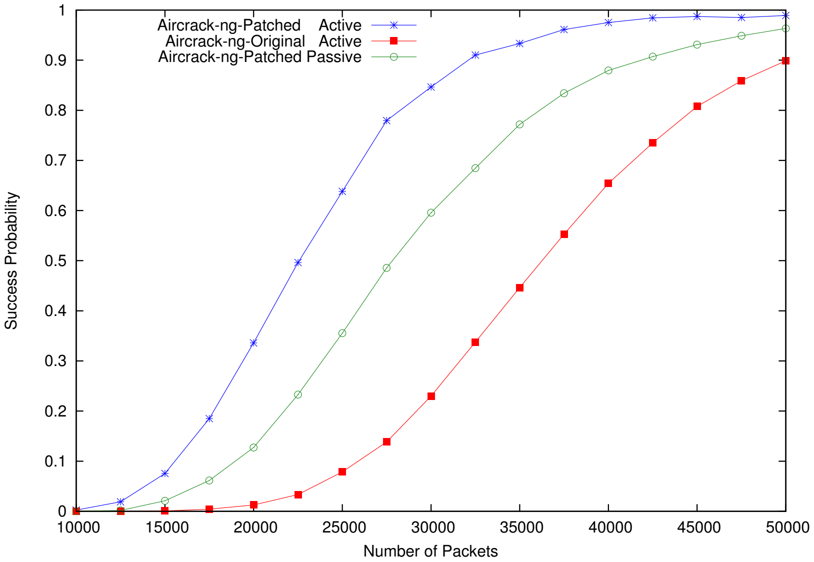

Fig. 5. Our attacks success probability (both active and passive attacks) with respect to the number of packets compared to Aircrack-ng in active attack mode.

a wrong guess on $\bar{K}[3]$, in many cases, we could still recover all the subsequent bytes correctly. This is because a wrong guess for $\bar{K}[3]$ mandates only 16 wrong swaps out of 256 iterations of the KSA. A further improvement to our work can be to adjust our theory to consider such cases. Hence, in our implementation, we perform a recursive attack to only find the best key candidate, and if it turns out to be a wrong key, we then use the pre-computed voted list to perform an exhaustive search, with no re-voting.

10 Conclusion

In this paper, we gave a precise theoretical background to improve the state of the art attacks on WEP. As an empirical proof, we updated Aircrack-ng and showed that our attack significantly outperforms the previous versions in all scenarios. We modified the algorithm according to the theoretical results, removed the ad-hoc constants which were initially found empirically in previous papers and implementations. We gave a theoretical background for all constants which affect the performance of the new Aircrack-ng. This result shows the significance of theoretical analysis in practical scenarios, and allows the attacker to break WEP even on constrained devices. As a result, the best attack to date requires 22 500 packets for the success probability of 50% to break WEP.

Note. The imprecision of distributions and variances also affect our analysis reported for WPA in [26]. But, we recomputed all numerical values with the precise theoretical formulas and observed only a negligible overheard compared to the derived complexity in [26].

Acknowledgment. We would like to sincerely thank Dr. Erik Tews for giving very helpful comments on Aicrack-ng implementation.

