

# WiFi WEP Security Lab

Breaking RC4 with the FMS Attack

---

*Network Security – Introduction to Cybersecurity*

*Luca Campa*

## Contents

---

Submission Rules & Setup	2
Introduction	2
RC4 Specification	2
The FMS Attack Logic	4
Your Task	6
References	7

## Rules & Setup

---

### Rules

- You can use any online resources, but you must write the code yourself. I'm aware of the usage of AI tools, but I expect you to understand, describe and justify your solution.
- You can ask for help from the instructor or teaching assistants during office hours.
- Please submit your code and a brief report (no page limit) describing your implementation and results. Upload a zip file containing the report (a PDF file) and your scripts. Include a short guide on how to execute your scripts if instructions are different from the one presented in this document. The report should include:
  - A description of your implementation approach for each exercise.
  - An analysis of the results, including any challenges you faced and how you overcame them.

The report should NOT include:

- Your full scripts.
- If you want to include code snippets, make sure they are relevant to the discussion. You can refer to the code by using the line numbers in the scripts.

The scripts must be well-commented. Each function should have a clear description of type of inputs, outputs and what it does. The code should be organized and readable.

## Introduction

---

The Fluhrer, Mantin, and Shamir (FMS) attack is a famous cryptographic attack that demonstrated critical weaknesses in the RC4 stream cipher [4], eventually breaking the WEP (Wired Equivalent Privacy) protocol used in early Wi-Fi networks [6]. WEP attempted to prevent keystream reuse by prepending a unique Initialization Vector (IV) to the secret key for each packet. However, the FMS attack revealed that this approach was flawed [1, 3].

The vulnerability lies in the Key Scheduling Algorithm (KSA) of RC4. The attack exploits a statistical bias that emerges when specific “weak” IVs are used [1]. By collecting enough encrypted messages that use these weak IVs, an attacker can deduce information about the secret key, one byte at a time [1, 2].

To understand this vulnerability, you will implement the FMS attack against an implementation of RC4.

## RC4 Specification

---

The RC4 cipher used in this exercise operates on bytes (modulo 256).

- State array size:  $N = 256$ . Elements are integers  $0, 1, \dots, 255$ .
- IV length: 3 bytes.
- Secret Key ( $SK$ ) length: 4 bytes.

- Total Key length:  $L = 3 + 4 = 7$  bytes.

The key used for KSA is the concatenation of the IV and the Secret Key:

$$K = IV \parallel SK$$

## 1. Key Scheduling Algorithm (KSA)

The KSA initializes the state array  $S$  and permutes it using the provided key  $K$ .

```

procedure KSA(Key K, KeyLength L)
  // 1. Initialize the state array S with identity permutation
  for i from 0 to 255 do
    S[i] := i
  end for

  // 2. Initialize the second index j
  let j be 0

  // 3. Permute the state array using the key
  for i from 0 to 255 do
    // Update j with key material
    j := (j + S[i] + K[i mod L]) mod 256
    // Swap elements in S
    swap S[i] and S[j]
  end for
end procedure

```

## 2. Pseudo-Random Generation Algorithm (PRGA)

The PRGA generates the keystream word by word. For this exercise, you only need the first keystream word ( $Z_1$ ).

```

procedure PRGA(State S, Size M)
  // 1. Initialize indices for keystream generation
  let i be 0
  let j be 0

  // --- Generate M Keystream Bytes ---
  // Initialize an empty list to hold the keystream bytes
  let Z be []

  for k from 0 to M-1 do
    // 2. Update indices
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256

    // 3. Swap elements in S
    swap S[i] and S[j]

    // 4. Look up the keystream byte from the permuted state
    let t be (S[i] + S[j]) mod 256
    let Z1 be S[t]
    append Z1 to Z
  end for

  return Z

```

## The FMS Attack Logic

The FMS attack is a statistical attack. It works because the choice of a “weak” IV creates a predictable correlation between the key and the initial bytes of the generated keystream [1]. An attacker doesn’t need to break the encryption for every single packet; instead, they gather statistics from many packets to make an educated guess about the key.

**Weak IVs:** The weakness is appearing when the IV has the form  $(A + 3, N - 1, X)$ , where:

- $A$  is the index of secret key ( $SK$ ) element the attacker is trying to guess ( $A \in \{0, 1, 2, 3\}$ ).
- $N - 1$  is the maximum value in the state array (255 for RC4).
- $X$  is any value, which provides the necessary statistical variation.

**The Bias:** When a weak IV is used, a specific vulnerability in the KSA makes it highly probable that the first keystream byte,  $Z_1$ , is related to the state of the cipher at round  $A + 3$  of the KSA. By simulating the KSA up to round  $A + 2$  (using the known IV and any previously recovered key bytes), the attacker can predict the state just before the unknown key byte is used. This allows the attacker to solve for the unknown key byte. The following condition holds with high probability:

$$K[A + 3] = (Z_1 - j_{A+2} - S[A + 3]) \pmod{256}$$

By collecting many packets that match this weak IV structure for a given  $A$ , the attacker can “vote” on the most likely value for that key byte. The value that appears most often is, with high probability, the correct one. The attacker repeats this process for each byte of  $SK$ .

