CS 6324: Information Security **Spring 2021**

Homework #2

Due date & time: 11:59pm CST on March 12, 2021. Submit to eLearning by the due time.

Additional Instructions: The submitted homework must be typed. Using LaTeX is recommended, but not required.

Problem 1 (10 pts) Cryptanalysis against Encryption Modes

- (4 pts) Suppose that everyone in the world is using the DES (Data Encryption Standard) algorithm in the ECB (Electronic Code Book) encryption mode, and you can use a chosen plaintext attack against everyone. Show how to perform a dictionary attack such that, after an expensive initialization step, everyone's key can be recovered in very little time. Write pseudo code both for the initialization step and for the function to recover everyones key.
    - Initialization:

Let CiphertextKeyPair[ ][ ] be a 2D array of string that has the total word count of the dictionary amount of rows, 2^56 amount of columns.

Let possibleKeys[ ] be an array of size of 2^56 consists of all possible keys generatable in 56 bits

Let Dictionary [ ] be words of dictionary in array format

```
For i = 0 to 2^56{
    Compute all permutation in 56 bits and fill in possibleKeys[i]
}
For j = 0 to total word count of dictionary{
    For k = 0 to 2^56{
        CiphertextKeyPair[ j ][ k ] = Encryption of Dictionary[ j ] using possibleKeys[k] ;
    }
}
```

    - Key finding function:

Let C_i be the input ciphertext

```
For i = 0 to 2^56{
    For j = 0 to word count of dictionary{
        If (C_i == CiphertextKeyPair[ i ][ j ]){
//Traverse to find the matching ciphertext in the previously prepared 2D array (table),
```

```
//then return the found corresponding key.
                    Return possibleKeys[i];
                }
        }
}
```

- (2pts) How effective would the above attack be if AES, instead of DES, is used?
    - It is not as effective as the 2D array look up table for corresponding ciphertext to keys is too larger and will take a long time to create.
- (2pts) How effective would the above attack be under known-plaintext (instead of chosen plain-text) attacks?
    - It will be more effective as those known pairs of plaintext-ciphertext yield what key it is used. It improve the speed to generate the 2D array.
- (2pts) How effective would the Chosen Plaintext attack described above be if everyone uses the CBC encryption mode with the IV randomly chosen?
    - The effectiveness will be similar because the possibleKeys array contains all kinds of keys as long as it in the 56-bits range defined by DES. Matching key and ciphertext can still be found.

Problem 2 (6 pts) Levels of security

- (2 pts) Assume that you are doing an exhaustive key search attack and can check 2^{32} (or about 4G) keys per second, about how many years would it take to for you to search through all keys while attacking ciphers of the following key lengths: 56, 64, 80?
    - Keys possible for length of 56 = 2^56 keys.

$$\frac{2^{56}\ keys}{2^{32}\frac{keys}{second}} = 2^{24}\ seconds$$

$$\frac{2^{24}\ seconds}{3600\frac{seconds}{hour}\times 24\frac{hours}{day}\times 365\frac{days}{year}} = \frac{2^{24}\ seconds}{31536000\frac{seconds}{year}} = 0.5320020294\ year\ \approx 0.532\ year$$

    - Keys possible for length of 64 = 2^64 keys
- $\frac{2^{64}\ keys}{2^{32}\frac{keys}{second}} = 2^{32}\ seconds$
- $\frac{2^{32}\ seconds}{3600\frac{seconds}{hour}\times 24\frac{hours}{day}\times 365\frac{days}{year}} = \frac{2^{32}\ seconds}{31536000\frac{seconds}{year}}\ \approx 136.193\ years$

    - Keys possible for length of 80 = 2^80 keys
- $\frac{2^{80}\ keys}{2^{32}\frac{keys}{second}} = 2^{48}\ seconds$

- $\frac{2^{48}\ seconds}{3600\frac{seconds}{hour}\times 24\frac{hours}{day}\times 365\frac{days}{year}} = \frac{2^{48}\ seconds}{31536000\frac{seconds}{year}} \approx 8925512.96\ years$

- (2 pts) Still assuming that you can check 2^{32} keys per second, further assume that the age of universe is 13.7 billion years, how many ages of the universe does it take to search through a key space of 128 bits.

