CSE350/550: Network Security (Jan-May 2024)

Programming assignment no. 1 (due 11.59 pm, THU FEB 01, 2024)

Listed below, you will find brief description of 3 projects, numbered 0 through 2. In groups of 2 you are required to:

a. pick one project (see algorithm below for you to pick a project),
b. complete that project, and
c. submit a report (with a working system) before **11.59 pm, THU FEB 01, 2024**. The outcome will be evaluated by me and the TAs in an oral presentation that you will make.

Further:

a. You may use any programming language that you are comfortable with, including C, C++, Java, Python, etc. on any platform, Linux, MS Windows, etc., and
b. Do not copy your assignment from another group, or allow others to copy your assignment – be aware it is easy for us to find out (it will also show up in the oral presentation you will make).

Algorithm to pick a project: pick project numbered 0, 1, or 2 as determined by k = (A1+A2) mod 3, where
A1 = last_4_digits_of_entry_no_of_first_student, and
A2 = last_4_digits_of_entry_no_of_second_student.

The submission will consist of four parts:

1. a 2- to 4-page Word or pdf document describing the system you have designed,
2. sample inputs and/or outputs from running the code you have written,
3. the code itself as a separate file, and
4. 5 to 8 slides that you will use to present your work during your presentation to me & TAs.

In each of the projects listed below:

a. You are required to develop executable programs to encrypt, decrypt, and more importantly launch brute-force attack to discover the key,
b. The difference between the three projects are (i) the character set and the manner of crafting the plaintext, (ii) the specific encryption/decryption algorithm you will use, and (iii) the method for launching a brute force attack,
c. In Project 0 the character set is {A, B, C}. In projects Project 1 and Project 2, the character set consists of lower case English letters, viz. {a, b, …, z}. In all cases the plaintext you will work with should be "recognizable". To make the text recognizable, the plaintext p should satisfy some property, π, that can be checked by an algorithm or a program and declare whether π(p) =true or false. This property π can take different forms. The simplest form where plaintext == original_text ||original_text, where || is the concatenation operator, will not work. Therefore, you should identify or construct a good hash function Hash(.) that you may use to construct a plaintext, p = (string, Hash(string)), where "string" is the original text. The Hash function should be such that the received or decrypted string can be recognized by an algorithm or program.
d. You are encouraged to explore different Hash functions. But, here is one function that is simple and will suffice for the present - not good in any real application. This one is described in Figure 11.5, Chapter 11 of Stallings book (7 th edition).

   This requires one to lay out the string in the form of a NxM matrix with say 8 columns. And then add the elements row-wise, as given below.

$$
\begin{aligned}
O_1 &= D_1 \\
O_2 &= O_1 \oplus D_2 \\
O_3 &= O_2 \oplus D_3 \\
&\dots \\
&\dots \\
O_N &= O_{N-1} \oplus D_N
\end{aligned}
$$

O_N is the Hash value.

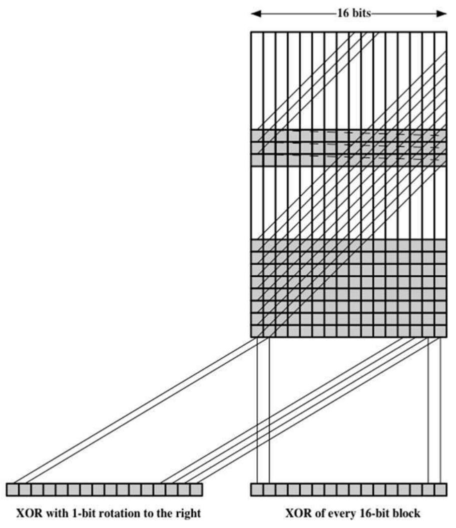
16 bits
XOR with 1-bit rotation to the right
XOR of every 16-bit block

Figure 11.5 Two Simple Hash Functions

Figure 11.5 assumes the string is a bit string, but in our case the string is (i) tuples of characters A|B|C in Project 0, or (ii) lower case English letters a|b|…|z in Projects 1 and 2. The addition rules for bits is simple, but what is it when the character set is {A, B, C} or {a, b, …, z}? The answer lies in mapping the characters set onto a set of integers and then doing modulo arithmetic.

e. In all projects, you will:
   i. Prepare some 5 plaintexts as above, p = (string, Hash(string)), that can then be encrypted to obtain corresponding ciphertexts using the given & known key;
   ii. This collection of ciphertexts will be decrypted one-by-one using the (same) known key to compute the plaintexts. You should then visually inspect the resulting plaintexts to be sure the encrypt and decrypt algorithms/programs are working fine.

f. Limit the length of plain-text as well as size of keys suitably so that the computations can be carried in good time; necessarily the length of plaintext and the key will be guided more by how much computer time will be used when you launch a brute force attack.

g. Finally, in all projects you will launch a brute force attack by trying out various keys, k1, k2, k3, etc. and decrypt (starting with) the first ciphertext. Once you discover a key say k4 such that the resulting plaintext satisfies the property π, use the same key k4 to decipher the second ciphertext and check if the resulting plaintext also satisfies the property π. If yes, continue with third ciphertexts, etc. ELSE, reject the key, k4, and restart to discover a new key that will generate (from all ciphertexts) plaintexts that satisfy property π.

Project 0: Encryption & decryption using mono-alphabetic substitution of a pair of characters at a time, viz. <xy>, where x ε {A, B, C}, y ε {A, B, C}. Encryption uses a table consisting of 9 rows of tuples of the kind e.g. AB→BD, AC → BA, etc. The resulting ciphertext pair of characters is <pq>, where p ε {A, B, C}, q ε {A, B, C}. Then develop the software to launch a brute-force attack to discover the key. The plaintext should be long enough and “recognizable” as described above.

Project 1: Encryption & decryption using poly-alphabetic substitution of the kind discussed in class. Then develop the software to launch a brute-force attack to discover the key. Assume that the key length is known, and it is 4.

Project 2: Encryption & decryption using transposition of the kind discussed in class. Then develop the software to launch a brute-force attack to discover the key. Here assume that the key length is known to be 9 or less.