ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Break WEP Faster with Statistical Analysis

Rafik Chaabouni

School of Computer and Communication Sciences

Semester Project

June 2006

Responsible
Prof. Serge Vaudenay
EPFL / LASEC

Supervisor
Martin Vuagnoux
EPFL / LASEC

LASEC

Contents

1 Introduction **5**

2 RC4 **7**
2.1 KSA . . . 7
2.2 PRGA . . . 8
2.3 RC4 analysis . . . 9

3 WEP **11**

4 Korek attacks **15**
4.1 1st attack: Korek A_s5_1 . . . 16
4.2 2nd attack: Korek A_s13 . . . 17
4.3 3rd attack: Korek A_u13_1 . . . 18
4.4 4th attack: Korek A_u5_1 . . . 20
4.5 5th attack: Korek A_u5_2 . . . 22
4.6 6th attack: Korek A_u13_2 . . . 23
4.7 7th attack: Korek A_u13_3 . . . 25
4.8 8th attack: Korek A_u5_3 . . . 26
4.9 9th attack: Korek A_s3 . . . 28
4.10 10th attack: Korek A_u15 . . . 30
4.11 11th attack: Korek A_s5_2 . . . 32
4.12 12th attack: Korek A_s5_3 . . . 35
4.13 13th attack: Korek A_4_s13 . . . 37
4.14 14th attack: Korek A_4_u5_1 . . . 40
4.15 15th attack: Korek A_4_u5_2 . . . 42
4.16 16th attack: Korek A_u5_4 . . . 45
4.17 17th attack: Korek A_neg . . . 48
4.17.1 . . . 48
4.17.2 . . . 48
4.17.3 . . . 49
4.17.4 . . . 50

5 New attacks? **51**
5.1 18th attack: Mansor A_4_s5_1 . . . 51

6 Conclusion **55**

Chapter 1

Introduction

The WEP (Wired Equivalent Privacy) protocol has been created in order to provide privacy to the 802.11 based wireless space. This protocol is a weak version of RC4, because the initial vector used before the secret key, to generate the output stream, needs to be public. In this analysis we will first give an overview of what is RC4, to be able to fully understand the WEP protocol, then we will present the first generalized attack done on it, know as the Fluhrer, Mantin and Shamir (FMS). In the next chapter we will explain the recent attacks presented by Korek as a piece of code on the NetStumbler forum. And finally, we will present a new attack and we will try to demonstrate how new potential attacks can still be found.

Chapter 2

RC4

RC4 is a stream cipher designed by Ron Rivest in 1987, kept trade secret until an anonymous release in 1994. This stream cipher is decomposed into two parts: a Key Scheduling Algorithm (KSA) and a Pseudo-Random Generation Algorithm (PRGA). The KSA is fed with a key, which will set up an internal state for the PRGA. PRGA will then be able to generate a pseudo-random output sequence. Here is a high level overview of the RC4 scheme:

$$\begin{tikzcd}
\text{key K} \arrow[r] & \boxed{\text{KSA}} \arrow[r] & \text{Internal State} \arrow[r] & \boxed{\text{PRGA}} \arrow[r] & \text{Key Stream}
\end{tikzcd}$$

In the following chapter we will deeper the mechanism above explaining KSA and PRGA.

2.1 KSA

This algorithm takes as input the secret key K of length l, begins with the initialization of the internal state S to set it as the identity permutation, and then uses the key K to scramble S.

```
KSA($K$)
Initialization:
    For $i = 0$ to $N - 1$
        $S[i] = i$
    $j = 0$
Scrambling:
    For $i = 0$ to $N - 1$
        $j = (j + S[i] + K[i \bmod l]) \bmod N$
        Swap($S[i], S[j]$)
```

After the initialization, the internal state S looks like this:

0	1	2	3	...	$N-1$

Then the index i and j are reset to 0, in order to proceed with the scrambling. This is done by stepping i across S, while updating the value of j with its previous value, plus the element of S at position i, plus the i^{th} modulo l element of K. This j computed value is used modulo N to swap the elements of S pointed by i and j. Let us denote each round of this loop as being a *step*. For instance if we have the following key K:

0	1	2	3	...
4	8	242	254	...

The internal state will evolve this way:

Initial S:

0	1	2	3	4	...	N-1
0	1	2	3	4	...	$N-1$

The first step will give us $i = 0$ and $j = 0 + S[0] + K[0] = 0 + 4 = 4$:

0	1	2	3	4	...	N-1
4	1	2	3	0	...	$N-1$

The second step will give us $i = 1$ and $j = 4 + S[1] + K[1] = 4 + 1 + 8 = 13$:

0	1	2	3	4	...	N-1
4	13	2	3	0	...	$N-1$

The internal state is a permutation of size $N = 2^n$ of all possible n bits word. This gives us a space of size $2^n!$. Along with it i and j take a value between 0 and $2^n - 1$, resulting to a total space size of $2^n! \cdot 2^8 \cdot 2^8 = 2^n! \cdot (2^8)^2$.

2.2 PRGA

Similarly to the KSA, the PRGA initializes i and j to 0 and enter into an infinite loop that produces the key output stream.

```
PRGA($K$)
Initialization:
    $i = 0$
    $j = 0$
Generation loop:
    $i = (i + 1) \bmod N$
    $j = (j + S[i]) \bmod N$
    Swap($S[i], S[j]$)
    Output $z = S[(S[i] + S[j]) \bmod N]$
```

The loop consists of increasing i by one (while respecting a maximum value of $N-1$) and j by the value $S[i]$. Then both values $S[i]$ and $S[j]$ are swapped and the key stream output byte is computed according to this formula: $z = S[(S[i]+S[j]) \bmod N]$. Let us continue our previous example in KSA. We recall that our key is:

0	1	2	3	...
4	8	242	254	...

And let us assume that our internal state after the KSA is as follows:

0	1	2	3	4	5	...
4	2	3	13	43	7	...

In the initialization of PRGA, i is set to 0 as well as j. Then i is incremented by one ($i = 1$) while j is incremented by $S[1] = 2$ ($j = 2$). $S[i]$ and $S[j]$ are swapped resulting to the following internal state:

0	1	2	3	4	5	...
4	3	2	13	43	7	...

Thus the first key output byte is $z = S[S[1] + S[2]] = S[5] = 7$.

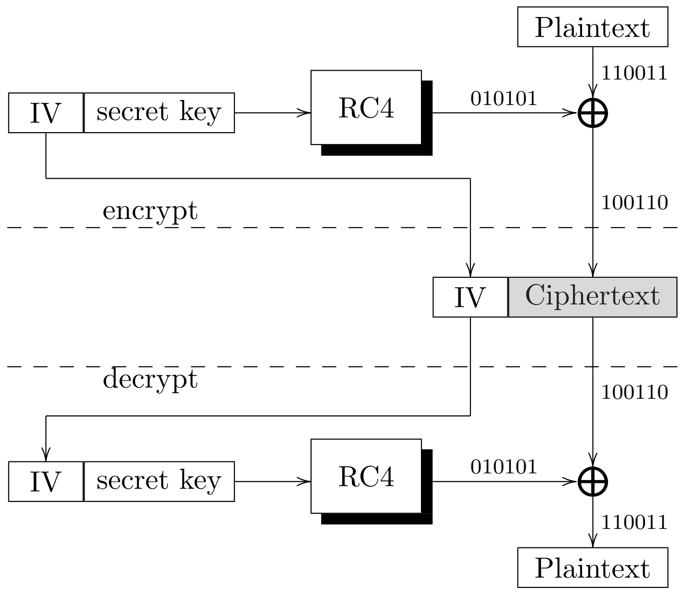
2.3 RC4 analysis

RC4 has initiated extensive research due to its simplicity, resulting to several weaknesses identified. First large classes of weak keys exist. These classes determine a large part of the internal state from a small chunk of the secret key. The patterns resulting from the KSA fed with this weak keys are translated with the PRGA into patterns in the prefix of the output stream. Secondly, the knowledge of some bytes of the secret key brings a high amount of knowledge on the other ones. This is done by analyzing encrypted packets with several exposed values of the key. This particular case is the major flaw in WEP because this protocol uses an Initial Vector (IV) of three bytes, preceding the secret key and sent in clear with the encrypted packet.

Chapter 3

WEP

WEP, Wired Equivalent Privacy, is a protocol based on the RC4 stream cipher. Its purpose is to enable the encryption of packets in a wireless environment. To do so, WEP injects a seed composed by a public Initial Vector (IV) preceding a secret key to the RC4 module, that will generate a key stream used in the encryption of data.



The size of the internal state is fixed and equal to $N = 256$. The initial vector is a three bytes size seed that is as random as possible in order to prevent attacks. However their knowledge (they are sent in clear) combined with the knowledge of the first output bytes of the PRGA, enables attacks based on statistical analysis. The first important attack called the FMS attack according to its authors (Fluhrer, Mantin and Shamir), started by looking at the IVs in order to find a relation enabling them to perform an attack. We will see in the next chapter why this approach is not as powerful as the one used by Korek. Thus they came with the following relation: if the IV is of the form $\begin{array}{|c|c|c|}\hline A+3 & 255 & X \\ \hline\end{array}$ the key which has the index $A+3$ can be recovered. The reason of the 3 is not to take into account the bytes of the IV. For instance if we want to recover the value of $K[3]$ and we receive in our network sniffing the following IV: $\begin{array}{|c|c|c|}\hline 3 & 255 & X \\ \hline\end{array}$ we can estimate the value of $K[3]$ according to the information provided by the first output byte $o1$. Let us follow the sequence of events that occurs during the KSA to understand how this relation react. (Please note that we are working modulus N, where N is equal to 256, in all the rest of this research).

Initialization of the KSA:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 3$

1st step:

0	1	2	3	4	5	6	...
3	1	2	**0**	4	5	6	...

$i_1 = 1$, $j_1 = 3$ (recall that $255 + 1 = 0$)

2nd step:

0	1	2	3	4	5	6	...
3	**0**	2	**1**	4	5	6	...

$i_2 = 2$, $j_2 = 5 + X$

3rd step:

0	1	2	3	4	5	6	...
3	0	$\boldsymbol{5+X}$	1	4	5	6	...

$i_3 = 3$, $j_3 = X + 6 + K[3]$
(recall that we are looking for $K[3]$ value)

4th step:

0	1	2	3	4	5	6	...
3	0	$5+X$	$\boldsymbol{S^3[X+6+K[3]]}$	4	5	6	...

Where S^3 represents the internal state in the 3rd step.

Now we will suppose that some values of the internal state will stay unchanged during the rest of the KSA process. This assumption is a key element in all the attacks because we need to have some degree of assumption on the internal state that will be presented to the PRGA. The number of unchanged elements and the step from which they must stay unchanged will determine the probability of success of these attacks. In this example we are willing that $S[0]$, $S[1]$ and $S[3]$ stay unchanged after the 4th step of the KSA, giving us a probability of success of 5.127% (we will explain in the next chapter how to compute this approximation). If we come back to our example, once our desired internal state will be presented to the PRGA, a swap will be made between $S[0]$ and $S[1]$ because $S[1] = 0$. Then $o1$ will receive the value $S[S[0] + S[1]] = S[0 + 3] = S[3] = S^3[X + 6 + K[3]]$. Therefore we are able to identify the value of $K[3]$ as being equal to $Si[o1]-$

$X-6$ (here Si represents the indexes of the elements in the third step of the KSA: $Si[3]=0$, $Si[0]=1$). The authors of the FMS attack managed to enhance their relation to the cases where $(S[1]+S[S[1]]) \bmod 256 = p$. The drawback of their method is that network constructors can easily discard packets that correspond to known IV relation.

