CS 4235/6035 Introduction to Information Security
Homework One - Symmetric Cryptography
Spring, 2016
Due at Start of Class on Thursday, 28 January

You may collaborate on this homework, but you must list your collaborators. Each student should return a submission. If you use any resources, cite them! Please print this document and write your answers in the space provided. Show enough work that we can give partial credit.

Problem Zero (0 Points)

Your Name: _____ Your GTID: _____

Collaborators: _____

Problem One (2 Points)

A. Assume that an encryption scheme is used to hide the information in a message and that the scheme is flawless, *i.e.* the only way to uncover any information about the plaintext is by searching the key space until you find the key. If the size of the key space is $|K|$, about how many trials will it take to have a 50% chance of finding the key? (Hint: This is an easy question.)

$\frac{|K|}{2}$ or 2^{K-1}

B. With specialized hardware it is quite conceivable that one could build a system that can try 2^{43} keys per second (some Bitcoin mining systems hashing speeds around this order of magnitude). What is the time (in years) to find a key by *exhaustive* search of the key space if the length of the key is 88 bits? 112 bits? 256 bits? (Protip: Use Google to convert seconds to years.)

88 bits: $2^{88}/2^{43} = 2^{45}$ seconds $\approx 1.11 \times 10^{6}$ years

112 bits $2^{112}/2^{43} = 2^{69}$ seconds $\approx 1.86 \times 10^{13}$ years

256 bits $2^{256}/2^{43} = 2^{213}$ seconds $\approx 4.125 \times 10^{56}$ years

C. AES-based cryptosystems generally require 128-bit, 192-bit, or 256-bit keys. Why might one choose a longer key over a shorter key, even if the shorter key would in theory be long enough to protect the ciphertext for millions of years to come?

① Brute-forcing a shorter key is easier ∴ and computers are getting more powerful.

② ~~AAA~~ Shorter keys are usually more vulnerable to more advanced attacks.

③ To comply with certain standards or policy of security.

⋮

Problem Two (2 Points)

Suppose that you need to encrypt a single 1,024-bit message with a 128-bit key.

A. Is it possible to achieve true perfect secrecy within these constraints? Why or why not?

No. True perfect secrecy requires a key at least as long as the message itself. $|k| \geq |m|$

B. Assuming you have a block cipher that, when given a 128-bit key, yields an instance of a true pseudorandom permutation, say how you can use that block cipher within the above constraints to achieve "practically" perfect secrecy.

We can break the message into blocks of 128-bits. Use the block cipher in a secure mode (CBC, CTR, etc) to encrypt the message

Problem Three (2 Points)

A. The CIA keeps a database of every informant in The Democratic Republic of Outtheresomewhere. The database is kept secret, but for added security the name of each informant is hashed using a cryptographically secure hash function. The hash value becomes the informant's ID number and the real name is discarded. If The Democratic Republic of Outtheresomewhere obtains a copy of the database, they can probably reveal the names of most informants. How?

D.R.O. can perform forward search. They can try public hashing functions to hash the name of their citizens. and then compare the digests to the ones in the database.

B. Imagine that we have a hash function that is collision resistant, but that is not one way. Explain why such a function cannot exist. (Your explanation should constitute the general idea behind a proof that collision resistance implies one-wayness. Consider only hash functions whose domain is much larger than their range.)

For hash functions not one-way, it is possible to find a pre-image x s.t. $X = H(x)$, given $H(x)$. For hash functions with a sufficiently large domain, many such pre-image x' must exist s.t. $x' = H(x)$. Thus, an algorithm $\mathcal{A}$ which can find $x = H(x)$, given $H(x)$ can likely also find some x' from the set of many possible pre-images s.t. $H(x) = H(x')$. This is by definition a collision, so the hash is not CR. Not one-way means collision prone, so no CR + OW hash function exists.

Problem Four (2 Points)

For this question you are expected to use outside sources. (Be sure to cite them!) Research a technique used to strengthen DES call triple DES (3DES).

A. Diagram the encryption and decryption process for a single 3DES block. Be sure to include all elements of the diagram, such as what key(s) to use where.

Choose keys $k_1 \neq k_2$

$$m \rightarrow E_{k_1}(\cdot) \rightarrow D_{k_2}(\cdot) \rightarrow E_{k_1}(\cdot) \rightarrow c$$

$$c \rightarrow D_{k_1}(\cdot) \rightarrow E_{k_2}(\cdot) \rightarrow D_{k_1}(\cdot) \rightarrow m$$

B. Explain why double encryption (2DES?) is not used. (Hint: Research "meet-in-the-middle" attacks.)

A meet-in-the-middle attack performs keyspace brute-force on $E_{k_1}(\cdot)$ and $D_{k_2}(\cdot)$, and looking for matches in the intermediate subcipher. The attacker needs perform 2^{k+1} operations to get the key. (2^{57} ops compared to DES 2^{56}). A known plaintext/cipher text pair is required.