- $\frac{2^{128}\ keys}{2^{32}\frac{keys}{second}} = 2^{96}\ seconds$
- $\frac{2^{96}\ seconds}{3600\frac{seconds}{hour}\times 24\frac{hours}{day}\times 365\frac{days}{year}\times 13.7\times 10^{9}\frac{years}{age}} = \frac{2^{96}\ seconds}{31536000\frac{seconds}{year}\times 13.7\times 10^{9}\frac{years}{age}} = \frac{2^{96}\ seconds}{4.320432\times 10^{17}\frac{seconds}{age}} \approx 1.8338\ \times 10^{11} ages$

- (2 pts) Does this mean that a cipher of 128 bit can never be broken by brute-force? Justify your answer.

    - No. Since computing speed is improving with the advancement of technology, such as higher speed processor and quantum computer, which allows checking keys in a higher speed, eventually making brute force possible.

Problem 3 (9 pts) We use birthday collision to illustrate breaking the three desirable properties for cryptographic hash functions: Preimage resistant, Second preimage resistant, and Collision-resistant. Assume that there are 365 days in a year, and the birthday of each person is distributed uniformly. We consider three groups: Group A has 25 persons, group B has 35 persons, and group C has 180 persons. We assume that the distribution of birthdays for group members are random. Answer the following questions for each of the three groups: A, B, and C. Write out the equation, then give the numerical results (which can be computed by using calculators or writing a small program. You don't need to provide your code).

- (3 pts) What is the probability that a group contains someone with date of birth being Jan 1? (This corresponds to preimage attack.)

Group A: $P(only\ one\ has\ birthday\ of\ Jan.\ 1) = \frac{365!}{365^{25}(365-25)!} \approx e^{\frac{-25(25-1)}{2\times 365}} \approx 0.43959$

Group B: $P(only\ one\ has\ birthday\ of\ Jan.\ 1) = \frac{365!}{365^{35}(365-35)!} \approx e^{\frac{-35(35-1)}{2\times 365}} \approx 0.19590$

Group C: $P(only\ one\ has\ birthday\ of\ Jan.\ 1) = \frac{365!}{365^{180}(365-180)!} \approx e^{\frac{-180(180-1)}{2\times 365}} \approx 6.7850 \times 10^{-20}$

- (3 pts) What is the probability that a group contains someone that has the same date of birth as you? You are not part of the group. (This corresponds to second preimage attack.)

Group A: $1 - \left(\frac{365-1}{365}\right)^{25} \approx 0.066288$

Group B: $1 - \left(\frac{365-1}{365}\right)^{35} \approx 0.091556$

Group C: $1 - \left(\frac{365-1}{365}\right)^{108} \approx 0.25643$

- (3 pts) What is the probability that one can find two members in a group such that they have the same date of birth? (This corresponds to collision attack.)
    - P(n): In group of n people, 2 have same birthday.
    - $\bar{p}(n)$: In group of n people, no one has a same birthday.
- Group A: $P(25) = 1 - \frac{365!}{365^{25}(365-25)!} \approx 1 - e^{\frac{-25(25-1)}{2\times365}} \approx 0.56041$
- Group B: $P(35) = 1 - \frac{365!}{365^{35}(365-35)!} \approx 1 - e^{\frac{-35(35-1)}{2\times365}} \approx 0.804097$
- Group C: $P(180) = 1 - \frac{365!}{365^{180}(365-180)!} \approx 1 - e^{\frac{-180(180-1)}{2\times365}} => very\ close\ to\ 1$

Problem 4 (10 pts) Message Authentication Code (MAC)

a (3 pts) When we say that an MAC needs to resist Existential Forgery under Chosen Plaintext Attack, what do we mean? State the precise definition.
We mean that the adversary should not be able to come up with another method to produce a value t_j' using message M' that is the same as the value of message M' hashed under challenger's key k.

