Applied Crypto

Exploring the Security of One-Time and Many-Time Pads

Overview

This team project investigates the cryptographic security of the one-time pad, known for its perfect security under certain conditions, and explores the implications of reusing a key, referred to as a many-time pad. We will examine the practical aspects of implementing these encryption methods, analyze their vulnerabilities, and develop strategies to exploit these vulnerabilities in a controlled environment. Here is some background story in signal intelligence. https://en.wikipedia.org/wiki/Venona_project

Objectives

1. Understand One-Time Pad
   Examine the concept of the one-time pad, its implementation, and why it is considered perfectly secure.
2. Implement One-Time Pad Encryption and Decryption
   Develop a Python program that simulates the encryption and decryption process between two parties.
3. Explore Many-Time Pad
   Extend the one-time pad implementation to simulate a many-time pad scenario, where a single key is used to encrypt multiple messages.
4. Cryptanalysis of Many-Time Pad
   Design and execute an attack strategy to decrypt messages encrypted with a many-time pad, focusing on exploiting the vulnerabilities introduced by key reuse.

Problem 1: Understanding One-Time Pad

Tasks:

1. Research the theoretical basis of the one-time pad, including its requirements and operational principles.

Problem 2: One-Time Pad Implementation

Tasks: the encryption and decryption process between two parties, Alice and Bob.

1. Alice's Program
   Should prompt for a message input (plaintext), then display the ciphertext, and save both the ciphertext (in hex) and the key (in hex) in separate files.
2. Bob's Program:
   Should read the key and ciphertext from their respective files and display the decrypted plaintext.

Problem 3: Implementing Many-Time Pad

Tasks: Modify the one-time pad implementation to encrypt multiple messages with the same key, simulating a many-time pad scenario. The purpose of this problem is to see if there are any recognizable patterns by observing the outputs. You can gain insights by changing the plaintexts or the key to verify your findings. These findings would be useful in the next problem.

1. The program should encrypt a list of 10 predefined plaintext messages with a single key, saving the plaintexts, key, and ciphertexts (all in hex) into a file. You can select 10 of your favorite messages. Assume the key is long enough to do encryption to all the 10 messages.

Problem 4: Cryptanalysis of Many-Time Pad[1]

Tasks: Develop a strategy to decrypt messages encrypted with a many-time pad. More specifically, assume Eva has collected the 10 ciphertexts and she knew they are generated by the same key. In addition, all the plaintexts are in English. Space, comma, period, and question mark are being used in the plaintext, but no other special characters are allowed. Eva wants to decrypt the last message (target message).

Collected ciphertexts:

1) 71fe1ace4389087266117cd7c98c4182851b3acff3b086e3f83f94d6eb05c4ba85d8e1fa14f11d1
c3b568ff6cff5c09c5d67ef5c9c71b7eeb3d45a5154ab17b83e071ce9d8988adb4afedf46a840

2) 71fe1ace559a1e7266117cd7ce8745d7be2e74c3f0f68eeef57e8884e607debf81dfa0f012f95819
681ae7f29fe4839b5175ef5e8760bef0b9d44b504eba12b22f5404f89dd085d550a48865a14f9b
15a94dabe609ca2df2cccf210cefdb1af5389719795e1f0179cb77c5c456954d88f3

3) 72fe069c51c81a20775928c7879d4fd2a93c3acff3f69fe5fe2e9493a303d9ea98c4e5b60ae40a1
46058e7c787fbd09a1474e25dc865b5e6af865d4a40a61bfd384e06e0cfc1ccd356ff8853ac4389
05fa5fe3fd41cb3bbc8ac9

4) 67e543885b9a5b2267177084cf8453ccb8633ad7fdb39de5b13f8a93a304d6bf8bc4f4ef5def11
0b6f56a3e186e2c68c1470ef5c9c2ffbd6a291571e40ba1afd3b4b1fe0c4cbccc15df5dc07b043d
a01fa6ae4fd158f37b3c0cd

5) 71fe029a148c1236320d7192878a59cfbc3a6ec5e7f68befb13196d6ea1ec4ea81d9e3fe50ea0f1
96d02a2f7cfe2c29c5577e35d8630baf6ea80465b01aa1abc394f57a1f4ccccda59ff8846e44b88
05bb5cabe608c231f2dec8364ae7d90ab4358c5c3a421b06

6) 6ef914ce5989152b321a769ad79c42c7be6f6ad2fab19de1fc339d84f04ad3a589dfa0ff09ab0c1
96f13e7e780b4c097556ded57c871fbeea393464a01aa0ab1381848cfd2d6898918efc046b00b8
940bb08e3f313cb23b3dfd8645cfcd80ff82489

7) 71fe1ace4389087266117cd7c4865bd2b93b7fd2b5a58ce9f4308c9ff01e97ab82cbf2ef5dfc101
d6a56b3fb8ab4d08b4167ef5c9c30b8f0ab97455b45e81efd364605e49ddb83df48eedc42b60c9
00fb14db4b229ca74b6c4d96442e1c34df8288f5c3a450a527ecc7c82865b8e

8) 71fe029a148c1437615978d7c58854dbec2c75cde5a39be5e37e9b97ef0697a285dfa0f01cff101
d764983f29bf5

9) 71fe1ace50875b31730d6ad7cb8640c7ec3c73d4e1bf81e7b13796d6e518d8a4988ceff05dff101
d2415a8fe9fe1d79a4623eb5e8430bfe3b3d442514faf40fd18420be0c8cb89924cf3cd5ee44895
0efd5cabe500c120f2d9d26440ebc34de029811977430b01748276d79012955cc6a65aebb9054
becda5c9278