Chapter 4

Korek attacks

While discussing in an Internet forum (NetStumbler), a person under the pseudonym of Korek published a code that describes seventeen attacks on the WEP protocol. These attacks can be regrouped in three major parts: a first group tries to recover key bytes based on the first outputted key byte from the stream generated by the PRGA, while the second one also include the knowledge of the first and the second outputted key byte. The third group consists of reverse methods to reduce the size of the search space, also known as “inverted attacks”.

We will explain in this section how the current attacks are done and what is the approximation of their success rate.

The approach used to perform the attacks are now no more based on identifiable IVs but on how they make the KSA and the PRGA behave, hence no more blind filtering can be done at a router. In order to proceed with each attack some preprocessing is done. Let us assume that we attack the n^{th} key byte (the three first ones being the IV - in the rest of this report we will assume that $n - 1 = p$ and in order to follow the logic of tables in C, the first key byte being in position 0), hence the first byte of the key unknown to an attacker. The KSA has to be run until this unknown key byte is reached (p steps needed), which will provide us an approximation of the final internal state (that we will denote as S) for the elements that are at the beginning of the internal state. While doing this process we record all the values that j takes and use this information to build a table containing the position of each element in the approximated value of the internal state. This last table will be denoted as Si.

The approximation done on the success rate formula, express our motivation to look at the probability that some elements of the internal state do not get modified in the rest of the KSA. Thus after p steps we want to estimate the probability that q elements must stay unchanged (up to the

PRGA). Thus we can estimate this probability as such:

$$\left(\frac{256-q}{256}\right)^{256-p}$$

4.1 1st attack: Korek A_s5_1

Probability of success: $\left(\frac{253}{256}\right)^{256-p} \approx 5.07\%$ $(p = 3)$

This first attack is the generalization of the classic FMS attack seen previously. We hence select all the IV that has the following properties:

1. $S[1] < p$
   This is done in order to maximize the chances that the first part of the internal states stay unchanged.

2. $(S[1] + S[S[1]]) \bmod 256 = p$
   This targets the cases where the first output byte of the key stream (denoted $o1$) will be dependent of known values and the value of our targeted key byte.

3. $Si[o1] \neq 1$ and 4. $Si[o1] \neq S[S[1]]$
   Because we want the values in $S[1]$ or $S[S[1]]$ not to be overwritten we set conditions 3 and 4.

As seen previously, these properties lead to two types of internal state for an attack on $K[3]$:

0	1	...	...
3	0	...	...

0	1	2	...
X	2	1	...

Let us have a look to a concrete example in order to understand it better. Assume we receive the following IV: $\boxed{3}\boxed{255}\boxed{1}$ and that we are looking for $\boldsymbol{K[3]}$ value (which is in our case equal to $\boldsymbol{Si[o1] - S[p] - j_{p-1} = Si[o1] - 7}$). If we follow the KSA, we will get the following moves for the internal state:

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 3$

1st step:

0	1	2	3	4	5	6	...
3	1	2	**0**	4	5	6	...

$i_1 = 1$, $j_1 = 3$ (recall that $255 + 1 = 0$)

2^{nd} step:

0	1	2	3	4	5	6	...
3	**0**	2	**1**	4	5	6	...

$i_2 = 2$, $j_2 = 6$

3^{rd} step:

0	1	2	3	4	5	6	...
3	0	**6**	1	4	5	**2**	...

$i_3 = 3$, $j_3 = 7 + K[3]$
(recall that we are looking for $K[3]$ value)

4^{th} step:

0	1	2	3	4	5	6	...
3	0	6	$\boldsymbol{S^3[7 + K[3]]}$	4	5	2	...

Where S^3 represents the internal state in the 3^{rd} step.

If we look at the outputted byte, while assuming that the rest of the KSA will not scramble the beginning of the internal state, we find that: $S[1] = 0$ and $S[0] = 3$ thus $o1 = S[S[1] + S[S[1]]] = S[3] = S^3[7 + K[3]]$. Therefore the attacked key byte can be computed as follow: $K[p] = Si[o1] - S[p] - j_{p-1}$, with $Si[o1]$ being the index of $o1$ in the internal state computed at step $p - 1$, $S[p]$ taken in the same internal state (step $p - 1$).

4.2 2^{nd} attack: Korek A_s13

Probability of success: $(\frac{254}{256})^{256-p} \approx 13.75\%$ $(p = 3)$

In this second attack, a trick is used in the PRGA. If we end up after the KSA with an internal state such as $S[1] = p$ and $S[p] = 0$, the PRGA will swap those values to output $o1 = p$. To achieve such a special case the following requirements are necessary:

1. $S[1] = p$
   This condition is to enable the PRGA to target for the element p in the internal state.

2. $o1 = p$
   By supposing that the first output byte will be p, we assume that after the PRGA swap the value p will be in position p, enforcing the supposition that j_p is being equal to the index of 0.

Again, let us have a look on a concrete example. Consider the following IV: $\boxed{6 \mid 252 \mid 193}$ with $K[3]$ as a target (in this example $\boldsymbol{K[3] = Si[0] - S[p] - j_{p-1} = 63}$). Then if we follow the KSA we will get:

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 6$

1st step:

0	1	2	3	4	5	6	...
6	1	2	3	4	5	**0**	...

$i_1 = 1$, $j_1 = 3$ (recall that $6 + 252 + 1 = 3$)

2nd step:

0	1	2	3	4	5	6	...
6	**3**	2	**1**	4	5	0	...

$i_2 = 2$, $j_2 = 198$

3rd step:

0	1	2	3	4	5	6	...
6	3	**198**	1	4	5	0	...

$i_3 = 3$, $j_3 = 199 + K[3]$
(recall that we are looking for $K[3]$ value)

Before trying to check what happens in the 4th step, we should analyze the behavior of the PRGA. In order to have $p(= 3)$ outputted as $o1$, while knowing that after the KSA $S[1] = p(= 3)$, we must have $S[p] = 0$:

$i = 1$ and $j = S[1] = 3$

0	1	2	3	4	5	6	...
6	3	198	0	4	5	1	...

1st step of PRGA: Recall that before the output byte is given a swap is made!

0	1	2	3	4	5	6	...
6	**0**	198	**3**	4	5	1	...

This condition will set $o1$ as
$S[S[1] + S[3]] = S[0 + 3] = S[3] = 3$.

Hence we know that $j_3 = Si[0]$ and more generally $j_p = Si[0]$. In our example this will lead to a value of $K[3]$ equal to $Si[0] - S[3] - j_2 = 6 - 1 - 198 = 63$ which can be generalized as $K[p] = Si[0] - S[p] - j_{p-1}$.

4.3 3rd attack: Korek A_u13_1

Probability of success: $(\frac{254}{256})^{256-p} \approx 13.75\%$ $(p = 3)$

In this attack the same trick is used as the one in the second attack (A_s13), with the following conditions:

1. $S[1] = p$
   We still target the p^{th} element in the internal state, which will be swapped with $S[1]$ at the beginning of the PRGA.

2. $o1 = ((1 - p) \bmod 256)$
   Now we are supposing that the first output byte will be $(1 - p)$ mod

256. Thus we assume that after the PRGA swap our target output will be in position 1, enforcing the supposition that j_p is being equal to the index of $((1-p) \bmod 256)$.

Consider the following IV: $\boxed{0}\boxed{2}\boxed{185}$ with $K[3]$ as a target (in this example $\boldsymbol{K[3] = Si[o1] - S[p] - j_{p-1} = 63}$). Then if we follow the KSA we will get:

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 0$

1st step:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_1 = 1$, $j_1 = 3$

2nd step:

0	1	2	3	4	5	6	...
0	**3**	2	**1**	4	5	6	...

$i_2 = 2$, $j_2 = 190$

3rd step:

0	1	2	3	4	5	6	...
0	3	**190**	1	4	5	6	...

$i_3 = 3$, $j_3 = 191 + K[3]$
(recall that we are looking for $K[3]$ value)

Now if we try to identify what is happening in the PRGA, we get to find the desired output $o1 = ((1-p) \bmod 256) = 254$. This value has to be set in $S[p] = S[3]$. Therefor when the swap will be made, $o1$ will be such as $S[S[1]+S[S[1]]] = S[p+1-p] = S[1] = ((1-p) \bmod 256)$. Let us see what it implies in our example:

1st step of PRGA:

Before the swap:
$i = 1$ and $j = S[1] = 3$

0	1	2	3	4	5	6	...
0	3	190	254	4	5	6	...

After the swap:

0	1	2	3	4	5	6	...
6	**254**	190	**3**	4	5	1	...

This condition will set $o1$ as
$S[S[1]+S[3]] = S[254+3] = S[1] = 254$

Hence we know that $j_3 = Si[o1] = Si[254]$ and more generally $j_p = Si[o1] = Si[(1-p) \bmod 256]$. In our example this will lead to a value of $K[3]$ equal to $Si[o1] - S[3] - j_2 = 254 - 1 - 190 = 63$ which can be generalized as $K[p] = Si[o1] - S[p] - j_{p-1}$.

4.4 4th attack: Korek A_u5_1

Probability of success: $(\frac{253}{256})^{256-p} \approx 5.07\%$ $(p = 3)$

In the previous attacks (A_u13_1), we tried to output a value that was directly dependent to the key byte searched. Now we enlarge this vision by saying that it is enough to output a value not directly dependent from the key if the process to output it depends on the key. Let us assume the following conditions:

1. $S[1] = p$
   We continue to target the p^{th} element for the PRGA.

2. $o1 \neq (1 - p) \bmod 256$
   We eliminate cases of the A_u13_1 attack.

3. $o1 \neq p$
   We also eliminate elements of the A_s13 attack.

4. $Si[o1] < p$
   The goal of this condition is to increase the chances that $Si[o1]$ will not change after the p^{th} step of KSA.

5. $Si[(Si[o1] - p) \bmod 256] \neq 1$
   This condition is protecting the integrity of $S[1] = p$, because if $j_p = Si[(Si[o1] - p) \bmod 256] = 1$ a swap will be made between $S[1]$ and $S[p]$ in the p^{th} step of the KSA. A similar case, where we suppose a precise value of j_p is illustrated with the attack A_u5_2.

We will look at two different example to illustrate the goal of these conditions. If we concentrate ourselves at looking for the first byte of the secret key ($\boldsymbol{K[3]}$, which is equal to $\boldsymbol{Si[(Si[o1]-p) \bmod 256] - S[p] - j_{p-1} = 64}$ in the first example), because of the fourth conditions we can predict that $o1$ will be either in position 0 or 2 in S. It cannot be in 1 because of the first condition.

If we start considering the case where $o1$ will be in position 0 (in S), we can for instance take the case of an IV equal to:

16	243	183

Then the KSA will be as follow:

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 16$
We see here that the output byte will be immediately set to its right position from the beginning.

0	1	2	3	4	5	6	...
16	1	2	3	4	5	6	...

1st step: $i_1 = 1$, $j_1 = 3$ (recall that $16 + 1 + 243 = 3$)
Note here that 3 (= p) will be
directly put into his desired index.

2nd step:

0	1	2	3	4	5	6	...
16	**3**	2	**1**	4	5	6	...

$i_2 = 2$, $j_2 = 188$

3rd step:

0	1	2	3	4	5	6	...
16	3	**188**	1	4	5	6	...