References

1. F.J. Anscombe. Sampling theory of the negative binomial and logarithmic series distributions. *Biometrika*, 37(3-4):358–382, 1950.
2. M. Beck and E. Tews. Practical Attacks Against WEP and WPA. In *WISEC*, pages 79–86. ACM, 2009.
3. C.I. Bliss and R.A. Fisher. Fitting the Negative Binomial Distribution to Biological Data. *Biometrika*, 9:176–200, 1953.
4. R. Chaabouni. Break WEP Faster with Statistical Analysis. Semester Project, EPFL, Switzerland, 2006.
5. C. Devine and T. Otreppe. Aircrack-ng, accessed October 22, 2011. http://www.aircrack-ng.org/.
6. W. Feller. On a general class of "contagious" distributions. *Ann. Math. Stat.*, 14:389–400, 1943.
7. S.R. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. In *SAC*, volume 2259, pages 1–24. Springer, 2001.
8. IEEE. IEEE Std 802.11, Standards for Local and Metropolitan Area Networks: Wireless Lan Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.
9. IEEE. ANSI/IEEE standard 802.11i, Amendment 6 Wireless LAN Medium Access Control (MAC) and Physical Layer (phy) Specifications, 2003. Draft 3.
10. R. Jenkins. ISAAC and RC4, 1996. http://burtleburtle.net/bob/rand/isaac.html.
11. A. Klein. Attacks on the RC4 Stream Cipher. *Design, Codes, and Cryptography*, 48:269–286, 2008.
12. Korek. chopchop (experimental WEP attacks), 2004. http://www.netstumbler.org/showthread.php?t=12489.
13. Korek. Need Security Pointers, 2004.
http://www.netstumbler.org/showthread.php?postid=89036#pos%t89036.
14. Korek. Next Generation of WEP Attacks?, 2004.
http://www.netstumbler.org/showpost.php?p=93942&postcount=%35.
15. S. Maitra and G. Paul. New Form of Permutation Bias and Secret Key Leakage in Keystream Bytes of RC4. In *FSE*, volume 5086, pages 253–269. Springer, 2008.
16. I. Mantin. Analysis of the Stream Cipher RC4. Master's thesis, Weizmann Institute of Science, 2001.
17. A. Maximov. Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness. In *FSE*, volume 3557, pages 342–358. Springer, 2005.
18. J. Neyman. On a new class of "contagious" distributions, applicable in entomology and bacteriology. *Ann. Math. Stat.*, 10:35–57, 1939.
19. J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer Series in Operations Research. Springer Verlag, second edition, 2006.

20. G. Paul and S. Maitra. Permutation After RC4 Key Scheduling Reveals the Secret. In *SAC*, volume 4876, pages 360–377. Springer, 2007.
21. J. Postel and J. Reynolds. A Standard for the Transmission of IP Datagrams over IEEE 802 Networks, 1988. http://www.cs.berkeley.edu/∼daw/my-posts/my-rc4-weak-keys.
22. A. Roos. A Class of Weak Keys in RC4 Stream Cipher (sci.crypt), 1995. http://marcel.wanda.ch/Archive/WeakKeys.
23. S. Sen Gupta, S. Maitra, G. Paul, and S. Sarkar. (Non)Random Sequences from (Non)Random Permutations - Analysis of RC4 Stream Cipher. *Journal of Cryptology*, 2012.
24. P. Sepehrdad. *Statistical and Algebraic Cryptanalysis of Lightweight and Ultra-lightweight Symmetric Primitives*. PhD thesis, EPFL, Switzerland, 2012.
25. P. Sepehrdad, S. Vaudenay, and M. Vuagnoux. Discovery and Exploitation of New Biases in RC4. In *SAC*, volume 6544, pages 74–91. Springer, 2010.
26. P. Sepehrdad, S. Vaudenay, and M. Vuagnoux. Statistical Attack on RC4: Distinguishing WPA. In *EUROCRYPT*, volume 6632, pages 343–363. Springer, 2011.
27. A. Stubblefield, J. Ioannidis, and A.D. Rubin. Using the Fluhrer, Mantin, and Shamir Attack to Break WEP. *Network and Distributed System Security Symposium (NDSS)*, 2002.
28. A. Stubblefield, J. Ioannidis, and A.D. Rubin. A key recovery attack on the 802.11b wired equivalent privacy protocol (WEP). *ACM Transactions on Information and System Security (TISSEC)*, 7(2), 2004.
29. Student. On the error of counting with a haemocytometer. *Biometrika*, 5:351–360, 1907.
30. E. Tews. Attacks on the WEP Protocol. In *Cryptology ePrint Archive*, 2007. http://eprint.iacr.org/2007/471.pdf.
31. E. Tews, R. Weinmann, and A. Pyshkin. Breaking 104 Bit WEP in Less Than 60 Seconds. In *WISA*, volume 4867, pages 188–202. Springer, 2007.
32. H.C.S. Thom. The Frequency of Hail Occurrence. *Theoretical and Applied Climatology*, 8:185–194, 1957.
33. H.C.S. Thom. Tornado Probabilities. *American Meteorological Society*, pages 730–736, 1963.
34. S. Vaudenay and M. Vuagnoux. Passive-only Key Recovery Attacks on RC4. In *SAC*, volume 4876, pages 344–359. Springer, 2007.
35. L. Whitaker. On the Poisson law of small numbers. *Biometrika*, 10:36–71, 1914.