[1] http://crypto.stanford.edu/~dabo/cs255/hw_and_proj/hw1.html

10) 71fe029a1483123c76597691878459cca9363ac4faf68ceffc2e8d82e61897b98fc5e5f809e20b0c7756b2e08aab83bc5560e257

Target Message:

71fe0680149d083b7c1e3996879a42d0a92e7780f6bf9fe8f42cd898e61cd2b8ccd9f3f35dff101d241da2eacff9cc8d5123fe5a897efbeda4974b

1. Design an attack strategy based on the vulnerabilities of key reuse in a many-time pad scenario. You can use some of the hints given below.
2. Implement the strategy in jupyter notebook. Remember, most of the time cryptoanalysis needs a human in the loop and a bit of luck.
3. Analyze and discuss the outcomes, including any partial decryption results and insights gained from the process.

Hints: Observe and reasoning.

Basic XOR Properties:

- **Hint A:** XORing 0 with any bit (0 or 1) leaves it unchanged, symbolized as $\mathbf{0 \oplus x = x}$. This property is fundamental to understanding how XOR affects data.
- **Hint B:** XORing 1 with any bit flips it, which is to say $\mathbf{1 \oplus x = 1 - x}$. This operation inverts the bit, playing a crucial role in encryption and decryption processes.
- **Hint C:** XORing any bit string with itself results in a string of zeros, expressed as $\mathbf{k \oplus k = 0}$. This property is useful for key and data recovery techniques.

Encryption and Decryption with XOR:

- **Hint D:** The ciphertext **c** is produced by XORing the plaintext **m** with the key **k**, denoted as $\mathbf{c = m \oplus k}$. This is the basic encryption formula.
- **Hint E:** Decrypting involves XORing the ciphertext **c** with the key **k** to retrieve the original message, represented as $\mathbf{m = c \oplus k}$.
- **Hint F:** The key **k** can be derived by XORing the ciphertext **c** with its plaintext **m**, shown as $\mathbf{k = c \oplus m}$. This equivalence highlights the symmetric nature of XOR-based encryption. It also reveals the key info if we know something about m and c. Here the phrase "we know something about m" means we know some of the characters of m in some positions, not necessarily the m.

Analyzing Multiple Encryptions (Many-Time Pad):

- **Hint G:** When two ciphertexts, $\mathbf{c1 = m1 \oplus k}$ and $\mathbf{c2 = m2 \oplus k}$, are XORed together, the result is $\mathbf{c1 \oplus c2 = m1 \oplus m2}$. This reveals the XORed plaintexts without the key, offering insights into patterns or repeating structures within the messages. Notice, we have ten of them. The combinations can reveal a lot more on the plaintexts.

ASCII Patterns and Cryptanalysis:

- **Hint H:** Observing ASCII representations, capital letters A-Z range from **0x41** to **0x5A**, showing a leading bit pattern of **010** (0x4 = 0100, 0x5 = 0101), while lowercase letters a-z range from **0x61** to **0x7A**, with a leading bit pattern of **011**. Spaces are represented by **0x20**, with a pattern of **0010** (=0x2). XOR operations between these (a letter and a space) can reveal shifts between uppercase and lowercase letters, aiding in pattern identification within encrypted texts.

- **Hint I:** Numeric characters in ASCII start with the bit pattern **0011**, assisting in distinguishing numbers from letters during analysis.

Advanced Cryptanalysis Techniques:

- **Hint J:** Beyond ASCII patterns, students shall employ crib dragging (exploiting known or guessed plaintexts) from above hints, particularly from Hint H and Hint K (Understanding the structure of the key, even partially, enables the decryption of the ciphertext including the target message. This often requires intuitive guesses and methodical verification to bridge gaps in the known data.
- **Hint K:** students may employ other cryptanalysis methods such as frequency analysis (identifying common letters or patterns), and dictionary attacks on short segments. These techniques leverage patterns and statistical properties of the plaintext to breach the encryption.

Deliverables

1. Written Report: Includes a comprehensive explanation of each problem, the one-time pad, implementation details for both one-time and many-time pads, the attack strategy on the many-time pad, and an analysis of the results.
2. Code: Use Jupyter Notebook for the one-time pad encryption/decryption, many-time pad encryption, and the cryptanalysis of the many-time pad.

Evaluation Criteria

- Correctness: The one-time and many-time pad implementations must function as specified, accurately encrypting and decrypting messages.
- Attack Strategy: The effectiveness of the cryptanalysis approach in exploiting the vulnerabilities of the many-time pad, including creativity and technical execution.
- Analysis: Depth of analysis in understanding the security implications of key reuse and the ability to communicate findings clearly.

Submission

It is the same as Project 1.

Grading rubrics (total 12)

	Max Point	Expectations
Problem 1	**1**	
Task 1	1	Describe one-time pad conditions clearly
Problem 2	**3**	

Task 1	1.5	Alice code is running correctly, and two files are generated. It meets the best coding practice such as comments, readability and maintainable.
Task 2	1.5	Bob code uses the key file to recover the plaintext correctly. It meets the best coding practice such as comments, readability and maintainable.
Problem 3	**2**	
Task 1	2	The code runs correctly. It meets the best coding practice such as comments, readability and maintainable.
Problem 4	**6**	
Task 1	1	Design a feasible attack strategy. If the strategy could not fully recover the target plaintext (task 3), some points will be deducted. The actual points deducted depend on the outcome of Task 3.
Task 2	3	The code runs correctly with or without human intervention. It meets the best coding practice such as comments, readability and maintainable.
Task 3	2	The target message is decrypted correctly.