$i_3 = 3$, $j_3 = 189 + K[3]$
(recall that we are looking for $K[3]$ value)

The KSA done, we are interested to find what value has j_3 in order to find K[3]. We have set $o1$ as being in position 0 thus $o1$ is equal to 16. We also know from the PRGA that the outputted byte will be obtained from $S[S[i]+S[j]]$ after the swap between $S[i]$ and $S[j]$. Furthermore we have set as a condition that $S[1] = 3$ ($i = 1$) resulting in the swap between $S[1]$ and $S[3]$. Since the outputted byte is in position 0, the PRGA implies that the sum of $S[1]$ and $S[3]$ has also to be null, to enable the return of the desired value ($o1 = S[S[1] + S[3]]$). Therefore before the swap is done in PRGA, $S[3]$ must be equal to -3 mod 256 which is 253. Because $S[3]$ has to be equal to 253, and knowing that in the fourth step of KSA $j_3 = 189 + K[3]$ will be put in $S[3]$, we can conclude that $K[3] = 253 - 189 = 64$.

We are now tempted to refine this attack with the condition that $o1 = S[0] = K[0]$ (the first IV byte). However by fixing $o1 = S[0]$, we do not improve the attack as we are just taking a portion of it, and by fixing the value of $o1$ to $K[0]$ we are decreasing the performance of the attack as we limit the potential modification of $S[0]$ in the KSA if we are looking for a higher byte in the secret key. Of course in some cases these conditions will be highly efficient against selected keys.

In the second example we illustrate the case where the output byte is coming from $S[2]$, in order to show that fixing $o1 = S[0] = K[0]$ does not provide more advantages. Note that here again we are looking for $\boldsymbol{K[3]}$ (whose value is $\boldsymbol{Si[(Si[o1] - p) \bmod 256] - S[p] - j_{p-1} = 155}$). For this case, if we take the IV $\boxed{48 \mid 210 \mid 94}$ we can derive the KSA as such:

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 48$

0	1	2	3	4	5	6	...
48	1	2	3	4	5	6	...

1st step: $i_1 = 1$, $j_1 = 3$ (recall that $48 + 1 + 210 = 3$)
Here again we notice that 3 (= p) will be directly put into his desired index.

2nd step:

0	1	2	3	4	5	6	...
48	**3**	2	**1**	4	5	6	...

$i_2 = 2$, $j_2 = 99$

3rd step:

0	1	2	3	4	5	6	...
48	3	**99**	1	4	5	6	...

$i_3 = 3$, $j_3 = 100 + K[3]$
(recall that we are looking for $K[3]$ value)

At this stage we recognize a similar pattern to the first example. The output byte being set in position 2 implies that $S[1] + S[S[1]] = 2$. Having set $S[1]$ to 3 implies that $S[3]$ has to be equal to 255 in order to produce a sum of 2. Knowing that j_3 will be put in $S[3]$ in the 4th step of KSA, this gives us that $j_3 = 255$ and thus $K[3] = 255 - 100 = 155$.

We noticed in both cases that $j_1 = p$. This is done to meet the requirements of the first condition. However this should not be inserted as another condition since when looking for key bytes that are further away, the first condition can be coped in a later step depending of the key's nature.

Finally we can sum up the recovery of $K[3]$ with the following reasoning:
$Si[o1] = S[1] + S[S[1]] = p + S[j_p]$
which is equivalent to $j_p = Si[(Si[o1] - p) \bmod 256]$
and $K[p] = j_p - S[p] - j_{p-1} = Si[(Si[o1] - p) \bmod 256] - S[p] - j_{p-1}$.

4.5 5th attack: Korek A_u5_2

Probability of success: $(\frac{253}{256})^{256-p} \approx 5.07\%$ $(p = 3)$

This fifth attack highlights how much information we can extract by making assumptions of the value j_p. For instance we are going to suppose here that it takes the value 1, while the following conditions are met:

1. $Si[o1] = 2$
   We want the output byte being at index 2, because we are willing that $S[1]$ in the PRGA is equal to 1.

2. $S[p] = 1$
   By setting this condition and with the hypothesis of $j_p = 1$, we will get a swap between $S[1]$ and $S[p]$, which will set $S[1] = 1$ for the PRGA.

while looking for $\boldsymbol{K[3]}$ (equal to $\mathbf{1 - S[p] - j_{p-1} = 64}$ in this example). Let us see what this implies in practice by taking the IV $\boxed{0 \mid 2 \mid 187}$ This IV will influence the KSA as follow:

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 0$

1st step:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_1 = 1$, $j_1 = 3$
Note here that 1 will be directly put into $S[3]$ ($= S[p]$)

2nd step:

0	1	2	3	4	5	6	...
0	**3**	2	**1**	4	5	6	...

$i_2 = 2$, $j_2 = 192$

3rd step:

0	1	2	3	4	5	6	...
0	3	**192**	1	4	5	6	...

$i_3 = 3$, $j_3 = 193 + K[3]$

Recall that we are looking for $K[3]$ value and that we suppose that $j_3 = 1$, hence this attack with the corresponding IV will reveal $K[3]$ when it is equal to $1-193 = 64$. The following idea can be extended to $K[p] = 1-S[p]-j_{p-1}$.

Thus the 4th step of KSA will be:

0	1	2	3	4	5	6	...
0	**1**	192	**3**	4	5	6	...

We can now attest that $Si[o1] = 2$ because $o1 = S[S[1] + S[S[1]]] = S[1 + S[1]] = S[2]$ (no swap will be made because $S[1] = 1$). To summarize this attack, we set the value of $S[p]$ to 1, then we suppose that $j_3 = 1$ which will set the value 1 in $S[1]$, thus outputting $o1 = S[2]$.

4.6 6th attack: Korek A_u13_2

Probability of success: $(\frac{254}{256})^{256-p} \approx 13.75\%$ ($p = 3$)

In this attack, we will again suppose that $j_p = 1$ and we will expect the swap of the PRGA to fit our output. The conditions used for this attack are:

1. $S[p] = p$
   We set this condition in order to send the p value into $S[1]$ with the assumption that $j_p = 1$. This will enable the PRGA to select $S[1]$ and $S[p]$ for the first swap.

2. $S[1] = 0$
   Since we want the PRGA to output a controlled value, we set $S[1] = 0$ so that $S[1] + S[p] = p$ in the PRGA.

3. $o1 = p$
   We expect the output to be equal to p because this value swaps twice (from $S[p]$ to $S[1]$ in the p^{th} step of the KSA, then back to $S[p]$ with the first PRGA swap.

To illustrate how does it happen, while looking for $\boldsymbol{K[3]}$ (equal to $\mathbf{1 - S[p] - j_{p-1} = 64}$ in this example), let us take the case of the IV

4	255	184

When following the KSA we get:

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 4$

1^{st} step:

0	1	2	3	4	5	6	...
4	1	2	3	**0**	5	6	...

$i_1 = 1$, $j_1 = 4$ (remember that $4 + 255 + 1 = 4$)

2^{nd} step:

0	1	2	3	4	5	6	...
4	**0**	2	3	**1**	5	6	...

$i_2 = 2$, $j_2 = 190$

3^{rd} step:

0	1	2	3	4	5	6	...
4	0	**190**	3	1	5	6	...

$i_3 = 3$, $j_3 = 193 + K[3]$

We are now interested in the value of $K[3]$, while assuming that j_3 will take the value 1. Thus we can use the formula $K[p] = 1 - S[p] - j_{p-1}$ for our case: $K[3] = 1 - 193 = 64$. But let us continue the KSA and the PRGA to better understand why we make this assumption.

4^{th} step:

0	1	2	3	4	5	6	...
4	**3**	190	**0**	1	5	6	...

Hence we see that when we will enter the PRGA, $i = 1$ and $j = S[1] = 3$ (= p). After the PRGA swap, we will get the following internal state:

0	1	2	3	4	5	6	...
4	**0**	190	**3**	1	5	6	...

Thus $o1 = S[S[i] + S[j]] = S[S[1] + S[3]] = S[0 + 3] = S[3] = 3$.

To summarize, once $S[p]$ will be set as p and $S[1]$ to 0 at the beginning of the KSA, assuming that $j_p = 1$ means that we will swap $S[1]$ and $S[p]$.

However this swap will be done again in the PRGA, because $S[1]$ will be equal to p. Therefore $o1$ will be equal to $S[S[1] + S[p]] = S[0 + p] = p$.

Note that in several cases $K[1]$ will be equal to 255, but this is not always the case. This value is to enable the retrieval of the 0 in the swap at the 2^{nd} step of KSA. However this is not the only way to set $S[1] = 0$: if we take the IV $\begin{array}{|c|c|c|}\hline 2 & 5 & 249 \\ \hline\end{array}$ we will set $S[1] = 0$ in the 3^{rd} step of KSA instead of the 2^{nd} step (here $\boldsymbol{K[3]}$ is equal to $\mathbf{1 - S[p] - j_{p-1} = 253}$).

	0	1	2	3	4	5	6	...
Initialization:	0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 2$

	0	1	2	3	4	5	6	...
1st step:	**2**	1	**0**	3	4	5	6	...

$i_1 = 1$, $j_1 = 8$

	0	1	2	3	4	5	6	...
2nd step:	2	**8**	0	3	4	5	6	...

$i_2 = 2$, $j_2 = 1$ (remind that $8 + 249 + 0 = 1$)

	0	1	2	3	4	5	6	...
3rd step:	2	**0**	**8**	3	4	5	6	...

$i_3 = 3$, $j_3 = 4 + K[3] \equiv 1 \Rightarrow K[3] = 253$

4.7 7th attack: Korek A_u13_3

Probability of success: $(\frac{254}{256})^{256-p} \approx 13.75\%$ $(p = 3)$

This attack is very comparable to the A_u13_2, since we suppose that $j_p = 1$ and we will expect that the swap of the PRGA will set up our output as being in $S[1]$. The conditions used for this attack are:

1. $S[p] = p$
   Exactly as in A_u13_2, we want the value p to be in $S[p]$ at the beginning of the KSA.

2. $o1 = S[1]$
   In A_u13_2 we wanted $o1$ to be equal to p. Now we try to target the controlled value in $S[1]$.

3. $S[1] = (1 - p) \bmod 256$
   To be able to target $S[1]$ for $o1$, we need the sum of $S[1] + S[p] = 1$. Because we said we wanted $S[p]$ to be equal to p, it will require from us the above condition.

We will take the IV $\begin{array}{|c|c|c|}\hline 246 & 7 & 185 \\ \hline\end{array}$ to feed the KSA while looking for $\boldsymbol{K[3]}$ (equal to $\boldsymbol{1 - S[p] - j_{p-1} = 69}$ in this example):

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 246$

1st step:

0	1	2	3	4	5	6	...
246	1	2	3	4	5	6	...

$i_1 = 1$, $j_1 = 254$ (this will place the value $(1 - p)$ in $S[1]$)

2nd step:

0	1	2	3	4	5	6	...
246	**254**	2	3	4	5	6	...

$i_2 = 2$, $j_2 = 185$ (recall that $254 + 2 + 185 = 185$)

3rd step:

0	1	2	3	4	5	6	...
246	254	**185**	3	4	5	6	...

$i_3 = 3$, $j_3 = 188 + K[3]$

We then suppose that $j_3 = 1 = j_p$ in order to swap $S[3]$ $(= S[p])$ and $S[1]$. This will give us in the 4th step: $S[1] = 3 = p$ and $S[3] = 254 \equiv S[p] = (1 - p) \bmod 256$. If $S[1]$ and $S[3]$ $(= S[p])$ do not change in the rest of the KSA process, they will be presented to the PRGA which will initialize $j = S[1] = 3 = p$ and then swap $S[1]$ and $S[3]$ $(= S[p])$ to obtain the following internal state:

0	1	2	3	4	5	6	...
246	**254**	185	**3**	4	5	6	...