To be secure, ∀M', otherMethod(M')≠MAC(k, M')

b (3 pts) Consider the following MAC construction that uses AES. Given a message M, it is padded to so that its length in bits is a multiple of 128, and divided into blocks of 128 bits each: $M_1, M_2, \ldots, M$. The MAC value under a key k is defined as $MAC_k(M) = AES_k(M_1) \oplus AES_k(M_2) \oplus \ldots \oplus AES_k(M)$. Show that this MAC construction is insecure by giving a concrete attack.

As the hash output is expected to be multiples of 128 bits, it is required to mod the XOR result of all AES_k with 2^128. We can first try some number of plaintext and obtain their hash, then analyze what values can produce the same hash by recovering the value prior to modding with 2^128. With those steps done, we have obtained a target XOR result such that as long as the XOR result of all message blocks matches the target, the attack will be successful.

c (4 pts) Consider another MAC construction that uses AES. We limit ourselves to consider messages whose lengths are multiple of 96 bits, and are at most $2^{32} \times 96$ bits long. Given a message M, it is divided into blocks of 96 bits each: $M_1, M_2, \ldots, M$. Each M_i is then padded to 128 bits by including the binary encoding of i in the beginning of each block M_i. We use M_i to denote the padded blocks. The MAC value under a key k is defined as $MAC_k(M) = AES_k(M_1) \oplus AES_k(M_2) \oplus \ldots \quad AES_k(M)$. Show that this MAC construction is insecure by giving a concrete attack.

We can first place message blocks in groups of greatest common factor of 96 and 128. Obtain the hash of our chosen message. Recover the XOR result prior to modding with 2^128. Now we have a target XOR result with padded bit. We can then analyze in groups what other messages after padding and XOR can yield to the target XOR result combined with the padded bits. Those other message can then be used to send in the name of victim.

Problem 5 (7 pts) CBC-MAC In class, we discussed HMAC. Another commonly used MAC scheme is Cipher Block Chaining Message Authentication Code, abbreviated CBC-MAC. This constructs a

mes- sage authentication code scheme from a block cipher. To compute the CBC-MAC of a message given a key K, one uses the key K to encrypt the message in CBC mode, using 0 as the IV, and the final ciphertext block is the MAC value.

a (2 pts) Let E_K be the encryption function with block size b and key K, $B_0 = 00\ldots0$ be the IV used in CBC-MAC, and $\oplus$ to denote bit-by-bite XOR. Write the formula for the CBC-MAC of a two-block message $M = M_1M_2$, i.e.

CBC-MAC$_K(M_1M_2) = E_K(\,(E_K(B_0 \oplus M_1)) \oplus M_2)$.

b (5 pts) CBC-MAC is only secure for fixed-length messages. Show that when variable-length messages are allowed, CBC-MAC is insecure against a chosen-plaintext attack. That is, show that given M and CBC-MAC$_K[M]$, how to come up M and CBC-MAC$_K[M]$ for some M.

An attacker who knows 2 correct pairs of M and CBC-MAC$_K$ tag, namely $M^{(1)}$ −CBC-MA$C_K^{(1)}$ pair and $M^{(2)}$ − CBC-MA$C_K^{(2)}$pair, can produce a method to generate a third message $M^{(3)}$ whose CBC-MA$C_K^{(3)}$ tag will also be CBC-MA$C_K^{(2)}$.

