Cryptography and Secure Development -

Midterm Assignment

Name: Sujan
UID : 2823161S

Introduction:

Cryptography is a technique used to secure communication from unintended recipients by converting plain text into cipher text through various methods. While this provides a level of security, encryption systems can be vulnerable to attacks. For instance, I discovered that a "known plaintext attack" can occur when an attacker has both the plaintext and the corresponding cipher text. By examining the plaintext and cipher text, the attacker can identify the encryption key used and decipher any other messages encrypted with the same key. Another type of attack that can compromise the security of an encryption system is the "cipher only attack." In this scenario, the attacker only has access to the cipher text and must use statistical analysis and other methods to find the encryption key, as neither the plaintext nor the encryption key are known to them. Through my studies, I realized the importance of understanding these attacks and their potential impact on the security of encrypted communication.To successfully decrypt messages encrypted using an 8-rotor encryption machine in Java, I must use a combination of analytical and computational methods, be familiar with the machine's design and operation, and thoroughly understand it. This will allow me to devise an effective plan of attack.

Known Plaintext Attack:

Key found: jerk
Plaintext : This is the plaintext (task 1) for you * with the ID number = 28231613148 to decode. Good luck!
Time taken to find the key: 15 milli-seconds
Time complexity: The total time complexity of the code is **O(NLS*W).**
(Number of passwords=N, length of the ciphertext = L,length of the string = S, number of words in the string=W)

- Scanning passwords: O(N)
- Decrypting-ciphertext: The code tries each password, so the time complexity of decrypting the ciphertext using all passwords is O(N*L).
- Checking English sentence:O(S*W)

Methodology:

Step1:Reading and Storing Data from Text File:

The passwords in the text file are read by this code, which keeps them in a list called keys. The file is read using a buffered reader, and any potential FileNotFoundException is caught in a try-catch block. Up until there are no more lines left to read, the while loop reads each line of the file and adds it to the keys list. In order to release resources, the bufReader is finally closed.

Step 2: Data Decryption and Filtering

This code snippet employs a loop and the Rotor96Crypto method encdec to decrypt data (). It decrypts a portion of ciphertext using a password from the keys list. If “Th” appears at the beginning of the decrypted string, the password is added to the finalkeylist. The finalkeylist is printed to the console at the end.

Step 3: Screening for Acceptable English Sentences and Decrypting (Manual texting of sentence is possible but i preferred to give an automatic approach to the problem)

This code piece employs a loop to go through the finalkeylist and the Rotor96Crypto method encdec to decode the data (). The data is then decrypted, and a custom EnglishSentenceChecker class is used to determine whether any legitimate English phrases are present. The FinalAns and Finalkey are set to the encrypted data and password key, respectively, if the decrypted data comprises proper English sentences. The correct key, the plaintext, and the length of the loop are then reported to the console.

EnglishwordValidator():

A static function called wordvalidate() that accepts a string sentence and returns a boolean value indicating whether the sentence contains an acceptable English word is defined in this code as a class called EnglishwordValidator. Based on a regular expression pattern, the validity is determined by looking for valid English words that only contain lowercase letters, hyphens, spaces, and terminate with a period (.,?, or!).

Estimation of LikelyHood:

1. **Probability Theory approach:**

   To obtain the decrypted form of the ciphertext, a set of keys were brute-forced upon it. Only 5 of the 9473 decrypted messages had the letter "Th" at the beginning and the keys are: [alpha, jimjim, jerk, elpaso, verena]

- Total number of keys with decrypted message started with “Th” :- 5
- For each key, the likelihood of not discovering the true key is 1 - 1/9473 = 0.9989.
- With all five keys, the likelihood that the genuine key won't be discovered is =(0.9989)^5 = 0.9945.