With this state we see easily that $o1 = S[S[1] + S[3]] = S[254 + 3] = S[1] = 254 \equiv S[S[1] + S[p]] = S[(1 - p) \bmod 256 + p] = S[1] = (1 - p) \bmod 256$. Note that by supposing that $j_p = 1$ we are looking for $K[p] = 1 - S[p] - j_{p-1}$ (in our case: $K[3] = 1 - 188 = 69$).

4.8 8th attack: Korek A_u5_3

Probability of success: $(\frac{253}{256})^{256-p} \approx 5.07\%$ $(p = 3)$

This attack is an extension of A_u13_2 and A_u13_3, hence uses exactly the same concepts. In A_u13_3 we have fixed $o1$ in $S[1]$ with the value $(1-p)$ mod 256, thus we wanted $S[1]$ and $S[p]$ to remain unchanged. Here $o1$ is not set in $S[1]$, however we still want that $Si[o1]$, $S[1]$ and $S[p]$ are kept unchanged (reducing the probability of success). Let us have a look at the conditions:

1. $S[p] = p$
   We keep the condition of $S[p]$ containing the value p, in order to set it in $S[1]$ later with $j_p = 1$ as assumption.

2. $S[1] \geqslant (-1 \cdot p) \bmod 256$
   In A_u13_2 we wanted to have a controlled value in $S[p]$ for $o1$. In A_u13_3, the controlled value was $S[1]$. Now we are looking for any controlled value which index is bellow p. Knowing that $0 \leqslant (S[1] + S[p]) \bmod 256 = Si[o1] < p$ (we do not take into account $Si[o1] = p$ otherwise we will fall back to A_u13_2), we can thus derive $-p \leqslant S[1] < 0$ which gives us the above condition.

3. $S[1] = (Si[o1] - p) \bmod 256$
   Because $(S[1] + S[p]) \bmod 256 = (S[1] + p) \bmod 256 = Si[o1]$, the above condition is explained.

4. $Si[o1] \neq 1$
   Finally we need to eliminate the A_u13_3 cases, and this is done with this condition (not present in WepLab, but included in AirCrack)

If we focus on $Si[o1]$ while looking for $\boldsymbol{K[3]}$ (equal to $\boldsymbol{1 - S[p] - j_{p-1} = 205}$ in this example), we see that $Si[o1]$ can be either equal to 0 or 2 (1 being forbidden). For the purpose of our example we will chose $Si[o1] = 0$ with the following IV $\boxed{12}\boxed{240}\boxed{50}$ while keeping in mind that the reasoning behind it is identical when $Si[o1] = 2$. Thus the KSA will be started as such:

	0	1	2	3	4	5	6	...
Initialization:	0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 12$

	0	1	2	3	4	5	6	...
1st step:	**12**	1	2	3	4	5	6	...

$i_1 = 1$, $j_1 = 253$
($S[1]$ will then receive the value $(Si[o1] - p) \bmod 256$)

	0	1	2	3	4	5	6	...
2nd step:	12	**253**	2	3	4	5	6	...

$i_2 = 2$, $j_2 = 49$ (remember that $253 + 2 + 50 = 49$)

	0	1	2	3	4	5	6	...
3rd step:	12	253	**49**	3	4	5	6	...

$i_3 = 3$, $j_3 = 52 + K[3]$

We then suppose that $j_3 = 1 = j_p$ in order to swap $S[3]$ $(= S[p])$ and $S[1]$. This will give us in the 4th step: $S[1] = 3 = p$ and $S[3] = 253 \equiv S[p] = (Si[o1] - p) \bmod 256$. If $S[1]$, $S[3]$ $(= S[p])$ and $Si[o1]$ do not change in the rest of the KSA, they will be presented to the PRGA which will initialize $j = S[1] = 3 = p$ and then swap $S[1]$ and $S[3]$ ($= S[p]$) to obtain the following internal state:

0	1	2	3	4	5	6	...
12	**253**	49	**3**	4	5	6	...

With this state we see easily that $o1 = S[S[1] + S[3]] = S[253 + 3] = S[0] = 12 \equiv S[S[1] + S[p]] = S[(Si[o1] - p) \bmod 256 + p] = S[Si[o1]]$. Note that by supposing that $j_p = 1$, we are looking for $K[p] = 1 - S[p] - j_{p-1}$ (in our case: $K[3] = 1 - 188 = 69$).

4.9 9^{th} attack: Korek A_s3

Probability of success: $(\frac{253}{256})^{256-p} \approx 5.13\%$ ($p = 4$)

With this attack we start the second group of attacks. These last ones include the knowledge of $o2$ in their concept. Thus while this attack is highly based on the FMS attack, it extends it in the sense that we do not extract information only from $o1$ but also from the second output byte stream $o2$. It has first been presented by h1kary and then implemented by Topo[LB]. The IV selected gives us internal states in the KSA that respect these constraints:

1. $S[1] \neq 2$
   If $S[1]$ equals 2 then we will loose, in the first swap of the PRGA, the value that $S[2]$ contains. Moreover the first PRGA j would be equal to 2. Thus the second one would be equal to 2 plus the value contained in $S[2]$ which has been set to 2. Hence we will no more have $o2$ equal to $S[4]$ ($S[p]$).

2. $S[2] \neq 0$
   We avoid the situation where the first two j are identical in the PRGA (remind that the second j is equal to the previous one plus $S[2]$), because $o2$ will be forced to 0:
   In the first step of the PRGA, $S[1] = \alpha$ thus $j = \alpha$. This will set α into $S[\alpha]$. In the second step, if we have $S[2] = 0$, j will not change its value. After the swap will be done we will have $o2 = S[0 + \alpha] = 0$.

3. $S[2] + S[1] < p$
   With the previous conditions, this one limits the value of the j in the second step of the PRGA. We target a box bellow the index p to be able to predict with a higher probability the value of $S[j]$.

4. $(S[2] + S[S[2] + S[1]]) \bmod 256 = p$
   Here we are clearly stating that we want the value that has been put in $S[p]$ at the beginning of the PRGA to be sent as $o2$, since $o2 = S[(S[2] + S[S[2] + S[1]]) \bmod 256]$.

5. $Si[o2] \neq 1$ and $Si[o2] \neq 2$ and $Si[o2] \neq S[1] + S[2]$
   Knowing that we are using the elements with indexes 1, 2 and $S[1] + S[2]$, we drop the cases where $o2$ happens to uses them as indexes, because this would mean that there value were modified.

If we try to identify the internal states that are selected for an attack on $K[4]$, this gives us the following two possibilities:

0	1	2	3	...
X	0	3	1	...

0	1	2	...	...
X	0	2	...	...

If we choose to identify the value of $K[4]$ according to the first scheme of internal state, we find a comparable behavior as for the 1st case (here $\boldsymbol{K[4] = Si[o2] - S[p] - j_{p-1} = Si[o2] - 10}$). Assuming the following IV $\boxed{3}\boxed{255}\boxed{253}$ with $K[3] = 3$ we get:

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 3$

1st step:

0	1	2	3	4	5	6	...
3	1	2	**0**	4	5	6	...

$i_1 = 1$, $j_1 = 3$ (recall that $255 + 1 = 0$)

2nd step:

0	1	2	3	4	5	6	...
3	**0**	2	**1**	4	5	6	...

$i_2 = 2$, $j_2 = 2$ (remember that $3 + 253 + 2 = 2$)

3rd step:

0	1	2	3	4	5	6	...
3	0	**2**	1	4	5	6	...

$i_3 = 3$, $j_3 = 6$

4th step:

0	1	2	3	4	5	6	...
3	0	2	**6**	4	5	**1**	...

$i_4 = 4$, $j_4 = 10 + K[4]$
(remind that we are looking for $K[4]$ value)

5th step:

0	1	2	3	4	5	6	...
3	0	2	6	$\boldsymbol{S^4[10 + K[4]]}$	5	1	...

Where S^4 represents the internal state in the 4th step.

Now let us have a look at how the PRGA will react:

1st step of PRGA:

$i = 1$ and $j = S[1] = 0$

0	1	2	3	4	5	6	...
0	**3**	2	?	$S^4[10 + K[4]]$	?	?	...

(The first output byte has a high chance to be $o1 = S[3]$, if the value in $S[0]$ has not changed)

2nd step of PRGA:

$i = 2$ and $j = S[2] = 2$

0	1	2	3	4	5	6	...
0	3	**2**	?	$S^4[10 + K[4]]$	?	?	...

The second output byte will be $o2 = S[S[2] + S[S[2] + 0]] = S[4] = S^4[10 + K[4]]$. Thus we can recover $K[4] = Si[o2] - S[4] - j_3$, and more generally any $K[p] = Si[o2] - S[p] - j_{p-1}$. Note that $Si[o2]$ is the index of $o2$ in the internal state computed at step p (step 4 in our example), and that $S[p]$ is taken in the same internal state (step p).

4.10 10th attack: Korek A_u15

Probability of success: $(\frac{254}{256})^{256-p} \approx 13.75\%$ $(p = 3)$

While looking for the information provided by the knowledge of $o2$, a greater consideration has to be taken for the behavior of the PRGA. First there is a swap before computing each output, then $j = j + S[i]$ and thus while the first j was simply equal to $S[1]$, the second one will be equal to the previous one plus $S[2]$.

Now if we manage to have in the second step of the PRGA $S[2] = 0$, we will preserve the value of j. Let us see in more detail what this implies:

1st step of PRGA:

Before the swap:

0	1	2	3	...	α	...
?	α	?	?	...	β	...

$i = 1$ and $j = S[1] = \alpha$

After the swap:

0	1	2	3	...	α	...
?	$\boldsymbol{\beta}$	?	?	...	$\boldsymbol{\alpha}$	...

This condition will put α in $S[\alpha]$

2^{nd} step of PRGA:

Before the swap:

0	1	2	3	...	α	...
?	β	**0**	?	...	α	...

$i = 2$ and $j = \alpha + S[2] = \alpha$
As said previously to keep j equals to α we need $S[2]$ equal to 0

After the swap:

0	1	2	3	...	α	...
?	β	$\boldsymbol{\alpha}$	?	...	**0**	...

With this case we can see that

$$\begin{aligned} o2 &= S[S[2] + S[\alpha]] \\ &= S[\alpha + 0] \\ &= S[\alpha] = 0. \end{aligned}$$

Therefore if we feed the PRGA with $S[2] = 0$, $o2$ will be equal to 0. To exploit this situation we only need for one value to remain unchanged during the unknown steps of the KSA. To meet these requirements the 10^{th} attack suppose that $j_3 = 2$ and uses these conditions:

1. $o2 = 0$
   We identify cases where $o2$ is equal to 0.

2. $S[p] = 0$
   We want this condition in order to be able to set the 0 in $S[2]$ while assuming $j_p = 2$ in the KSA.

3. $S[2] \neq 0$
   This condition is useless because the 0 has to be set in $S[p]$, p being greater or equal to 3. It is used in AirCrack as an execution optimization, while WepLab uses it blindly.

Let us take the IV $\boxed{3 \mid 10 \mid 64}$ to look after $\boldsymbol{K[3]}$ (equal to $\boldsymbol{2 - S[p] - j_{p-1} = 178}$ in our case) in order to understand how $S[2]$ will receive the value 0.

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 3$

Even though here the 0 is directly put into the right place, formally $S[p]$, it cannot be taken as a condition. The placement of 0 in $S[p]$ could be done in a later phase.

1st step:

0	1	2	3	4	5	6	...
3	1	2	**0**	4	5	6	...

$i_1 = 1$, $j_1 = 14$

2nd step:

0	1	2	3	4	5	6	...
3	**14**	2	0	4	5	6	...

$i_2 = 2$, $j_2 = 80$

3rd step:

0	1	2	3	4	5	6	...
3	14	**80**	0	4	5	6	...