Proof: Let $M^{(3)} = M^{(1)} || (\,(M^{(2)}{}_1 \oplus$ CBC-MA$C_K^{(1)}) || M^{(2)}{}_2 || \ldots . || M^{(2)}{}_b)$,

In order to get CBC-MA$C_K^{(3)}$, CBC-MAC$_K(M^{(3)})$

$= E_K(E_K(\,(E_K(B_0 \oplus M_1^{(3)})) \oplus M_2^{(3)}) \ldots \oplus M_b^{(3)})$

$= E_K(\,(M^{(2)}{}_1 \oplus$ CBC-MA$C_K^{(1)} \oplus$ CBC-MA$C_K^{(1)}))$

$= E_K(\,(M^{(2)}{}_1) =$ CBC-MA$C_K^{(2)}$

So, CBC-MAC$_K[M_3] =$ CBC-MA$C_K^{(3)} =$ CBC-MA$C_K^{(2)}$

Problem 6 (12 pts) RSA encryption. Assume that the message space consists of all 24-bit non-negative integer numbers. We use $N = 60529591$ for RSA encryption.

a (3 pts) Write a program that computes p and q such that $pq = N = 60529591$. Provide the values of p, q, and the value of $\Phi(N)$.

b (3 pt) We choose $e = 31$. Provide the value of d such that $(ed \bmod \Phi(N)) = 1$.
(The algorithm for doing this is known as the extended Euclidean algorithm. If you do not know the algorithm, and do not want to learn the algorithm yourself, you can use an inefficient method such as enumerating over all positive integer numbers less than $\varphi(N)$ and checking which one satisfies the requirement. For large N, this inefficient method takes too long time; but for the number here, it should be fast enough.)

c (0 pt) If you are unfamiliar with RSA, write a program that loops through all value x's such that $1 \leq x \leq N - 1$, to verify that $(x^{ed} \bmod N) = x$. This is just to help convince yourself that RSA decryption works.

Hint: When computing $x^y \bmod N$, directly computing x^y may overflow the representation of integers you have. You can use the Square and Multiply method, which exploits the fact that $x^{ab} \bmod N = (x^a \bmod N)^b \bmod N$. You can also exploit the fact that $x^{y+1} \bmod N =$

$(x^y \bmod N)x \bmod N$.

d (3 pts) For each of the following ciphertexts 10000, 20000, 30000, 40000, 50000, 60000, show the decrypted message, as a base-10 number. Here we allow these messages to be beyond a 24-bit integer.

//I am unable to output the correct cipher text because of failing to due with the over flow problem

e (3 pts) Describe how an adversary can recover any message encrypted in this way without factoring N. That is, the message space remains the same. However, the value of N is so large that factoring it becomes infeasible.

- One may try to recover the multiplication result of e * d to recover without factoring N.

Problem 7 (12 pts) ElGamal encryption. Again assume that the message space consists of all 16-bit integers. Consider ElGamal encryption. Let us choose $p = 96737$. Then we have

$$Z_p^* = \{1, \ldots, 96736\}$$

a (3 pts) Write a program to find g, the smallest generator of Z_p^*, i.e., g is the smallest number in Z_p^* such that the set $\{g^1 \bmod p,\ g^2 \bmod p, \cdots,\ g^{96736} \bmod p\} = Z_p^*$. Provide the value g.
(The program can simply tries each number starting with 2 and see whether it is a generator. A number a is a generator for Z_p^* if and only if the smallest positive integer $\boldsymbol{j}$ such that $g^j \bmod p = 1$ is $j = p - 1$.)
//Program does not work due to unable to deal with overflowing representation.

b (3 pts) Write a program that computes the discrete logarithm in Z_p^*, and use it to compute the value $x = \log_g 90307$ in Z_p^*. That is, find x such that $(g^x \bmod p) = 90307$. (You can use the inefficient brute-force method.)

c (3 pts) Write a program that implements ElGamal decryption. Then decrypt the following ciphertexts: (5000, 5001), (10000, 20000), (30000, 40000), (50000, 60000), (70000, 80000), (90000, 100000), which were encrypted under the public key given above, i.e., $(p, g, 90307)$. Show each decrypted message as a base-10 number.
(In El Gamal decryption of (c_1, c_2), you can compute x and need to find M such that $(xM \bmod p) = c_2$. To do this, you need to first compute $z = (x^{-1} \bmod p)$, that is, compute z such that $(xz \bmod p) = 1$. Given z, you can compute $M = (zc_2 \bmod p)$. Refer to Part b of the RSA question for how to find z.)

d (3 pts) In Problem 6e, you showed how to break RSA encryption without factoring N. Can you recover a the plaintext from an ElGamal ciphertext without solving the discrete log problem or the Computational Diffie-Hellman problem, which are infeasible when p is very large? If yes, describe how. If no, explain why.