A IEEE 802.11 Data Frames Encapsulating ARP and TCP/IPv4 Protocols

ARP Packet	
0xAA	DSAP
0xAA	SSAP
0x03	CTRL
0x00	ORG Code
0x00	
0x00	
0x08	ARP
0x06	
0x00	Ethernet
0x01	
0x08	IP
0x00	
0x06	Hardware size
0x04	Protocol
0x00	Opcode Request/Reply
0x??	
0x??	MAC addr src
0x??	
0x??	
0x??	
0x??	
0x??	
0x??	IP src
0x??	
0x??	
0x??	
0x??	MAC addr dst
0x??	
0x??	
0x??	
0x??	
0x??	

TCP/IPv4 Packet	
0xAA	DSAP
0xAA	SSAP
0x03	CTRL
0x00	ORG Code
0x00	
0x00	
0x08	IP
0x00	
0x45	IP Version + Header length
0x00	Type of Service
0x??	Packet length
0x??	
0x??	IP ID RFC815
0x??	
0x40	Fragment type and offset
0x??	
0x??	TTL
0x06	TCP type
0x??	Header checksum
0x??	
0x??	IP src
0x??	
0x??	
0x??	
0x??	IP dst
0x??	
0x??	
0x??	
0x??	Port src
0x??	
0x??	Port dst
0x??	

Fig. 6. The plaintext bytes of the 802.11 data frames encapsulating ARP and TCP/IPv4 protocols [31,34]. The values in white are almost fixed or can be computed dynamically. The values in light Grey can be guessed. The values in dark Grey are not predictable. Often one of Port src or Port dest can be guessed, but not both.