$i_3 = 3$, $j_3 = 80 + K[3]$

We are now interested in the value of $K[3]$, while assuming that j_3 (j_p) will take the value 2. Thus we can use the formula $K[p] = 2 - S[p] - j_{p-1}$ to compute $K[3]$: $K[3] = 2 - 80 = 178$. But let us continue the KSA to understand why we make this assumption.

4th step:

0	1	2	3	4	5	6	...
3	14	**0**	**80**	4	5	6	...

Thus the 4th step goal is to set the 0 in $S[2]$. For the rest of the KSA we just need that the $Si[0]$ ($= 2$) is kept unchanged to feed the PRGA with the following internal state (this will bring us back to our above explanation of the PRGA behavior):

0	1	2	3	...	α	...
?	α	**0**	?	...	β	...

4.11 11th attack: Korek A_s5_2

Probability of success: $(\frac{253}{256})^{256-p} \approx 5.07\%$ ($p = 3$)

With this attack, another way to exploit the knowledge of $o2$ is shown to us. To be able to extract a good amount of information, we need to determine the value that j will take in the second step of the PRGA. Before going through a concrete example, let us check the conditions for the A_s5_2 attack:

1. $S[1] > p$
   First we select the IVs that set into $S[1]$ a value strictly greater than p, in order to ensure the PRGA to not modify boxes with index lower or equal to p during its first step.

2. $(S[2] + S[1]) \bmod 256 = p$
   Then we are willing that the j in the second step of the PRGA takes the value p, in order to include what has been put in $S[p]$ in the computation process of the KSA.

3. $o2 = S[1]$
   Now, we are checking the cases where it is $S[1]$ that is outputted as $o2$. This condition enables us to identify the assumption made on the value of j_p in the KSA: $j_p = Si[(S[1] - S[2]) \bmod 256]$.

4. $Si[(S[1] - S[2]) \bmod 256] \neq 1$ and $Si[(S[1] - S[2]) \bmod 256] \neq 2$
   We do not want j_p to neither take the values 1 or 2, in order to preserve their content, and output $S[1]$.

The idea behind this attack is to output a controlled value at a particular index ($o2 = S[1]$, where $S[1]$ is set in place at the beginning of the KSA). In order to do that, $S[1]$, $S[2]$ and $S[p]$ must remain unchanged after the p^{th} step of the KSA, to let the PRGA deal with the subtilities. In the first step of it (PRGA), j will take the value that $S[1]$ has (i being equal to 1). According to the protocol a swap will be made between $S[1]$ and $S[S[1]]$, thus putting the value contained in $S[1]$ in the index $S[1]$.

1st step of PRGA:

Before the swap:

0	1	2	...	p	...	α	...
?	α	β	...	$(\alpha-\beta)$	...	γ	...

$i = 1$ and $j = S[1] = \alpha$

After the swap:

0	1	2	...	p	...	α	...
?	$\boldsymbol{\gamma}$	β	...	$(\alpha-\beta)$	...	$\boldsymbol{\alpha}$	...

α is set in $S[\alpha]$

Once this done, the second step of the PRGA will set i to 2, and will add $S[2]$ to the previous j. Here, with the preceding conditions we see that j will be equal to p ($j = S[1] + S[2]$ where $S[1]$ is taken from the previous step). Therefore $o2$ will be computed as follow: $o2 = S[S[p] + S[2]]$. We know that we look for $o2 = S[1]$ and that this check is made for the value that is now in the index $S[1]$. Thus if we keep the notation according to the internal state that we can compute (step p of the KSA), we have $o2 = S[S[j_p] + S[2]] = S[S[1]]$, which is equivalent to $j_p = Si[(S[1] - S[2]) \bmod 256]$

2nd step of PRGA:

Before the swap:

0	1	2	...	p	...	α	...
?	γ	β	...	$(\alpha-\beta)$	...	α	...

$i = 2$ and $j = \alpha + S[2] = \alpha + \beta = p$
(As according to our condition:
$(S[2] + S[1]) \bmod 256 = p$)

After the swap:

0	1	2	...	p	...	α	...
?	γ	$(\alpha-\beta)$	...	$\boldsymbol{\beta}$	...	α	...

$$\begin{aligned} o2 &= S[S[2] + S[p]] = S[\alpha - \beta + \beta] \\ &= S[\alpha] = \alpha \end{aligned}$$
(As expected: $o2 = S[1]$)

As we know, $K[p]$ can be derived from j_p: $K[p] = j_p - S[p] - j_{p-1}$ (values taken from the p^{th} step of the KSA). We have then the general formula as $\boldsymbol{K[p] = Si[(S[1] - S[2]) \bmod 256] - S[p] - j_{p-1}}$. In our example we will look for the value $\boldsymbol{K[3]}$ which has the value **178**. We select the IV

248	240	47

and analyze how the KSA behave.

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 248$

1st step:

0	1	2	3	4	5	6	...
248	1	2	3	4	5	6	...

$i_1 = 1$, $j_1 = 233$ (recall that $248 + 240 + 1 = 233$)

2nd step:

0	1	2	3	4	5	6	...
248	**233**	2	3	4	5	6	...

$i_2 = 2$, $j_2 = 26$ (remember that $233 + 47 + 2 = 26$)

3rd step:

0	1	2	3	4	5	6	...
248	233	**26**	3	4	5	6	...

$i_3 = 3$, $j_3 = 29 + K[3]$
(recall that we are looking for $K[3]$ value)

4th step:

0	1	2	3	4	5	6	...
248	233	26	$\boldsymbol{S^3[29 + K[3]]}$	4	5	6	...

Where S^3 represents the internal state in the 3rd step.

Once we have arrived to this step, we assume that $S[1]$, $S[2]$ and $S[3]$ ($S[p]$) will remain unchanged during the following steps of the KSA, until we reach the PRGA:

1st step of PRGA:
$i = 1$ and $j = S[1] = 233$

0	1	2	3	4	...	233	...
?	**?**	26	$S^3[29 + K[3]]$	?	...	**233**	...

We do not pay attention to the first output byte.

2nd step of PRGA:
$i = 2$ and $j = 233 + S[2] = 233 + 26 = 3\ (= p)$

0	1	2	3	4	...	233	...
?	?	$\boldsymbol{S^3[29 + K[3]]}$	**26**	?	...	233	...

Because we know that $o2 = 233$ we can conclude that $26 + S^3[29 + K[3]] = 233$, which gives us $K[3] = Si^3[233-26]-29 = Si^3[207]-29 = 207-29 = 178$ (recall that in the third step of the KSA, the index $Si[207]$ has not been modified yet).

4.12 12th attack: Korek A_s5_3

Probability of success: $(\frac{253}{256})^{256-p} \approx 5.07\%$ $(p = 3)$

In this attack we set our conditions such as the value $S[j_p]$ will be outputted in $o2$. To be able to do this, we use the following conditions:

1. $S[1] > p$
   As previously (A_s5_2), we do not want the PRGA to change values of index bellow p, thus we need to filter cases where the KSA will put a value greater than p in $S[1]$.

2. $(S[2] + S[1]) \bmod 256 = p$
   Again, we want j in the second step of the PRGA to be equal to p.

3. $o2 = (2 - S[2]) \bmod 256$
   The idea behind this condition, starts with the supposition of the value j_p. Moreover, we are analyzing the cases where j_p is equal to $(2 - S[2])$ mod 256, which is a value that will be set in $S[p]$. According to our previous condition, once in the second step of the PRGA, the value contained in $S[p]$ will be called and swapped with $S[2]$ $(j = p)$. Then the sum of $S[2]$ and $S[p]$ will give the index of the box outputted by o2. As we have set $S[p] = (2 - S[2]) \bmod 256$, $o2$ will then output $S[2]$. Note that $S[2]$ has changed value due to the swap, and now contains what $S[p]$ contained, in other words $(2 - S[2]) \bmod 256$.

4. $Si[o2] \neq 1$ and $Si[o2] \neq 2$
   In order to achieve this particular sequence of events, $o2$ should not output $S[1]$ neither $S[2]$, which means that $S[1]$ should not contain

$(2 - S[2]) \bmod 256$, and $S[2]$ should not be equal to 1 nor 129 (cases where $S[2] = (2 - S[2]) \bmod 256$).

Before presenting a concrete example, we will remind how the PRGA behave to understand the sequence that is required. The beginning of the PRGA is identical to the attack A_s5_2: we set the value in $S[1]$ at the index $S[1]$.

1st step of PRGA:

Before the swap:

0	1	2	...	p	...	α	...
?	α	β	...	$(2-\beta)$	...	γ	...

$i = 1$ and $j = S[1] = \alpha$

After the swap:

0	1	2	...	p	...	α	...
?	$\boldsymbol{\gamma}$	β	...	$(2-\beta)$	...	$\boldsymbol{\alpha}$	...

α is set in $S[\alpha]$

It is during the second step that we are able to understand fully the condition made on $o2$. j is increased by the value contained in $S[2]$, setting him to the value p according to our conditions. Then $S[2]$ and $S[p]$ are swapped, and the sum of their value gives us the index of the outputted byte. We see clearly that the condition made on $o2$ implies that $S[p]$ contains the value $(2-S[2]) \bmod 256$. This requirement made on $S[p]$ explains us the supposition made on the value of j_p, giving us a way to compute $K[p]$: $\boldsymbol{K[p] = Si[(2 - S[2]) \bmod 256] - S[p] - j_{p-1}}$.

2nd step of PRGA:

Before the swap:

0	1	2	...	p	...	α	...
?	γ	β	...	$(2-\beta)$	...	α	...

$i = 2$ and $j = \alpha + S[2] = \alpha + \beta = p$
(As according to our condition:
$(S[2] + S[1]) \bmod 256 = p$)

After the swap:

0	1	2	...	p	...	α	...
?	γ	$\boldsymbol{(2-\beta)}$	...	$\boldsymbol{\beta}$	...	α	...

$$\begin{aligned} o2 &= S[S[2] + S[p]] = S[2 - \beta + \beta] \\ &= S[2] = 2 - \beta \end{aligned}$$

(As expected: $o2 = (2 - S[2]) \bmod 256$)

For our example we search the value of $\boldsymbol{K[3]}$ (equal to **157** in our example), while taking into consideration the IV $\begin{array}{|c|c|c|}\hline 96 & 113 & 93 \\ \hline\end{array}$ The KSA will be as such:

Initialization:

0	1	2	3	4	5	6	...
0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 96$

1^{st} step:

0	1	2	3	4	5	6	...
96	1	2	3	4	5	6	...

$i_1 = 1$, $j_1 = 210$

2^{nd} step:

0	1	2	3	4	5	6	...
96	**210**	2	3	4	5	6	...

$i_2 = 2$, $j_2 = 49$ (recall that $210 + 93 + 2 = 49$)

3^{rd} step:

0	1	2	3	4	5	6	...
96	210	**49**	3	4	5	6	...

$i_3 = 3$, $j_3 = 52 + K[3]$
(recall that we are looking for $K[3]$ value)

4^{th} step:

0	1	2	3	4	5	6	...
96	210	49	$\boldsymbol{S^3[52 + K[3]]}$	4	5	6	...

Where S^3 represents the internal state in the 3^{rd} step.

Again, here we want that $S[1]$, $S[2]$ and $S[3]$ ($S[p]$) remain unchanged during the following steps of the KSA, until we reach the PRGA:

1^{st} step of PRGA:
$i = 1$ and $j = S[1] = 210$

0	1	2	3	4	...	210	...
?	**?**	49	$S^3[52 + K[3]]$	?	...	**210**	...

We do not take into account the first output byte.