### 1. A Concrete Example: Recovering the First Key Byte ( $A = 0$ )

Let’s walk through an example to see how the attack works for the very first byte of the secret key.

#### Example Walkthrough

Our goal is to find the first byte of the secret key, which corresponds to setting the attack index  $A = 0$ .

#### Select Weak IVs

According to the FMS attack, for  $A = 0$ , the weak IVs have the form  $(A + 3, N - 1, X)$ , which is  $(3, 255, X)$ . We filter our “intercepted\_traffic.csv” file to only consider packets where the IV matches this pattern. For each of these packets, we have the value of  $X$  (from “iv2”) and the corresponding first keystream byte  $Z_1$ . In principle, the value of  $Z_1$  is not known. But since we know the first byte of the plaintext (e.g., a known SNAP header), we can recover  $Z_1$  using the formula:

$$Z_1 = C_1 \oplus P_1$$

where  $C_1$  is the first byte of the ciphertext and  $P_1$  is the known first byte of the plaintext. In this exercise sheet, we assume that  $Z_1$  has already been recovered for each packet.

## Simulate the KSA

The bias formula for  $A = 0$  is:

$$K[3] = (Z_1 - j_2 - S[3]) \pmod{256}$$

Here,  $K[3]$  is the first byte of the secret key we want to find. To use this formula, we need to calculate the state of the KSA simulation just before the unknown key byte is used. This means we simulate the KSA for  $i = 0, 1, 2$  (up to step  $A + 2$ ).

The key used for the simulation is  $K = IV \parallel SK = [3, 255, X, k_0, \dots]$ . We only need the first 3 bytes ( $IV$ ) for this partial simulation.

- **Initial State:**  $S = [0, 1, 2, 3, \dots, 255]$  and  $j = 0$ .
- **KSA round  $i = 0$ :**
  - $j_0 = (j + S[0] + K[0]) \pmod{256} = (0 + 0 + 3) \pmod{256} = 3$ .
  - Swap  $S[0]$  and  $S[3]$ . The state array  $S$  becomes  $[3, 1, 2, 0, \dots]$ .
- **KSA round  $i = 1$ :**
  - $j_1 = (j_0 + S[1] + K[1]) \pmod{256} = (3 + 1 + 255) \pmod{256} = 259 \pmod{256} = 3$ .
  - Swap  $S[1]$  and  $S[3]$ . The state array  $S$  becomes  $[3, 0, 2, 1, \dots]$ .
- **KSA round  $i = 2$ :**
  - $j_2 = (j_1 + S[2] + K[2]) \pmod{256} = (3 + 2 + X) \pmod{256} = (5 + X) \pmod{256}$ .
  - Swap  $S[2]$  and  $S[j_2]$ .

After these three steps, we have the final value for  $j_2$  and the state array  $S$  needed for our formula. Specifically, we need the value of  $S[3]$  *after* the swap at  $i = 2$  has occurred.

## Calculate the Guess and Vote

For each intercepted packet with IV  $(3, 255, X)$  and keystream byte  $Z_1$ , we can now compute a guess for the key byte  $k_0 = K[3]$ .

1. Perform the KSA simulation as described above to find the value of  $j_2$  and  $S[3]$ .
2. Plug these values into the bias formula:

$$k_{0,\text{guess}} = (Z_1 - j_2 - S[3]) \pmod{256}$$

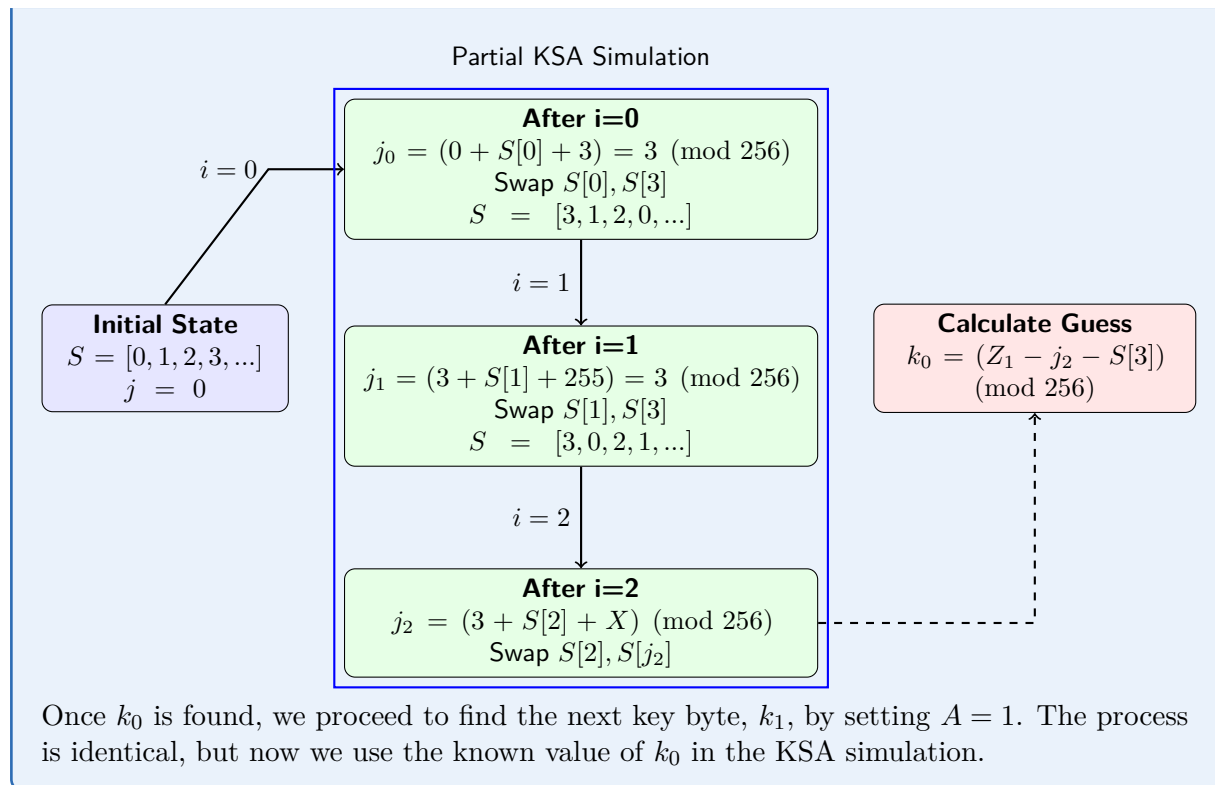
3. Record this guess and increment a counter for that specific value of  $k_0$ .

We do this for every packet with a weak IV for  $A = 0$ . Each calculation produces a guess for which we increment the corresponding counter. After processing all packets, we determine the value that received the most votes.

The value that receives the most votes is, with very high probability, the correct first byte of the secret key,  $k_0$ .

## Visualizing the KSA State

The following diagram illustrates the state of the KSA simulation at each step for  $A = 0$ .



## Your Task

You have intercepted wireless traffic encrypted with RC4. For each packet, the IV is sent in plaintext, and you know the first element of the plaintext (e.g., a known SNAP header). Because  $C_1 = P_1 \oplus Z_1$ , you have successfully recovered the first keystream element  $Z_1$  for many packets.

You are provided with:

- `intercepted_traffic.csv`: 500.000 intercepted packets. Each row (packet) contains `iv0`, `iv1`, `iv2`, `Z1`.
- `toy_rc4.py`: A reference implementation of RC4's KSA.
- `generate_traffic.py`: A script to generate traffic with a random secret key and weak IVs. This will help you to validate and debug your implementation.

### >\_ Task

#### Task

1. Implement the FMS attack in Python to recover the 4-byte secret key from the intercepted traffic.
2. Your script should read the `intercepted_traffic.csv` file, filter for weak IVs, simulate the KSA, and apply the bias formula to guess each byte of the secret key.
3. For each byte of the secret key, keep track of the votes for each possible value and determine the most likely value based on the collected statistics.
4. Output the recovered secret key.

What is the secret key you recovered from the provided intercepted traffic?

## References

---

- [1] S. Fluhrer, I. Mantin, and A. Shamir, “Weaknesses in the Key Scheduling Algorithm of RC4”, in *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001, Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers*, ser. Lecture Notes in Computer Science, vol. 2259. Springer, 2001, pp. 1–24. DOI: [10.1007/3-540-45537-X\\_1](https://doi.org/10.1007/3-540-45537-X_1).
- [2] C. Declan, “FMS Attack PoC on simulated WEP (RC4)”, *Medium*, 2021. [Online]. Available: <https://medium.com/@cdeclanx90/fms-attack-poc-on-simulated-wep-rc4-4fd7be4b0640>.
- [3] Wikipedia, “Fluhrer, Mantin and Shamir attack”, *Wikipedia, The Free Encyclopedia*, 2026. [Online]. Available: [https://en.wikipedia.org/wiki/Fluhrer,\\_Mantin\\_and\\_Shamir\\_attack](https://en.wikipedia.org/wiki/Fluhrer,_Mantin_and_Shamir_attack).
- [4] Wikipedia, “RC4”, *Wikipedia, The Free Encyclopedia*, 2026. [Online]. Available: <https://en.wikipedia.org/wiki/RC4>.
- [5] M. Blaze, “A Cryptanalytic Attack on the RC4 Key Setup”, *Technical Report*, 1995. [Online]. Available: [https://www.mattblaze.org/papers/others/rc4\\_ksaproc.pdf](https://www.mattblaze.org/papers/others/rc4_ksaproc.pdf).
- [6] Wikipedia, “IEEE 802.11”, *Wikipedia, The Free Encyclopedia*, 2026. [Online]. Available: [https://en.wikipedia.org/wiki/IEEE\\_802.11](https://en.wikipedia.org/wiki/IEEE_802.11).