B Computation of Biases

Table 1. The biases for RC4, exploitable against WEP and WPA

row	reference	$\bar{f}$	$\bar{g}$	p
i	Klein − Improved	$S_t^{-1}[-z_i+i]-\sigma_i(t)$	$(i-z_i)\notin\{S_t[t+1],\dots,S_t[i-1]\}$	$P_{\mathrm{KI}}(i,t)$
$i\neq 1$	MP − Improved	$z_{i+1}-\sigma_i(t)$	$i\neq 1,\ z_{i+1}\geq i,\ \forall 0\leq i'\leq t: j_{i'}\neq z_{i+1}$	$P_{\mathrm{MPI}}(i,t)$
i	A_u15	$2-\sigma_i(t)$	$S_t[i]=0,\ z_2=0$	$P^1_{\mathsf{fixed}-j}$
i	A_s13	$S_t^{-1}[0]-\sigma_i(t)$	$S_t[1]=i,\ (S_t^{-1}[0]<t+1$ or $S_t^{-1}[0]>i-1),\ z_1=i$	Kor_1^2
i	A_u13_1	$S_t^{-1}[z_1]-\sigma_i(t)$	$S_t[1]=i,\ (S_t^{-1}[z_1]<t+1$ or $S_t^{-1}[z_1]>i-1),\ z_1=1-i$	Kor_1^2
i	A_u13_2	$1-\sigma_i(t)$	$S_t[i]=i,\ S_t[1]=0,\ z_1=i$	$P^3_{\mathsf{fixed}-j}$
i	A_u13_3	$1-\sigma_i(t)$	$S_t[i]=i,\ S_t[1]=1-i,\ z_1=1-i$	$P^3_{\mathsf{fixed}-j}$
i	A_s5_1	$S_t^{-1}[z_1]-\sigma_i(t)$	$S_t[1]<t+1,\ S_t[1]+S_t[S_t[1]]=i,\ z_1\neq\{S_t[1],S_t[S_t[1]]\},\ (S_t^{-1}[z_1]<t+1$ or $S_t^{-1}[z_1]>i-1)$	Kor_2^3
i	A_s5_2	$S_t^{-1}[S_t[1]-S_t[2]]-\sigma_i(t)$	$S_t[2]+S_t[1]=i,\ S_t^{-1}[S_t[1]-S_t[2]]\neq\{1,2\},\ (S_t^{-1}[S_t[1]-S_t[2]]<t+1$ or $S_t^{-1}[S_t[1]-S_t[2]]>i-1),\ z_2=S_t[1]$	Kor_2^3
i	A_s5_3	$S_t^{-1}[z_2]-\sigma_i(t)$	$S_t[2]+S_t[1]=i,\ S_t^{-1}[z_2]\neq\{1,2\},$ $(S_t^{-1}[z_2]<t+1$ or $S_t^{-1}[z_2]>i-1),$ $z_2=2-S_t[2]$	Kor_2^3
i	A_u5_1	$S_t^{-1}[S_t^{-1}[z_1]-i]-\sigma_i(t)$	$S_t[1]=i, S_t^{-1}[z_1]<t+1, S_t^{-1}[S_t^{-1}[z_1]-i]\neq 1,\ (S_t^{-1}[S_t^{-1}[z_1]-i]<t+1$ or $S_t^{-1}[S_t^{-1}[z_1]-i]>i-1),\ z_1\neq\{i,1-i,S_t^{-1}[z_1]-i\},\ S_t^{-1}[z_1]\neq 2i$	Kor_2^3
i	A_u5_2	$1-\sigma_i(t)$	$S_t[i]=1,\ z_1=S_t[2]$	$P^2_{\mathsf{fixed}-j}$
i	A_u5_3	$1-\sigma_i(t)$	$S_t[i]=i,\ S_t^{-1}[z_1]\neq 1,\ S_t^{-1}[z_1]<t+1,$ $z_1=S_t[S_t[1]+i]$	$P^5_{\mathsf{fixed}-j}$
i	A_s3	$S_t^{-1}[z_2]-\sigma_i(t)$	$S_t[1]\neq 2, S_t[2]\neq 0, S_t[2]+S_t[1]<t+1,$ $S_t[2]+S_t[S_t[2]+S_t[1]]=i,\ S_t^{-1}[z_2]\neq\{1,2,S_t[1]+S_t[2]\},\ S_t[1]+S_t[2]\neq\{1,2\},\ (S_t^{-1}[z_2]<t+1$ or $S_t^{-1}[z_2]>i-1)$	Kor_3^4
4	A_4_s13	$S_t^{-1}[0]-\sigma_4(t)$	$S_t[1]=2,\ S_t[4]\neq 0,\ (S_t^{-1}[0]<t+1$ or $S_t^{-1}[0]>i-1),\ z_2=0$	$P^4_{\mathsf{fixed}-j}$
4	A_4_u5_1	$S_t^{-1}[N-2]-\sigma_4(t)$	$S_t[1]=2,\ z_2\neq 0,\ z_2=S_t[0],\ z_2\neq N-2,\ (S_t^{-1}[N-2]<t+1$ or $S_t^{-1}[N-2]>3)$	Kor_2^3
4	A_4_u5_2	$S_t^{-1}[N-1]-\sigma_4(t)$	$S_t[1]=2,\ z_2\neq 0,\ (S_t^{-1}[N-1]<t+1$ or $S_t^{-1}[N-1]>3),\ z_2=S_t[2]$	Kor_2^3
i	A_neg_1	$1-\sigma_i(t)$ or $2-\sigma_i(t)$	$S_t[2]=0,\ S_t[1]=2,\ z_1=2$	$P_{\mathrm{neg}}(i,t)$
i	A_neg_2	$2-\sigma_i(t)$	$S_t[2]=0,\ S_t[1]\neq 2,\ z_2=0$	$P_{\mathrm{neg}}(i,t)$
i	A_neg_3	$1-\sigma_i(t)$ or $2-\sigma_i(t)$	$S_t[1]=1,\ z_1=S_t[2]$	$P_{\mathrm{neg}}(i,t)$
i	A_neg_4	$-\sigma_i(t)$ or $1-\sigma_i(t)$	$S_t[1]=0,\ S_t[0]=1,\ z_1=1$	$P_{\mathrm{neg}}(i,t)$
16	SVV_10	$S_t^{-1}[0]-\sigma_{16}(t)$	$S_t^{-1}[0]<t+1$ or $S_t^{-1}[0]>15,\ z_{16}=-16,\ j_2\notin\{t+1,\dots,15\}$	$P_{\mathsf{SVV10}}(t)$

$$
\begin{aligned}
P_{\mathrm{KI}}(i,t) &= P_J \otimes P_0 \otimes P_A^1(i,t) \otimes P_B(i,t) \\
P_{\mathrm{MPI}}(i,t) &= P_D(i) \otimes P_B(i,t) \\
\mathsf{Kor}_c^b(i,t) &= R_c^b(i,t) \otimes P_B(i,t) \\
P_{\mathrm{neg}}(i,t) &= \left(\tfrac{1-P_B(i,t)}{N-1}\right) \\
P_{\mathsf{SVV10}}(t) &= P_{db2} \otimes P_A^1(16,t) \otimes P_B(16,t)
\end{aligned}
$$