2^{nd} step of PRGA:
$i = 2$ and $j = 210 + S[2] = 210 + 49 = 3$ $(= p)$

0	1	2	3	4	...	210	...
?	?	$\boldsymbol{S^3[52 + K[3]]}$	**49**	?	...	210	...

Because we expect $o2 = S[S[2] + S[3]] = S[S^3[52 + K[3]] + 49] = S[2]$, we conclude that $K[3] = Si^3[2 - 49] - 52 = 209 - 52 = 157$ (recall that in the third step of the KSA, the index $Si[209]$ has not been modified yet).

4.13 13^{th} attack: Korek A_4_s13

Probability of success: $(\frac{254}{256})^{256-p} \approx 13.85\%$ $(p = 4)$

In the following three attacks (A_4_s13, A_4_u5_1 and A_4_u5_2), we are going to limit ourselves to the search of $K[4]$ value, hence $p = 4$ will be one of our condition. We will also present a new attack (A_4_s5_1) that obeys to this limitation. In these attacks, not only we will force ourselves to search $K[4]$ value, but we will also assume that $S[1] = 2$ in order to catch the value of $S[4]$ in the computation process of $o2$. The difference present between these four attacks are the assumption made on j_4 value in the KSA, and thus the value that $o2$ will have. (Notice that we will not use the term p as it is equal to 4).

In this attack, we are assuming that j_4 will take the value of the index of 0 in order to place 0 into $S[4]$. With this assumption, $o2$ will inherit the value 0. Let us see the conditions for this attack:

1. $p = 4$
   As said previously, we are looking for the value of $K[4]$.

2. $S[1] = 2$
   By setting the value 2 in $S[1]$ we will set during the first step of the PRGA this value into $S[2]$. Hence when starting the second step of the PRGA the internal state will be such as $S[2] = 2$, with a previous j equal to 2. This means that the next j will be equal to 4, swapping the values of $S[2]$ and $S[4]$.

3. $o2 = 0$
   Once the swap of the second step of the PRGA has been done, the index of the outputted byte will be equal to the sum $S[2]+S[4]$ (recall that after the swap $S[4] = 2$). Therefore the assumption made on the value j_4 will tell us what $o2$ will be. Here we selected $o2 = 0$ which can be obtained if $S[2] = 0$. The index of $o2$ will be equal to $0+2 = 2$ and $S[2]$ will contain the desired 0 value.

4. $S[4] \neq 0$ (equivalent to $S[p] \neq 0$)
   In order to avoid redundancy with the attack A_u15, we need to put aside cases where $S[p] = 0$ (in our case $p = 4$). This condition has not been taken into account by both AirCrack and WepLab.

In order to achieve this attack, we suppose that $j_4 = Si[0]$, hence we are looking for values of $\boldsymbol{K[4]}$ equal to $\boldsymbol{Si[0] - S[4] - j_3}$. This assumption will put 0 in $S[4]$ during the 5^{th} step of the KSA. Therefore when entering in the PRGA, $S[4]$ will be equal to 0 (note the similarity with the attack A_u15). In the second step of the PRGA, $o2$ will be equal to $S[S[2]+S[4]] = S[0+2] = S[2] = 0$ which corresponds to our condition. To understand the previous equation, we will look at the sequence of events in the PRGA:

1^{st} step of PRGA:

Before the swap:

0	1	2	3	4	...
?	2	α	?	0	...

$i = 1$ and $j = S[1] = 2$

After the swap:

0	1	2	3	4	...
?	$\boldsymbol{\alpha}$	**2**	?	0	...

The value 2 is set in $S[2]$

2^{nd} step of PRGA:

Before the swap:

0	1	2	3	4	...
?	α	2	?	0	...

$i = 2$ and $j = 2 + S[2] = 4$

After the swap:

0	1	2	3	4	...
?	α	**0**	?	**2**	...

$o2 = S[0 + 2] = 0$

Now if we consider the IV $\boxed{164 \mid 93 \mid 113}$ with $K[3] = 207$, and we look for the value of $\boldsymbol{K[4]}$ (equal to $\boldsymbol{Si[0] - S[4] - j_3 = 90}$ in our example), we get the following for the KSA:

Initialization:

0	1	2	3	4	...	164	...
0	1	2	3	4	...	164	...

$i_0 = 0$, $j_0 = 164$

1^{st} step:

0	1	2	3	4	...	164	...
164	1	2	3	4	...	**0**	...

$i_1 = 1$, $j_1 = 2$ (recall that $164 + 93 + 1 = 2$)

2^{nd} step:

0	1	2	3	4	...	164	...
164	**2**	**1**	3	4	...	0	...

$i_2 = 2$, $j_2 = 116$

3^{rd} step:

0	1	2	3	4	...	164	...
164	2	**116**	3	4	...	0	...

$i_3 = 3$, $j_3 = 70$

4^{th} step:

0	1	2	3	4	...	164	...
164	2	116	**70**	4	...	0	...

$i_4 = 4$, $j_4 = 74 + K[4] = 164$
(recall that we are looking for $K[4]$ value and that we are assuming that $j_4 = Si[0]$)

5^{th} step:

0	1	2	3	4	...	164	...
164	2	116	70	**0**	...	**4**	...

From the above calculation we can easily derive that $K[4] = 164 - 74 = 90$ and this comes from our assumption of $j_4 = Si[0] = 164$. Now if we continue with the PRGA we get:

1st step of PRGA:

Before the swap:

0	1	2	3	4	...
?	2	α	?	0	...

$i = 1$ and $j = S[1] = 2$

After the swap:

0	1	2	3	4	...
?	$\boldsymbol{\alpha}$	**2**	?	0	...

The value 2 is set in $S[2]$.

2nd step of PRGA:

Before the swap:

0	1	2	3	4	...
?	α	2	?	0	...

$i = 2$ and $j = 2 + S[2] = 4$

After the swap:

0	1	2	3	4	...
?	α	**0**	?	**2**	...

Hence $o2 = S[0 + 2] = 0$

4.14 14th attack: Korek A_4_u5_1

Probability of success: $(\frac{253}{256})^{256-p} \approx 5.13\%$ $(p = 4)$

We are going to limit ourselves to the search of $K[4]$ value ($p = 4$ is one of our condition, thus we will not use the term p as it is equal to 4). We will also assume that $S[1] = 2$ in order to catch the value of $S[4]$ in the computation process of $o2$. In this attack, we are assuming that j_4 will take the value of the index of 254 in order to place 254 into $S[4]$. With this assumption, $o2$ will inherit the value in the index 0. Let us see how this is possible and what are the necessary conditions for this attack:

1. $p = 4$
   As said previously, we are looking for the value of $K[4]$.

2. $S[1] = 2$
   By setting the value 2 in $S[1]$ we will set during the first step of the PRGA this value into $S[2]$. Hence when starting the second step of the PRGA the internal state will be such as $S[2] = 2$, with a previous j equal to 2. This means that the next j will be equal to 4, swapping the values of $S[2]$ and $S[4]$.

3. $o2 \neq 0$
   We are no longer targeting a null $o2$, to avoid redundancy with A_4_s13.

4. $Si[o2] = 0$
   Now we want to output $S[0]$ as $o2$. This can be achieved with the assumption $j_4 = 254$, due to the fact that $S[4]$ will receive the value 254 and that it will be swapped during the second step of the PRGA with $S[2]$ (containing the value 2). This will set $o2$ as $S[S[2] + S[4]] = S[254 + 2] = S[0]$.

5. $j_1 = 2$
   j_1 is the value of j in the 1st step of the KSA. This constraint condition the way we set the value 2 in $S[1]$. This is done in order to restrain false positive and to reduce the search space.

To achieve this attack, we suppose that $j_4 = Si[254]$, hence we are looking for values of $\boldsymbol{K[4]}$ equal to $\boldsymbol{Si[254] - S[4] - j_3}$. . This assumption will set the value 254 in $S[4]$ during the 5th step of the KSA. Therefore when entering in the PRGA, $S[4]$ will be equal to 254. In the second step of the PRGA, $o2$ will be then equal to $S[S[2] + S[4]] = S[254 + 2] = S[0]$ which corresponds to our condition. To understand the previous equation, we will look at the sequence of events in the PRGA:

1st step of PRGA:

Before the swap:

0	1	2	3	4	...
α	2	β	?	254	...

$i = 1$ and $j = S[1] = 2$

After the swap:

0	1	2	3	4	...
α	$\boldsymbol{\beta}$	**2**	?	254	...

the value 2 is set in $S[2]$

2nd step of PRGA:

Before the swap:

0	1	2	3	4	...
α	β	2	?	254	...

$i = 2$ and $j = 2 + S[2] = 4$

After the swap:

0	1	2	3	4	...
α	β	**254**	?	**2**	...

$o2 = S[254 + 2] = S[0]$

Now if we consider the IV $\boxed{239 \mid 18 \mid 167}$ with $K[3] = 161$, and we look for the value of $\boldsymbol{K[4]}$ (equal to $\boldsymbol{Si[254] - S[4] - j_3 = 172}$ in our example), we get the following for the KSA:

Initialization:

0	1	2	3	4	...
0	1	2	3	4	...

$i_0 = 0$, $j_0 = 239$

1st step:

0	1	2	3	4	...
239	1	2	3	4	...

$i_1 = 1$, $j_1 = 2$ (notice that as required $239 + 18 + 1 = 2$)

2nd step:

0	1	2	3	4	...
239	**2**	**1**	3	4	...

$i_2 = 2$, $j_2 = 170$

3rd step:

0	1	2	3	4	...
239	2	**170**	3	4	...

$i_3 = 3$, $j_3 = 78$
(recall that $170 + 161 + 3 = 78$)

4th step:

0	1	2	3	4	...
239	2	170	**78**	4	...

$i_4 = 4$, $j_4 = 82 + K[4] = 254$
(recall that we are looking for $K[4]$ value
and that we are assuming that $j_4 = Si[254]$)

5th step:

0	1	2	3	4	...
239	2	170	78	**254**	...

We can thus derive $K[4]$ as being equal to $254 - 82 = 172$ due to the assumption made on the value of j_4. The behavior of the PRGA shows us the importance of such hypothesis.

1st step of PRGA:

Before the swap:

0	1	2	3	4	...
α	2	β	?	254	...

$i = 1$ and $j = S[1] = 2$

After the swap:

0	1	2	3	4	...
α	$\boldsymbol{\beta}$	**2**	?	254	...

The value 2 is set in $S[2]$.

2nd step of PRGA:

Before the swap:

0	1	2	3	4	...
α	β	2	?	254	...

$i = 2$ and $j = 2 + S[2] = 4$

After the swap:

0	1	2	3	4	...
α	β	**254**	?	**2**	...

Hence $o2 = S[254 + 2] = S[0] = \alpha$

4.15 15th attack: Korek A_4_u5_2

Probability of success: $(\frac{253}{256})^{256-p} \approx 5.13\%$ ($p = 4$)

We are going to limit ourselves to the search of $K[4]$ value ($p = 4$ is one of our condition, thus we will not use the term p as it is equal to 4). We will also assume that $S[1] = 2$ in order to catch the value of $S[4]$ in the computation process of $o2$. In this attack, we are assuming that j_4 will take the value of the index of 255 in order to place 255 into $S[4]$. With this assumption, $o2$ will inherit the value in the index 1 (which in fact has been modified to receive the value of $S[2]$). Let us see how this is possible and what are the necessary conditions for this attack:

1. $p = 4$
   As said previously, we are looking for the value of $K[4]$.