- No, it is not feasible to solve it. There are 3 hard problems relating to ElGamal, DLG, CDH and Decision Diffie-Hellman (DDH). While DDH does not regard to p, the chance of getting 3 random values g^a, g^b, g^c such that $g^c = g^{ab}$ is very low unless the random number generator is tampered.

Problem 8 (9 pts) Hash Collisions and Digital Signatures. In a brute-force attack to find collisions for a hash function with a n-bit output, one chooses a group of $m = 2^{n/2}$ different messages. The probability that among them there are two messages that have the same hash is quite high, because the m messages give $\frac{m(m-1)}{2}$ pairs, each of which having the probability $\frac{1}{2^n}$ to give a collision.

a (4 pts) Given two groups X and Y of messages, we aim to finding a collision between a message in X and a message in Y. Suppose that each of X and Y contains $m = 2^{n/2}$ messages, what is the probability that at least one collision between a message in X and one in Y exists? Write out the formula, and compute/estimate it for $n = 128$.

P(collision exist) $= 1 - e^{\frac{-m(m-1)}{2n}} = 1 - e^{\frac{-2\times 2^{\frac{128}{2}}\left(2\times 2^{\frac{128}{2}}-1\right)}{2\times 128}} = 1 - e^{\frac{-2^{65}(2^{65}-1)}{256}}$ = very close to 1

b (5 pts) Suppose that Bob is preparing a contract that Alice purchases a product from Bob at a cost of 1 million dollars for Alice to digitally sign. Bob wants to generate also a contract that sets the price at 10 million dollars. Bob is willing to put in $O(2^{n/2})$ amount of computation for the chance to cheat 9 million.

More specifically, Bob can apply the hash function $2 \times 2^{n/2} = 2^{n/2+1}$ times, which is still $O(2^{n/2})$. In addition, Bob can run an algorithm that takes time $O(n2^{n/2})$. For all practical purposes, we can view $O(n2^{n/2})$ as just $O(2^{n/2})$, since n is small as a multiplication factor.

Describe a way for Bob to come up with two versions of the contracts with the two different prices, but have the same hash value. For simplicity, let us assume that the document is a long and wordy (as legal documents are) ASCII text file. You will need to generate many candidates for the contracts. Describe how you can systematically generate the ones you need.

To cheat, Bob needs to utilize a hash function with smaller n-output bits to secure a higher chance of collision so his fraudulent document can pass the hash integrity check. He may first find the hash value of the original document, then bruteforce for another value producing the same hash, then pad the fraud document so its ASCII values matches the new number found. After running the hash function, collision should happen and the fraud document can stand in the new document's place.

Problem 9 (9 pts) The UNIX `crypt` function is a hash function that only looks at the first eight bytes of the input message. For example, `crypt(helloworld)` returns the same value as `crypt(hellowor)`.

Some web sites use the following authentication method to authenticate users: (1) the user types in a user-id and a password P into his web browser, (2) the site, upon verification of the password P, computes $T =$ `crypt(`user-id $\|$ K`)`, where $\|$ denotes string concatenation, and K is a l-byte site secret key $l \leq 8$, (3) the site sends a cookie back to the user containing T, (4) the user can use T to authenticate himself to the site in future connections.

Show that by choosing clever user-id's (of varying length) an attacker can expose the site's secret key K in time approximately $128l$. More concretely, the user creates an account, logs in and receives the corresponding T; he then creates another account (with a different user-id, logs in and receives another T. By repeating this sufficient times, the user recovers K completely. We are assuming there are 128 possible values for each character in a string.