- Hence, the likelihood that a set of five keys has at least one real key is 1 - 0.9945 = 0.0055, or 0.5%

2. **Theory of Entropy, Absolute rate, Actual rate and unicity distance:**
    - Number of messages, all are equally likely(n) = 5 (text starts with “Th”)
    - H = n (1/n log2(n)) = log2(n)
    - H= log2(5) = > **2.322**
    - Absolute rate (R) = R = log2(L), where L is the number of English letters. Since two letters are given, L=2
    - R=> log2(2) => **1**
    - Actual rate (r) =H(X) / N, where N is the number of letters in a plaintext.Therefore N=95.
    - r=> 2.322 / 95 => **0.02444**
    - Unicity distance = 2^(rN) / 2^(RN) => **0.50**
    - Hence the probability of getting meaningful message is **0.5**
3. **Explanation of how I found the key.**
    - After finding the set of 5 keys. I could have checked manually but I chose to construct a separate class to check the validity of English sentences.
    - Based on the assumption that no English word has capital letters in between,Then checking for valid alphabets, then later hyphens should always be surrounded by letters,and at last all punctuation marks should be at the end of the word.

Ciphertext Only Attack:

Key found: noname

Plaintext : This is the second task (Ciphertext Only Attack) of the assignment of the Cryptography and Secure Development course. * - you are expected to decode it with the assumption that this plaintext is English sentences. This plaintext contains this random number 4466 so that you cannot guess it from the others :) Finger-crossed!

Time taken to find the key: 8 seconds

Time complexity: Overall, the time complexity of this code is O(N * min(50, M) * L * S), which can be simplified to O(NMLS). Where N is the length of the keys array, where M is the length of the ciphertext string.

Methodology:

Step 1:Reading and Storing Data from Text File:

Reading the passwords from the text file and storing them in a list called keys. A buffered reader is used to read the file, and any possible FileNotFoundException exceptions are captured in a try-catch block. The while loop reads each line of the file and adds it to the keys list until there are no more lines left to read. The bufReader is finally closed to free up resources.

Step 2: Creating a list of stopwords and filtering the decrypted messages

Determining the unicity distance, or the number of ciphertext letters required to unambiguously decode the message, by computing the entropy, absolute rate, and real rate of the password. The unicity distance is then printed by the programme both theoretically and practically. The second section of the code consists of a loop that iterates through a list of final keys, decrypts the ciphertext with each key using the Rotor96Crypto.encdec() method, determines whether the ciphertext is a valid English sentence using the isvalidEnglishSentence() method, and prints the key and plaintext if it is.

isvalidEnglishSentence():

When a sentence is provided as input, the isvalidEnglishSentence method returns a boolean value indicating if the text is or is not a valid English sentence. Before creating an array to contain the frequency of each English letter in the sentence, the sentence is first converted to lowercase. By iterating through the text and just taking the lowercase letters into account, it updates the letter frequency array. The proportion of English letters in the sentence is then calculated, along with the overall number of English letters in the sentence. The method returns true, indicating that the statement is a legitimate English sentence, if the percentage is more than or equal to 0.5. Otherwise, false is returned.

Theoretical calculations:

- Number of messages, all are equally likely(n) = 9473
- H = n (1/n log2(n)) = log2(n)
- H= log2(9473) = > **13.2096**
- Absolute rate (R) = R = log2(L), where L is the number of English letters. Since two letters are given, L=26
- R=> log2(26) => **4.7**
- Actual rate (r) =**1.5**
- D= R-r => **3.2**
- Unicity distance => Nu =H(K)/D
- Unicity distance => 13.2096 / 3.2
- Unicity distance => **4** (approx)
- Hence number of cipher text letters needed before unambiguous decoding is **4**

Difference between Theoretical approach and practical approach

Theoretical approach	Practical approach
● Suggested only 4 letters is sufficient to decrypt the ciphertext	● Needed 7 letters in practical to decrypt the cipher text
● Reason being that it only considers the entropy, absolute rate, actual rate and unicity distance but in reality the english language is very complex and its highly unambiguous to determine the validity just by 4 letters	● It could be possible to determine the validity of the sentence by starting with 4 letters but it is a good practice to check a few more characters to avoid failure of the system.
● May reduce time complexity	● May increase the time complexity
● But reduced accuracy	● Increased accuracy
● Example :“Thisadadsdsd” Above sentence has “this” in the starting. By theoretical explanation 4 letters is enough to check the validity of an english sentence but it fails when the next characters are not checked.	● Example :“Thisadadsdsd” We check for the next character even after finding the stopwords at first to increase the accuracy. Hence I used 7 letters to decrypt the ciphertext to plaintext.