2. $S[1] = 2$
   By setting the value 2 in $S[1]$ we will set during the first step of the PRGA this value into $S[2]$. Hence when starting the second step of the PRGA the internal state will be such as $S[2] = 2$, with a previous j equal to 2. This means that the next j will be equal to 4, swapping the values of $S[2]$ and $S[4]$.

3. $o2 \neq 0$
   We are no longer targeting a null $o2$, to avoid redundancy with A_4_s13.

4. $Si[o2] = 2$
   Now we want to output $S[2]$ as $o2$. This can be achieved with the assumption $j_4 = 255$, due to the fact that $S[4]$ will receive the value 255. A first swap will set $S[2]$ in the index 1 while setting the value 2 in $S[2]$, and a second one will swap $S[4]$ and $S[2]$. These swaps are done respectively in the first and second step of the PRGA. This will set $o2$ as $S[S[2] + S[4]] = S[255 + 2] = S[1]$ (which will have the old value of $S[2]$).

5. $j_1 = 2$
   j_1 is the value of j in the 1st step of the KSA. This constraint condition the way we set the value 2 in $S[1]$. This is done in order to restrain false positive and to reduce the search space.

To achieve this attack, we suppose that $j_4 = Si[255]$, hence we are looking for values of $\boldsymbol{K[4]}$ equal to $\boldsymbol{Si[255] - S[4] - j_3}$. . This assumption will set the value 255 in $S[4]$ during the 5th step of the KSA. Therefore when entering in the PRGA, $S[4]$ will be equal to 255. In the first step of the PRGA, the value contained in $S[2]$ will be preserved in $S[1]$ while the value 2 will be given to $S[2]$. In the second step of the PRGA, $o2$ will receive the value $S[S[2] + S[4]] = S[255 + 2] = S[1]$ which corresponds to the value preserved at the beginning of the PRGA. To understand the previous sequence of events in detail, we will look into the PRGA:

1st step of PRGA:

Before the swap:

0	1	2	3	4	...
?	2	α	?	255	...

$i = 1$ and $j = S[1] = 2$

After the swap:

0	1	2	3	4	...
?	$\boldsymbol{\alpha}$	**2**	?	255	...

the value 2 is set in $S[2]$

2nd step of PRGA:

Before the swap:

0	1	2	3	4	...
?	α	2	?	255	...

$i = 2$ and $j = 2 + S[2] = 4$

After the swap:

0	1	2	3	4	...
?	α	**255**	?	**2**	...

$o2 = S[255 + 2] = S[1] = \alpha$

Now if we consider the IV $\boxed{160}\boxed{97}\boxed{195}$ with $K[3] = 250$, and we look for the value of $\boldsymbol{K[4]}$ (equal to $\boldsymbol{Si[255] - S[4] - j_3 = 56}$ in our example), we get the following for the KSA:

Initialization:

0	1	2	3	4	...
0	1	2	3	4	...

$i_0 = 0$, $j_0 = 160$

1st step:

0	1	2	3	4	...
160	1	2	3	4	...

$i_1 = 1$, $j_1 = 2$ (notice that as required $160 + 97 + 1 = 2$)

2nd step:

0	1	2	3	4	...
160	**2**	**1**	3	4	...

$i_2 = 2$, $j_2 = 198$

3rd step:

0	1	2	3	4	...
160	2	**198**	3	4	...

$i_3 = 3$, $j_3 = 195$
(recall that $198 + 3 + 250 = 195$)

4th step:

0	1	2	3	4	...
160	2	198	**195**	4	...

$i_4 = 4$, $j_4 = 199 + K[4] = 255$
(recall that we are looking for $K[4]$ value
and that we are assuming that $j_4 = Si[255]$)

5th step:

0	1	2	3	4	...
160	2	198	195	**255**	...

We can thus derive $K[4]$ as being equal to $255 - 199 = 56$ due to the assumption made on the value of j_4. The behavior of the PRGA shows us the importance of such hypothesis.

1^{st} step of PRGA:

Before the swap:

0	1	2	3	4	...
?	2	α	?	255	...

$i = 1$ and $j = S[1] = 2$

After the swap:

0	1	2	3	4	...
?	$\boldsymbol{\alpha}$	**2**	?	255	...

The value 2 is set in $S[2]$.

2^{nd} step of PRGA:

Before the swap:

0	1	2	3	4	...
?	α	2	?	255	...

$i = 2$ and $j = 2 + S[2] = 4$

After the swap:

0	1	2	3	4	...
?	α	**255**	?	**2**	...

Hence $o2 = S[255 + 2] = S[1] = \alpha$

4.16 16^{th} attack: Korek A_u5_4

Probability of success: $(\frac{253}{256})^{256-p} \approx 5.25\%$ ($p = 6$)

In this attack we are going to enlarge the scope of the value of p to elements strictly greater than 4. We will also assume that $S[1] = 2$ in order to catch the value of $S[4]$ in the computation process of $o2$. We will assume that j_4 will target directly the value that $o2$ will have, hence $\boldsymbol{K[p]}$ will be computed such as: $\boldsymbol{K[p] = Si[o2] - S[p] - j_{p-1}}$. With this assumption, $o2$ will inherit its value from $S[p]$. Let us see how this is possible and what are the necessary conditions for this attack:

1. $p > 4$
   As said previously, we are looking for values of p greater than 4. The reason behind it is that we need both $S[1]$ and $S[4]$ to remain unchanged and to be fixed in the KSA. Thus to fix $S[4]$, p has to be strictly greater than 4.

2. $S[1] = 2$
   By setting the value 2 in $S[1]$, we will set during the first step of the PRGA this value into $S[2]$. Hence when starting the second step of the PRGA the internal state will be such as $S[2] = 2$, with a previous j equal to 2. This means that the next j will be equal to 4, swapping the values of $S[2]$ and $S[4]$. (This condition is not forgotten in WepLab, while AirCrack made use of it).

3. $S[4] + 2 = p$
   Because we use the same scheme as in the attack A_4_s13, we call and use the value $S[4]$ in order to set the output, and due to the fact that we want $S[p]$ to be outputted as $o2$, $S[4]$ needs to be equal to $p - 2$ so when it will get summed by 2 the index of $o2$ will target p.

4. $Si[o2] \neq 1$ and $Si[o2] \neq 4$
   As said just above we do not want to modify values in $S[1]$ nor in $S[4]$. Thus we need to discard the cases where they are outputted. These cases may happen only if they were altered.

Let us try to explain this attack with the PRGA steps.

1st step of PRGA:

Before the swap:

0	1	2	3	4	...	p	...
?	2	α	β	$p-2$	...	γ	...

$i = 1$ and $j = S[1] = 2$

After the swap:

0	1	2	3	4	...	p	...
?	$\boldsymbol{\alpha}$	**2**	β	$p-2$	...	γ	...

2 is set in $S[2]$

Once the first step of the PRGA is done, the second one will call the value of $S[4]$ in order to output $S[p]$.

2nd step of PRGA:

Before the swap:

0	1	2	3	4	...	p	...
?	α	2	β	$p-2$	...	γ	...

$i = 2$ and $j = 2 + S[2] = 4$

After the swap:

0	1	2	3	4	...	p	...
?	α	$\boldsymbol{p-2}$	β	**2**	...	γ	...

We can see here that $o2$ will take the value
$S[S[2] + S[4]] = S[(p-2) + 2] = S[p] = \gamma$

This attack is feasible because we suppose that $S[p]$ contains the byte that will be outputted as $o2$. For this j_p has to be equal to the index of $o2$ in the p^{th} step of the KSA: $j_p = Si[o2]$. Therefore we can compute $\boldsymbol{K[p]}$ such as: $\boldsymbol{K[p] = Si[o2] - S[p] - j_{p-1}}$. To fully understand this attack, let us see a concrete example. If we take the IV $\boxed{49}\boxed{208}\boxed{76}$ with $K[3] = 233$, $K[4] = 197$, $K[5] = 66$ and $o2 = 216$, while looking for the value of $\boldsymbol{K[6]}$ (which is equal to $\boldsymbol{Si[o2] - S[p] - j_{p-1} = 135}$ in our example), we will obtain the following in the KSA:

	0	1	2	3	4	5	6	...
Initialization:	0	1	2	3	4	5	6	...

$i_0 = 0$, $j_0 = 49$

1st step:

0	1	2	3	4	5	6	...
49	1	2	3	4	5	6	...

$i_1 = 1$, $j_1 = 2$ (recall that $49 + 208 + 1 = 2$)

2nd step:

0	1	2	3	4	5	6	...
49	**2**	**1**	3	4	5	6	...

$i_2 = 2$, $j_2 = 79$

3rd step:

0	1	2	3	4	5	6	...
49	2	**79**	3	4	5	6	...

$i_3 = 3$, $j_3 = 59$ (recall that $79 + 233 + 3 = 59$)

4rd step:

0	1	2	3	4	5	6	...
49	2	79	**59**	4	5	6	...

$i_4 = 4$, $j_4 = 4$
(recall that $59 + 197 + 4 = 4$,
which respects our condition: $p - 2 = 6 - 2 = 4$)

5rd step:

0	1	2	3	4	5	6	...
49	2	79	59	**4**	5	6	...

$i_5 = 5$, $j_5 = 75$

6th step:

0	1	2	3	4	5	6	...
49	2	79	59	4	**75**	6	...

$i_6 = 6$, $j_6 = 81 + K[6]$
(recall that we are looking for $K[6]$ value)

7th step:

0	1	2	3	4	5	6	...
49	2	79	59	4	75	$S^6[81 + K[6]]$	...

Where S^6 represents the internal state in the 6th step.

Now let us have a look at how the PRGA will react:

1st step of PRGA: $i = 1$ and $j = S[1] = 2$

0	1	2	3	4	5	6	...
?	$\boldsymbol{\alpha}$	**2**	?	4	?	$S^6[81 + K[6]]$	...

(we do not take into consideration
the first output byte here)

2nd step of PRGA: $i = 2$ and $j = 2 + S[2] = 4$

0	1	2	3	4	5	6	...
?	α	**4**	?	**2**	?	$S^6[81 + K[6]]$	...

We see that $S^6[81 + K[6]]$ will be outputted as $o2$, due to the fact that $o2 = S[S[2]+S[4]] = S[4+2] = S[6]$. Since we know the value of $S^6[81+K[6]]$ ($= 216$) we can then derive the value of $K[6]$: $K[6] = Si^6[216] - 81 = 216 - 81 = 135$. (recall that in the sixth step of the KSA, the index $Si[216]$

has not been modified yet).

4.17 17th attack: Korek A_neg

This last attack corresponds to the last group of attacks covered by Korek: the reduction of search space size. In this section, Korek identified four subsections of cases in which certain values of key byte are rejected.

4.17.1

The first subsection filters the cases where the KSA at the p^{th} step is as follows:

0	1	2	3	4
?	2	0	?	...

According to this internal state, and if $S[1]$, $S[2]$ do not change values later in the KSA, we will obtain a first output equal to 2.

1^{st} step of PRGA:

Before the swap:

0	1	2	3	...
?	2	0	?	...

$i = 1$ and $j = S[1] = 2$

After the swap:

0	1	2	3	...
?	**0**	**2**	?	...

$o1 = S[0 + 2] = 2$

Therefore, if we notice an output $o1$ equal to 2, with the above estimation of the KSA, we can deduce that the value of j_p has not altered $S[1]$ neither $S[2]$. Hence in this case we can reject some key bytes according to the following formulas: $\boldsymbol{K[p] \neq 1 - S[p] - j_{p-1}}$ and $\boldsymbol{K[p] \neq 2 - S[p] - j_{p-1}}$.