$$
\begin{aligned}
&P^1_{\mathsf{fixed}-j} = C(i,t).\left[\tfrac{1}{2}P_A^1(i,t)\left(\tfrac{N-1}{N}\right)^{N-i} + \tfrac{1}{N}\left(1 - P_A^1(i,t)\left(\tfrac{N-1}{N}\right)^{N-i}\right)\right] \\
&+P_{\mathrm{neg}}(i,t) \\
&P^2_{\mathsf{fixed}-j} = \left[\tfrac{1}{\xi}P_A^2(i,t)\left(\tfrac{N}{N-1}\right)^{t-2}\left(\tfrac{N-2}{N}\right)^{N-1-i} + \left(1 - P_A^2(i,t)\left(\tfrac{N-2}{N}\right)^{N-i-1}\right)\right]. \\
&\left(\tfrac{1}{N}\right)C(i,t) + P_{\mathrm{neg}}(i,t) \\
&P^3_{\mathsf{fixed}-j} = \left[\left(\tfrac{N-1}{N}\right)^{t+1}\left(\tfrac{N-2}{N}\right)^{N-1-i}\ .\ P_A^2(i,t) + \tfrac{1}{N}\left(1 - P_A^2(i,t)\left(\tfrac{N-2}{N}\right)^{N-i-1}\right)\right]. \\
&C(i,t) + P_{\mathrm{neg}}(i,t) \\
&P^4_{\mathsf{fixed}-j} = \left[\tfrac{1}{2}\left(\tfrac{N-1}{N}\right)^{t+1}\left(\tfrac{N-2}{N}\right)^{N-1-i}\ .\ P_A^2(i,t) + \tfrac{1}{N}\left(1 - P_A^2(i,t)\left(\tfrac{N-2}{N}\right)^{N-i-1}\right)\right]. \\
&C(i,t) + P_{\mathrm{neg}}(i,t) \\
&P^5_{\mathsf{fixed}-j} = \left[\tfrac{\left(\frac{N-1}{N}\right)^{t+1}\left(\frac{t}{N}\right)\left(\frac{N-3}{N}\right)^{N-1-i}}{\left(1-\frac{1}{N}\right)\left(\frac{N-1}{N}\right)^{t+1}\left(\frac{t}{N}\right)+\frac{1}{N}}\ .\ P_A^3(i,t) + \tfrac{1}{N}\left(1 - P_A^3(i,t)\left(\tfrac{N-3}{N}\right)^{N-i-1}\right)\right]. \\
&C(i,t) + P_{\mathrm{neg}}(i,t)
\end{aligned}
$$

where $P_J = \frac{2}{N}$, $P_0 = \left(\frac{N-1}{N}\right)^{N-2}$, $P_{\mathsf{db2}} = \frac{9.444}{N}$,
$\xi = \frac{1}{N}\left[\left(\frac{N-1}{N}\right)^N\left(1 - \frac{1}{N} + \frac{1}{N^2}\right) + \frac{1}{N^2} + 1\right]$.

$$
\begin{aligned}
C(i,t) &= \left(\tfrac{NP_B(i,t)-1}{N-1}\right) \\
P_A^b(i,t) &= \left(\tfrac{N-b}{N}\right)^{i-t-1} \\
P_B(i,t) &= \textstyle\prod_{k=0}^{i-t-1}\left(\tfrac{N-k}{N}\right) + \tfrac{1}{N}\left(1 - \prod_{k=0}^{i-t-1}\left(\tfrac{N-k}{N}\right)\right) \\
P_D(i) &= \tfrac{(N-i-1)(N-i)}{N^3}\left(\tfrac{N-2}{N}\right)^{N-3+i}\left(\tfrac{N-1}{N}\right)^3 \\
R_c^b(i,t) &= r_c(i)P_A^b(i,t) + \tfrac{1}{N}(1 - r_c(i)P_A^b(i,t)) \\
r_1(i) &= \left(\tfrac{N-2}{N}\right)^{N-i-1} \\
r_2(i) &= \left(\tfrac{N-3}{N}\right)^{N-i-1} \\
r_3(i) &= \left(\tfrac{N-4}{N}\right)^{N-i-1}
\end{aligned}
$$