Fall 2024, CMPUT 496 - A1 Cryptography for Digital Privacy Assignment One

Professor: Bailey Kacsmar

Please use Canvas for questions and clarifications, you can send messages in canvas to the TA. The TA may decide to make responses to your private questions public so that everyone benefits from knowing the same information. Questions where you need to post partial solutions or questions that describe the locations of vulnerabilities or code to exploit those vulnerabilities will be sanitized before the TA provides a public response.

Total Marks: 40 . *Due: September 26, 2024 by 4pm MT*
TA: Steven Hai
TA Office Hours: Tuesdays at 3:30pm-4:30pm in room CAB 313

Sample Explanation: Recall the many-time pad attack from class. If you were to explain it as you will for questions 1(a-c), it could look something like the following.

- First, xor the two ciphertexts together, this will result in an output corresponding to the two messages xord together
- At this point, you know that the values that have been xord are non-uniform (they correspond to English text). This limits the space of possibilities to components from the English language. Start by looking for expected values (possibly a space), if the space xord with another value produces a legitimate character, that is likely the correct character. Proceed through common/likely English characters until the output is either i) a coherent english phrase or ii) close enough to guess the phrase
- The solution can be "verified" by xoring the messages with their original ciphertexts, if the output is the same value, this is the key.

1 Written (20 Marks)

1. **(3 marks)** Identify the plaintext for each of the following ciphers. You **must** include an explanation of each step and analysis you took to arrive at this plaintext to receive credit for the solution. Note that saying you inputted the ciphertext into a cipher solver does not count for explaining analysis. The ciphertext is everything between quotation marks, but does not include the quotation marks.

   (a) Ciphertext: "fleivrazeiiluixpxciwfnetrmctuizyajdxwewzbqegusesmxchvlolrjevucfcdinmt enfdmtrtowffmogtayofspkppzdqedxruleqypzsaopetuuminlfmogpfcpeylmtuyoqvmboiyr flelfcfcuxyhgltgqoertidfzgofnoylzhigbpacatrbbtpfzpeltawwmjfxdtpobermjedsmzeufeyyaxiy jeo"

   - Cipher is: Vigenere cipher
   - Plaintext source: "Ron was wrong, Whit is right" [1].

[1] https://eprint.iacr.org/2012/064.pdf

(b) Ciphertext: "ioaboayqimppjojombpgvabynpgcbmnjyqzomblyabynpgcbmnjyjgkodobpjobokmi magzvwiqgzmtgwpkjmpkmipgtoeonpioabopabynpgiyiposiiwajmipjoamoimbaqnjobqzkjqaj omajloppobqibonlmaohtypjogzopjboonlmaoivwbpjobgzigmqiambbqohpghtpgoopahonozho hvgbpjoqbioawbqpygzeoonqzcpjoozpqboozabynpqgznbgaoiiioabop"

- Cipher is: affine cipher
- Plaintext source: "New Directions in Cryptography" [2]

2. **(5 marks)** Break a many-time pad.
The many-time pad should specifically exploits knowledge that the text is a sub-strings are from a larger document. For the following, you have three ciphertexts, each have been encrypted using the same key. You also know that the three messages are from three different documents. Thus each ciphertext decrypts to text that can be found in either "Ron is Wrong, Whit is Right" `https://eprint.iacr.org/2012/064.pdf`, "New Directions in Cryptography", `https://www-ee.stanford.edu/~hellman/publications/24.pdf`, and "Turtles, Locks and Bathrooms", `%https://petsymposium.org/popets/2018/popets-2018-0029.php`. The three ciphertexts are provided in the assignment folder.

   (a) Identify the plaintext for each of the ciphertexts

   (b) You must include an explanation of each step and analysis taken to arrive at the plaintext in order to receive credit for the analysis.

   (c) You must provide any code you use to solve this.

   (d) Note that saying you inputted the ciphertext into a solver does not count for explaining analysis.

   (e) You do not know if the messages had spaces removed and/or punctuation removed before they were encrypted.

   (f) Your explanation should clearly delineate how your approach used the source text for the attack and how the approach is distinct from the general sample explanation provided. Someone else should be able to follow your instructions and also acquire the messages.

3. **(5 marks)** Alice wants to send a message m to Bob. She wants to ensure confidentiality and integrity using public key cryptography. What can she do? Identify appropriate cryptographic primitives/protocols and explain how they achieve these requirements.

4. **(5 marks)** Alice wants to have Bob sign a message for her, but she does not want Bob to learn the message. Develop an algorithm for this using RSA. (Hint: Multiply the "to be signed message" before sending it to Bob, such that you can undo the multiplication once Bob responded.)

5. **(2 marks)** Alice wants to use El Gamal, but wants to better understand the security guarantees and limitations. Assume the adversary guesses correctly in the El Gamal IND-CPA security proof. What is the solution of the simulator?