4.17.2

The second group of cases still takes as a condition $S[2] = 0$. Moreover, we look for the second output byte, and we select those who have a null value ($o2 = 0$), while excluding the elements of the previous cases. In other words we can summarize the conditions as follow:

1. $S[2] = 0$
2. $o2 = 0$
3. $S[1] \neq 2$ or $o1 \neq 2$

Let us first analyze this subsection by taking in consideration only the fact that $S[1]$ has to be different than 2. We will then be in the situation where the p^{th} step of the KSA will be as follow:

0	1	2	3	4
?	α	0	?	...

With $\alpha \neq 2$

Notice that here α will neither be equal to 0, due to the fact that 0 is already attributed to $S[2]$. If we follow the behavior of the PRGA we will get:

1^{st} step of PRGA:

Before the swap:

0	1	2	3	...	α	...
?	α	0	?	...	?	...

$i = 1$ and $j = S[1] = \alpha$

After the swap:

0	1	2	3	...	α	...
?	**?**	0	?	...	$\boldsymbol{\alpha}$	...

(We do not take into consideration the value of $o1$ for the moment)

2^{st} step of PRGA:

Before the swap:

0	1	2	3	...	α	...
?	?	0	?	...	α	...

$i = 2$ and $j = \alpha + S[2] = \alpha$

After the swap:

0	1	2	3	...	α	...
?	?	$\boldsymbol{\alpha}$	?	...	**0**	...

$o2 = S[\alpha + 0] = 0$
(corresponding to our requirement)

Here it is easy to see that the value in $S[2]$ should be kept unchanged, thus key bytes that make such a modification should be discarded, which gives us the following criteria $\boldsymbol{K[p] \neq 2 - S[p] - j_{p-1}}$ should be discarded.

Now if we consider the case where $o1 \neq 2$, we can check its relevance by analyzing what happens when we have at the same time $S[1] = 2$ (this situation is possible according the "or" in our conditions). By looking at the first group of cases, we understand that this situation will not bring us much knowledge. WepLab and AirCrack have used it as a mean of optimization in their code, and not in the scope of a theoretical conclusion.

4.17.3

The third group changes the condition of selection as we are now interested in the cases where $S[1] = 1$. If $S[1]$ is not modified, it will force the value in $S[2]$ to be outputted first, hence the second condition which is $o1 = S[2]$.

0	1	2	3	4
?	1	$o1$	?	...

We see easily that if $S[1]$ is kept unchanged and remained equal to 1, the first step of the PRGA will output $o1 = S[S[1] + S[S[1]]] = S[2]$. Therefore to obtain this behavior neither $S[1]$ nor $S[2]$ have to be changed. This means that key bytes that modified them have to be discarded, which is translated into these two equations: $\boldsymbol{K[p] \neq 1 - S[p] - j_{p-1}}$ and $\boldsymbol{K[p] \neq 2 - S[p] - j_{p-1}}$. WepLab and AirCrack implements both of them.

4.17.4

The last group of dropped key bytes given by Korek, is related to the case where values 0 and 1 are kept at the beginning of the internal state: $S[1] = 0$ and $S[0] = 1$. If these two values are unmodified after the pre-computation of the KSA, they will result in the PRGA to create an output $o1$ equal to 1 ($o1 = 1$ which will be our criteria to determine this case).

1st step of PRGA:

Before the swap:

0	1	2	...
1	0	?	...

$i = 1$ and $j = S[1] = 0$

After the swap:

0	1	2	...
0	**1**	?	...

$o1$ will be then equal to $S[0 + 1] = 1$

Therefore we are tempted to say that both $S[0]$ and $S[1]$ need to stay constant until the PRGA, as AirCrack and WepLab did. However, here both are wrong, due to the fact that if we set $j_p = 0$ we will still get an output $o1$ equal to 1. Recall our p^{th} step of the KSA where we have:

0	1	...	p	...
1	0	...	p	...

If we assume that $j_p = 0$ we will have at the next step:

0	1	...	p	...
$\boldsymbol{p}$	0	...	**1**	...

Which gives the same result for $o1$ once given to the PRGA:

1st step of PRGA:

Before the swap:

0	1	...	p	...
p	0	...	1	...

$i = 1$ and $j = S[1] = 0$

After the swap:

0	1	...	p	...
0	$\boldsymbol{p}$	...	1	...

$o1$ will be then equal to $S[0 + p] = 1$

To conclude with this group the only key bytes that need to be dropped are those which modify $S[1]$, thus $\boldsymbol{K[p] \neq 1 - S[p] - j_{p-1}}$.

Chapter 5

New attacks?

Just by looking at the types of these three groups presented by Korek, we can see that the knowledge of the following outputted key byte (the seven first ones and even the eight one sometimes) is not yet exploited. Therefore new attacks can still be founded. To prove this fact, we present here a new one based on the logic of the A_4_s13 attack. This attack should be taken with some precaution, as we have noticed some bias, and is left for further analysis.

5.1 18^{th} attack: Mansor A_4_s5_1

Probability of success: $(\frac{253}{256})^{256-p} \approx 5.13\%$ ($p = 4$)

In this new attack we are going to limit ourselves to the search of $K[4]$ value ($p = 4$ is one of our condition, thus we will not use the term p as it is equal to 4). We will also assume that $S[1] = 2$ in order to catch the value of $S[4]$ in the computation process of $o2$. In this attack, we are assuming that j_4 will take the value of the index of 1 in order to place 1 into $S[4]$. With this assumption, $o2$ will inherit the value in the index 3. The probability of success of this attack is given as a theoretical result. We noticed in practice that some bias exist and leave this part for further research. Let us see how this is possible and what are the necessary conditions for this attack:

1. $p = 4$
   As said previously, we are looking for the value of $K[4]$.

2. $S[1] = 2$
   By setting the value 2 in $S[1]$ we will set during the first step of the PRGA this value into $S[2]$. Hence when starting the second step of the PRGA the internal state will be such as $S[2] = 2$, with a previous j equal to 2. This means that the next j will be equal to 4, swapping the values of $S[2]$ and $S[4]$.

3. $o2 \neq 0$
   We are no longer targeting a null $o2$, to avoid redundancy with A_4_s13.

4. $Si[o2] = 3$
   Now we want to output $S[3]$ as $o2$. This can be achieved with the assumption $j_4 = 1$, due to the fact that $S[4]$ will receive the value 1. A first swap will set the value 2 in $S[2]$, and a second one will swap $S[4]$ and $S[2]$. These swaps are done respectively in the first and second step of the PRGA. This will set $o2$ as $S[S[2] + S[4]] = S[1 + 2] = S[3]$.

5. $S[3] > p$
   Because we do not want the value of j_3 to interfere with our required fields, during the KSA, we limit the values that we allow to be strictly greater than p.

To achieve this attack, we suppose that $j_4 = Si[1]$, hence we are looking for values of $\boldsymbol{K[4]}$ equal to $\boldsymbol{Si[1] - S[4] - j_3}$. This assumption will set the value 1 in $S[4]$ during the 5th step of the KSA. Therefore when entering in the PRGA, $S[4]$ will be equal to 1. In the first step of the PRGA, the value 2 will be given to $S[2]$. In the second step of the PRGA, $o2$ will receive the value $S[S[2] + S[4]] = S[1 + 2] = S[3]$ which corresponds to the value expected. To understand the previous sequence of events in detail, we will look into the PRGA:

1st step of PRGA:

Before the swap:

0	1	2	3	4	...
?	2	α	β	1	...

$i = 1$ and $j = S[1] = 2$

After the swap:

0	1	2	3	4	...
?	$\boldsymbol{\alpha}$	**2**	β	1	...

the value 2 is set in $S[2]$

2nd step of PRGA:

Before the swap:

0	1	2	3	4	...
?	α	2	β	1	...

$i = 2$ and $j = 2 + S[2] = 4$

After the swap:

0	1	2	3	4	...
?	α	**1**	β	**2**	...

$o2 = S[1 + 2] = S[3] = \beta$

Now if we consider the IV

71	89	94

with $K[3] = 186$, and we look for the value of $\boldsymbol{K[4]}$ (equal to $\boldsymbol{Si[1] - S[4] - j_3 = 223}$ in our example), we get the following for the KSA:

Initialization:

0	1	2	3	4	...	161	...
0	1	2	3	4	...	161	...

$i_0 = 0$, $j_0 = 71$

1^{st} step:

0	1	2	3	4	...	161	...
71	1	2	3	4	...	161	...

$i_1 = 1,\ j_1 = 161$

2^{nd} step:

0	1	2	3	4	...	161	...
71	**161**	2	3	4	...	**1**	...

$i_2 = 2,\ j_2 = 1$ (recall that $161 + 94 + 2 = 1$)

3^{rd} step:

0	1	2	3	4	...	161	...
71	**2**	**161**	3	4	...	1	...

$i_3 = 3,\ j_3 = 190$

4^{th} step:

0	1	2	3	4	...	161	...
71	2	161	190	4	...	1	...

$i_4 = 4,\ j_4 = 194 + K[4] = 161$
(recall that we are looking for $K[4]$ value
and that we are assuming
that $j_4 = Si[1] = 161$ in our case)

5^{th} step:

0	1	2	3	4	...	161	...
71	2	161	190	**1**	...	**4**	...

With what is above we can easily derive that $K[4] = 161 - 194 = 223$ and this is possible because we assume $j_4 = Si[1] = 161$. Now if we continue with the PRGA we get:

1^{st} step of PRGA:

Before the swap:

0	1	2	3	4	...
?	2	α	β	1	...

$i = 1$ and $j = S[1] = 2$

After the swap:

0	1	2	3	4	...
?	$\boldsymbol{\alpha}$	**2**	β	1	...

The value 2 is set in $S[2]$.

2^{nd} step of PRGA:

Before the swap:

0	1	2	3	4	...
?	α	2	β	1	...

$i = 2$ and $j = 2 + S[2] = 4$

After the swap:

0	1	2	3	4	...
?	α	**1**	β	**2**	...

Hence $o2 = S[1 + 2] = \beta$

Chapter 6

Conclusion

When we tackled this research, we first studied how RC4 and WEP were functioning. Then we started to gather information on what has already been done. Once the information gathered, we noticed two major advances during the past have been made in order to break WEP. The first one was the work done by Fluhrer, Mantin and Shamir, which gave us a good feeling on what the attacks should look like. Then we noticed the work done by Korek, but we did not want to go directly into his code study. We started first to see if we were able to find new attacks on our own. Unfortunately we ended up rediscovering what Korek did. After several deceptions we looked at how Korek did his attacks, and we explained all his techniques. This work was very instructive as it gave us a good feeling of what research is about and how glorifying it is when we discover a new attack, as small as the one presented here. What we found most gratifying in this work is not the result that we have found, but the knowledge gained during the research.
A future perspective for this domain is still existent. We found a small attack but others might be discovered soon. A good path to follow is in the amelioration of the A_neg attack based on some existent bias (we noticed that $o2 = 0$ is a parameter that comes recursively). We can also improve the computation of the probability of success if we use a good likelihood ratio approach. Finally the knowledge of the first seven output byte can be exploited, since only the two first ones have been so far. We expect that the number of needed packet in order to break WEP will decrease around 40'000 in the near future.

Bibliography

[1] Scott R. Fluhrer, Itsik Mantin, Adi Shamir:
Weaknesses in the Key Scheduling Algorithm of RC4.
Selected Areas in Cryptography 2001: 1-24

[2] Adam Stubblefield, John Ioannidis, Aviel D. Rubin:
Using the Fluhrer, Mantin, and Shamir Attack to Break WEP.
NDSS 2002

[3] AirCrack, http://freshmeat.net/projects/aircrack/

[4] WepLab, analyzing WEP encryption security on wireless networks,
http://weplab.sourceforge.net/