CS6160 - Cryptology
Assignment 2

Vignan Kota
CS21BTECH11029

1. Introduction

This report presents the cryptanalysis of a reduced RC4 scheme to demonstrate vulnerabilities in its keystream generation and key scheduling processes.

In this assignment, we are given a modified RC4 scheme with a 5-bit word size (n = 5) and an internal state consisting of a 32-word table S. Each ciphertext is generated from the same plaintext, a 6-digit passcode containing digit characters ranging from 0 to 9, using a unique randomly generated 5-byte key. A substantial dataset of 2^{24} ciphertexts are available for analysis.

The objective is to exploit potential biases and weaknesses in the RC4 keystream output to deduce the encrypted passcode effectively. By examining the underlying structure and observing patterns across the ciphertexts, we aim to uncover the specific passcode used in the encryption.

2. Approach

To exploit the vulnerabilities of RC4, we implemented a single-bias attack using likelihood estimation to identify the most probable plaintext bytes based on observed keystream biases.

1. **Keystream Generation and Analysis**:
   We generated 2^{24} random keys(seed) of size 5 bytes and input each into the Key Scheduling Algorithm (KSA) and Pseudo-Random Generation (PRG) of RC4 to produce 2^{24} keystreams of size 6 bytes.
   Using these keystreams, we analysed the keystream distribution $Z(r, k)$, where r denotes the byte position(0 - 5), and k represents the possible byte value(0 - 255). This distribution served as the probability basis for identifying keystream biases.

2. **Ciphertext Distribution and Likelihood Calculation**:
   Given 2^{24} ciphertexts generated from plaintext using 2^{24} unique random keys

   Using these ciphertexts, we analysed the ciphertext distribution $N(r, k)$, where r denotes the byte position(0 - 5), and k represents the possible byte value(0 - 255).

   For potential plaintext values μ ranging from '0' to '9', we derived a biased count, $N^{*}(r, k) = N(r, k \oplus \mu)$ to observe the distribution shift when XOR-ing with each possible plaintext digit.

   Using these biased counts, we calculated the likelihood λ, defined as:

$$\lambda = \sum_{k=0}^{255} N^{*}(r, k) \times log(p(r, k))$$

   where $p(r, k) = \frac{Z(r, k)}{2^{24}}$ represents the keystream probability at each byte.

3. **Plaintext Deduction**:
   For each byte position, we identified the value of μ (a digit character from 0 to 9) that maximised λ and used it as the most likely byte of the passcode.

   This approach allowed us to determine the passcode by iteratively selecting the byte values that exhibited the highest likelihood of occurrence based on the biased distribution in the keystream output.

3. Observation

During the analysis, we observed several key aspects of the RC4 keystream and the ciphertext distribution:

1. **Keystream Biases**:
   Specific byte positions in the RC4 keystream showed predictable biases, revealing weaknesses in the RC4 setup.

2. **Likelihood Peaks**:
   Certain digits consistently maximised the likelihood λ, aligning with expected keystream biases and making them strong passcode candidates.

3. **Effectiveness of XOR**:
   XOR with plaintext guesses amplified key distribution patterns, helping identify probable byte values.

4. **Consistent Results**:
   Across all passcode bytes, the approach provided clear, consistent maximum likelihood values, confirming RC4's susceptibility to this type of analysis.

5. **Keystream Distribution Sizes**:
   Testing with different keystream distribution sizes (S = 2^{24}, 2^{25}, 2^{26}, 2^{27}) produced the same passcode, indicating stability in the approach and consistency across larger datasets.

4. Cracked Passcode

Through the likelihood-based cryptanalysis approach, the 6-digit passcode encrypted by the RC4 scheme was successfully deduced as: **475103**

This result was consistent across different keystream distribution sizes, further validating the approach's accuracy in exploiting RC4's keystream biases.

5. Conclusion

The RC4 scheme's vulnerabilities were successfully exploited using bias-based analysis, allowing us to accurately retrieve the passcode.

Consistent results across varying keystream sizes confirm this approach's effectiveness, emphasising the risks of statistical biases in RC4's design.