2 Programming (20 Marks)

You are free to use either python or c++. If you want to use a different language please reach out to the instructors. You must implement each question yourself from scratch. You may use

[2] `https://ee.stanford.edu/~hellman/publications/24.pdf`

Algorithm 1 EXTENDED EUCLIDEAN ALGORITHM(A,B)

▷ Inputs a and b are integers, the algorithm computes integers r, s, and t such that $r = gcd(a, b)$ and $sa + tb = r$.

$a_0 \leftarrow a$
$b_0 \leftarrow b$
$t_0 \leftarrow 0$
$t \leftarrow 1$
$s_0 \leftarrow 1$
$s \leftarrow 0$
$q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
$r \leftarrow a_0 - qb_0$
while $r > 0$ **do**
    $temp \leftarrow t_0 - qt$
    $t_0 \leftarrow t$
    $t \leftarrow temp$
    $temp \leftarrow s_0 - qs$
    $s_0 \leftarrow s$
    $s \leftarrow temp$
    $a_0 \leftarrow b_0$
    $b_0 \leftarrow r$
    $q \leftarrow \lfloor \frac{a_0}{b_0} \rfloor$
    $r \leftarrow a_0 - qb_0$
$r \leftarrow b_0$
return (r, s, t)

the math and IO libraries from your language, but not any other library (no cryptography library). It might be worth noting that the programming can be solved without importing any libraries in python. You may re-use code from previous questions to help solve subsequent questions. We have included a sample input for each question. The assignment will be graded on different but similarly formatted inputs of the same size.

1. **(3 marks)** Implement extended GCD (extended Euclidean algorithm for finding GCD) shown in Algorithm 1 on the this page.

   Formatting:
   When run, it should read the inputs (as integers) from a file "input_q1.txt". "input_q1.txt" will have one number per line with "a" on the first line and "b" on the second line. It should then write the results (as integers) to a file "output_q1.txt" with one number per line in the following order: r, s, t.

2. **(7 marks)** In this problem, you must implement the Baby-Step Giant-Step algorithm presented in class for computing discrete logarithms in $\mathbb{Z}_p^*$. Specifically, for a prime p, in the group $\mathbb{Z}_p^*$, given g and h in $\mathbb{Z}_p^*$, find x such that $g^x = h \mod p$. Moreover, if the algorithm fails to find a solution x, it must report it. (Think about why the algorithm might not find a solution)

   Notes:

   - You are not allowed to use any cryptographic library for modular exponentiation and other operations. You are allowed to use a sorting function and libraries such as math and random if necessary. If you are uncertain, reach out to the TA.

- Since it should work with very large numbers, its simple mentation should run in $\mathcal{O}(log(b))$ where b is the exponent when computing $a^b \mod p$.
- Make sure to implement the Baby-Step Giant-Step algorithm. A brute-force approach will not be accepted.
- In the test cases, the prime number p will be up to 40 bits, so your code must be implemented efficiently.
- Your code must run in a reasonable amount of time, preferably in less than a minute. We will use a strong machine to test the code and will be lenient in this requirement. However, if you are uncertain about this requirement, contact the TA.
- The code will be tested on a machine with a single core, so avoid using parallelization to speed up your code. Improving the code using parallelization is not helpful.

Formatting:
When run, it should read the inputs (as integers) from a file "input_q2.txt" placed next to the source code. "input_q2.txt" will have one number per line with p on the first line and g and h on the second line and third line. It should then write the result, x, (as an integer) to a file "output_q2.txt". If there does not exist a solution x, print 0 to the file.

3. **(10 marks)** In question 1, you saw the paper "Ron was wrong, Whit is right" `https://eprint.iacr.org/2012/064.pdf`. The paper studies a number of RSA public keys and determines how many offer no security. We have provided a file (*decoded_keys.txt*) filled with the modulus n for 50 RSA public keys (one per line), some of which offer no security[3]. Your job is to identify and break the vulnerable keys.
Formatting:
When run, it should read the keys in the provided "decoded_keys.txt" file, then identify and break the vulnerable keys. It should then write the results to a file "output_q3.txt" where each line contains 3 comma-separated values: first the index of the key in the file (starting at 0), followed by the two prime factors of the broken key (also comma-separated).

3 General Information

- Submissions are to be uploaded to the appropriate Canvas folder as a zipped folder containing both the written and programming components. The written component **must** be a PDF. Name the written file following the format of yourCCID_assignment1.pdf.
- The standard late policy found in the syllabus from this courses applies.
- The standard policies found in the syllabus on academic integrity and avoiding improper collaboration applies.
- If you use python, your code file should be "q1.py" (or q2.py for quesiton two etc.). It should run on its own without any dependency besides basic python library imports like math and random. It should be able to be run using the command "python3 q1.py". If you use c or c++, your code must compile and be contained within a singular file or no marks will be given.

[3]The original RSA public keys are contained in a file called "real_keys.txt". Each key contains e,n and some additional meta-data, encoded in PEM format. For simplicity we have decoded the keys for you and "decoded_keys.txt" contains only the modulus value n for each key (one per line).