SIL765: Networks and System Security
Semester II, 2020-2021
Assignment-1

February 11, 2021

Submission Instructions

1. You may use one of the three languages: C++, Java and Python.
2. Your code should run on Linux (version 16 and above).
3. You will submit the assignment on Moodle.
4. You should submit a single folder which should contain a separate sub-folder corresponding to each problem in the assignment.
5. Each sub-folder should contain all files corresponding to the problem. For example, each sub-folder may contain:
   (a) source codes,
   (b) executables of your program,
   (c) plots (in jpg format), and
   (d) screenshots of output (in jpg format), and
   (e) corresponding readme file (in pdf format).
6. It is important to use the following naming convention:
   - The folder should be named as: ⟨Your Entry Number⟩-assignment-⟨Assignment Number⟩.
     Example: 2020EE10350-assignment-1
   - The sub-folder should be named as: problem-⟨Problem Number⟩.
     Example: problem-1
   - Each file should be named as specified in the problem.

7. The readme file should contain the details about the solution. It should also explain the steps to build and execute your code. It should also contain some sample inputs and outputs. Since the TA will evaluate your submission based on the details mentioned in readme file, please make sure to provide all necessary details here.
8. The TA should be able to evaluate your code by giving command line arguments. To facilitate that, please do not hard-code any input.
9. You can use Piazza for any queries related to the assignment.
10. Failure to strictly follow the above steps will adversely affect the grades.

Problem-1: Basic Cryptanalysis (30 Marks)

Background

Substitution is one of the ways to encrypt messages. In fact, the first few ciphers in the history utilized just the substitution. However, the substitution-based ciphers can be easily broken using basic methods of cryptanalysis.

To-Do List

In this assignment, you will utilize a basic cryptanalysis technique to break a substitution cipher. The character set used for substitution is:

```
character set for substitution mapping: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '@',
'#', '$', 'z', 'y', 'x', 'w', 'v', 'u', 't', 's', 'r', 'q', 'p', 'o', 'n']

original character set: ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q',
'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
```

The following are the two ciphertexts (also provided in the files) encrypted using a substitution cipher. By use of any cryptanalysis method, decrypt the ciphertext to obtain plaintext. Also, find out the letter substitution mappings.
Hint: The space, comma and full-stop characters are not part of the encryption/decryption process. Hence, you can use them as anchors to decrypt other characters.

Cipher 1:

v 85yv86v@ov79 $p oq9 79@92vu 0$@$vu3qv19o5t t53q92 5y oqvo 1$oq y9@892 v@8 29t95692 0syo t$005o oq9 3920so98 t53q92 y94s9@t9 o$ 090$2x. v t$00$@ o9tq@54s9 p$2 v6$585@7 oq5y 5y o$ sy9 v w9x#$28 p2$0 #q5tq oq9 t53q92 y94s9@t9 tv@ 19 79@92vo98. p$2 9rv03u9, sy5@7 oq9 w9x#$28 t53q92, #25o9 $so oq9 w9x#$28 puu#98 1x s@sy98 u9oo92y 5@ @$20vu $2892 v@8 0votq oq5y v7v5@yo oq9 3uv5@o9ro u9oo92y. 0vw9 29vy$@v1u9 vyys03o5$@y v1$so q$# o$ o29vo 298s@8v@o u9oo92y v@8 9rt9yy u9oo92y 5@ oq9 090$2x #$28y v@8 q$# o$ o29vo y3vt9y v@8 3s@tosvo5$@. 5@85tvo9 #qvo x$s2 vyys03o5$@y v29. @$o9, oq9 09yyv79 5y p2$0 oq9 yq92u$tw q$u09y @$69u, oq9 y57@ $p p$s2.

Cipher 2:

zqwq1@qz 1yz 4q19ryp nrv zryyqt #y@$#3nqz, nq 5qy@ @$ 7qz. @n1@ yrpn@ nq zrz y$@ v4qq8 5q44, n19ryp wq9qtrvn ztq1sv, n19ryp y$ tqv@. nq 51v #yv#tq 5nq@nqt nq 51v 1v4qq8 $t ztq1sryp. 3$yv3r$#v, #y3$yv3r$#v, 144 51v 1 74#t. nq tqsqs7qtqz 3t2ryp, 5rvnryp, n$8ryp, 7qppryp, q9qy 41#pnryp. nq w4$1@qz @nt$#pn @nq #yr9qtvq, vqqryp v@1tv, 841yq@v, vqqryp q1t@n, 144 7#@ nrsvq4w. 5nqy nq 4$$6qz z$5y, @t2ryp @$ vqq nrv 7$z2, @nqtq 51v y$@nryp. r@ 51v x#v@ @n1@ nq 51v @nqtq, 7#@ nq 3$#4z y$@ wqq4 1y2@nryp w$t x#v@ nrv 8tqvqy3q.