Hint: Try to recover one character of K for each account created. The attack is described in the paper "Dos and Don'ts of Client Authentication on the Web" in USENIX Security Symposium, 2001. Reading the paper is allowed.

- Since the key space of crypt() is DES-based, it only has 2^56 possible keys. Record all possible keys

- Obtain the returned cookie T using a user ID of length 1
- T' should now return a result based on the first 7 characters of K from the 2nd character on.
- Repeatedly obtain the returned cookies using a user ID of length 8, where the first 4 character remains unchanged, but the last 4 characters are chosen in order of {AAAA, AAAB,… to ZZZZ}.
- Check whether the returned cookies has repeated portion. If it does, mark down the key used in the conversion.
- Since the key used in conversion is found, it can be used to decrypt the rest of key from cookie to plaintext. K is then busted.

Problem 10 (16 pts) Read the article "New Directions in Cryptography" by Diffie and Hellman (`dh.pdf` is attached in eLearning), and answer the following questions. [To get a good grade in this question make sure your answer is short and directly to the point. Bullet-points answers are recommended.]

a (4 pts) The paper gives rationales for building encryption schemes that are secure against known plaintext attacks and chosen plaintext attacks, by discussing how such schemes remove restrictions that are placed on the ways of using them. Discuss these rationale in your own words.

- Having a secure system against known plaintext attack frees the user of the system form paraphrasing the secret messages before declassification and always keeps the messages safe.
- Chances for a known plaintext attack is frequent enough to ruin the security of a system
- Having a cipher that stands against chosen plaintext attack frees its user from worrying about attackers planting their messages in the security system.

b (4 pts) List all the limitations and shortcomings discussed in the paper about symmetric encryption schemes.

- Public key cryptosystem:
  - Enciphering and deciphering plaintext requires more operation and involves hard problem such as matrix inversion.
  - Enormous block size is required during enciphering and deciphering
  - Time required is n^3, greater than cryptanalytic n, making results hard to output.
- Public key distribution system:
  - Cryptanalytic cost grows as n^2, where n is the cost of legitimate user.
  - Cost to legitimate users of the system is much in transmission time as in computation
  - Cost ratio is too small for most application.

c (4 pts) List all the limitations and shortcomings discussed in the paper about symmetric message authentication schemes.

- One-way message authentication, Leslie Lamport's solution
  - 100-fold data expansion required
- One-way message authentication, giving system result of one-time function of evaluated at X = key

- Require long computation for legitimate login
- Both the user and the system must then iterate f an average of 1.3 million times per login.

d (4 pts) The paper establishes the relationships among (1) public-key encryption, (2) public key distribution, and (3) digital signature (referred to in the paper as one-way authentication). By relationships, we mean it is possible to use one scheme to implement another. List these relationships, and explain the constructions involved to use one scheme to implement another.

- A cryptosystem which is secure against a known plaintext attack can be used to produce a one-way function.
  - Take the cryptosystem $\{S_K : \{P\} \rightarrow \{C\}\}_{K \in \{K\}}$ which is secure against a known plaintext attack, fix P = PO and consider the map f:{k}->{C} defined by $f(X) = S_X(P_0)$
  - This function is one-way because solving for X given f(X) is equivalent to the cryptanalytic problem of finding the key from a single known plaintext-cryptogram pair. Public knowledge of f is now equivalent to public knowledge of S_K and P_0.
  - The converse does not always hold, while it is possible to find one-way functions to yield a good cryptosystem.
- A public key cryptosystem can be used to generate a one-way authentication system.
  - The converse does not appear to hold, making the construction of a public key cryptosystem a strictly more difficult problem than one-way authentication.
  - Since in a public key cryptosystem the general system in which E and D are used must be public, specifying E specifies a complete algorithm for transforming input messages into output cryptograms. As such a public key system is really a set of trap-door one-way functions, whose functions are an algorithm for the forward function, which is computationally infeasible.