Given Files

- **cipher1.txt** → cipher1 in a .txt format file.
- **cipher2.txt** → cipher2 in a .txt format file.

Expected Submission

- *source code and executable:* You need to provide the source code used. The executable should be named as *substitutioncrack* so that your code can be executed in the terminal as:
  ./substitutioncrack ciphert1.txt
- *readme file:* This should include how you obtained the plaintext, i.e., the approach used for cracking the cipher. You have to include the mapping of alphabet characters to substitution characters for both cases. Also, you have to present the plaintext obtained from each ciphertext.

Hint: You can try frequency analysis of characters.

Problem-2: Birthday Attack (35 Marks)

Background

SHA-3 is the state-of-the-art method to compute hash functions. To achieve acceptable level of cryptographic security (similar to encryption using a 128-bit key), the suggested length of the hash output is 256 bits. Note that we have to double the hash output length. This is to prevent the feasibility of the birthday attack that expedites finding the hash collisions.

To-Do List

In this assignment, you have to implement the birthday attack against SHA-3 with smaller length of the hash output as follows.

- Given the length (in bits) of the hash output d, write a program which returns a tuple (s_1, s_2, h, m, n) where s_1 and s_2 are two strings of the same length and with the same hash output h, m is the largest memory (in bits) required to store the strings during the execution of the program, and n is the number of attempted/tested string pairs to find the collision.
- For each value of d between 1 and 24 bits, output three tuples with collisions.
- Compute the average number of attempts n_{avg} and average memory m_{avg}.
- Draw a plot where X-axis represents the length of the hash output d and Y-axis represents the average number of attempts n_{avg}.
- Draw another plot where X-axis represents the length of the hash output d and Y-axis represents the average memory m_{avg}.
- Ask yourself (or Google), are there ways to reduce either the number of attempts or the memory. If so, refine your program.

Expected Submission

- *source code:* Given the length of the hash output d, your code should return a tuple (s_1, s_2, h, m, n) as discussed above.
- *plot-1:* X-axis: d and Y-axis: n_{avg}.
- *plot-2:* X-axis: d and Y-axis: m_{avg}.
- *readme file:* You can discuss the algorithm used in your code. You can provide details about the plots, i.e., report all the tuples utilized to obtain the plots.

Note: The marks in this problem will be inversely proportional to the memory used and number of attempts. In other words, higher marks will be given to those who could report collisions with lower memory and/or lower attempts.

Problem-3: Cracking RSA (35 Marks)

Background

Consider the public key cryptographic algorithm, RSA. It uses a public key and a private key. Messages encrypted using the public key can only be correctly decrypted using the private key. The RSA works as follows:

- Key Setup
  - Choose two prime numbers, p and q.
  - Compute $n = p \cdot q$.
  - Compute $\Phi = (p-1) \cdot (q-1)$.
  - Choose an integer e such that $gcd(e, \Phi) = 1$.
  - Choose an integer d such that $d \cdot e \pmod{\Phi} = 1$.
  - Publish the public key $\{e, n\}$.
  - Keep the private key $\{d, n\}$.
- Encryption
  - Given a plaintext M, use the public key, and compute the ciphertext $C = M^e \pmod{n}$.
- Decryption
  - Given a ciphertext C, use the private key, and compute the plaintext $M = C^d \pmod{n}$.

Note that RSA works with numeric inputs, and the security of RSA depends on the computational difficulty of factoring large values of n.

Given Files

- **nlist.txt** $\rightarrow$ List of numbers n with increasing number of digits.
- **plaintext.txt** $\rightarrow$ Plaintext M.

To-Do List

In this assignment, you have to crack RSA for small values of n as follows.

- Given a number n from the file *nlist.txt*, compute the prime factors p and q.
- Record the time taken t to obtain these prime factors.
- Compute the public and private keys.
- Given the message M in the file *plaintext.txt*, use the public key to encrypt the message M to ciphertext C.
- Decrypt the obtained ciphertext C back to the plaintext to validate your code.

Expected Submission

- *source code:* You need to provide the source code for cracking RSA given a value of n, i.e., it should output the values of t, p, q, e, d, M and C as discussed above.
- *executable:* The executable should be named as *crack* so that your code can be executed in the terminal as: *./crack n*
- *plot:* You have to plot a graph where X-axis represents the number of digits in n and Y-axis represents the time taken to factorize n.
- *readme file:* You should specify the largest n from the file *nlist.txt* that can be cracked by your algorithm within five minutes. You should also specify the configuration of the laptop/desktop used by you. Further, you can add the discussion of the obtained plot.