Problem Five (2 Points)

In class we discussed message authentication codes. Informally speaking, a MAC is secure if no efficient forger can generate any valid (message, tag) pair, even if the forger is given a tagging oracle. (As we discussed in class a tagging oracle is an algorithm that will tag arbitrary messages for an attacker.)

Consider a MAC technique called CBC-MAC. The algorithm takes a message, m, a secret key, k, and runs CBC mode encryption (as we discussed in class) on the blocks of the message. For purposes of this problem the initialization vector will always be zero. The tag is the final block of the ciphertext.

Demonstrate that this MAC scheme is not secure for variable length messages. Write pseudocode for an algorithm, F, that, given a tagging oracle $T_k(m)$, can easily generate a (message, tag) pair for a message that was not sent to the tagging oracle. (Hint: I can think of a way to make a forged message using only two queries to the tagging oracle.)

Define $A()$:

$m_1 \leftarrow \{0\}^l$

$T_1 \leftarrow T_k(m_1)$

$T_2 \leftarrow T_k(T_1)$

Return $m_2 = \{0\}^{2l} \mid T_2$ as msg/tag

Problem Six (2 Points)

Alice and her brother, Bob, are can't agree on the main course for their family Christmas dinner. Alice wants a ham and Bob wants a turkey. Alice lives in Dallas and Bob lives in Atlanta, so they decide to use cryptography to perform a "coin flip" over the phone. Here is the protocol that they use.

1. Alice picks a secret bit, $b \in \{0, 1\}$.
2. Alice picks a secret key, k, and use a block cipher to compute $c = E_k(m)$ where m is b padded with random bits to the correct block length (the first bit of m is b).
3. Alice reads the bits of c to Bob over the telephone.
4. Bob tries to guess the value of b. Call his guess b'. Bob tells Alice his b' over the phone.
5. Alice reads k to Bob. Bob decrypts c to recover b (the first bit of m).
6. If $b' = b$, Bob wins and they have turkey, otherwise they have ham.

A. Describe how Alice can cheat.

Alice can easily uses another key k', that $D_{k'}(c) \rightarrow m'$, where the first bit of m' is opposite of m. Then, Alice can tell Bob k' in step 5 to make sure Bob is wrong with the expected bit.

B. Describe how a one-way hash function can prevent this attack.

Alice must provide the hash of key and tell it to Bob in step 3. Then Bob can verify that Alice does not cheat in step 5 by comparing the hash Alice gives to him and his own calculation of $H(k)$

Problem Seven (2 Points)

This problem demonstrates that block length is important to the security of a block cipher. Suppose that I have a block cipher whose key size is 128 bits, but whose block length is eight bits. Let $E(m)$ be an encryption oracle that encrypts by running our "short" block cipher in CBC mode. Write pseudocode for an adversary, $A(c)$, that, given $E(m)$, can probably decrypt any input in a reasonable amount of time using modern hardware. (Hint: The m that A passes to E need not be based on the c given to A.)

An 8-bit block length is short enough to learn PRP, because all possible permutations can be computed and referenced easily.

Use $E(\cdot)$ to find all plaintext / ciphertext pairs, and store to a table T.

P_n / C_n is the nth block of P/C.

$P_n = T(C_n) \oplus C_{n-1}$

For P_0, xor $T(C_0)$ with all 2^8 possible IVs to get P_0.

Return $m = P_0 | P_1 | \cdots | P_n$

Problem Eight (3 Points)

Recall our definition of perfect secrecy. An encryption scheme, E, is perfectly secret if for any pair of messages m_1 and m_2

$$Pr_{k \leftarrow K}[E_k(m_1) = c] = Pr_{k \leftarrow K}[E_k(m_2) = c]$$

where c is an element of the ciphertext space. This says that, given some ciphertext, any message in the message space is equally likely to be encrypted message. The intuition is that our encryption scheme hides all of the information contained in the original message.

An equivalent formulation of perfect secrecy, sometimes referred to as *adversarial indistinguishability,* is commonly used to prove or disprove the security of an encryption scheme. This definition is based on an experiment. The experiment uses an adversary, A, whose job is to distinguish the encryption of two messages, m_1 and m_2 (note that A is an algorithm). The intuition is that if A can tell which of the two messages is encrypted, some information must be leaking. A uses an encryption oracle, $O(m_1, m_2)$, which is another algorithm. During the experiment O always encrypts either its first input or its second input with a secret key k. Note that the same input (first or second) is always encrypted during the experiment and that k is chosen randomly at the beginning of the experiment and never changes throughout the experiment. The steps of the experiment are:

1. The experiment is initialized with the random choice of a secret bit b, and a secret key k. These secrets are given to O, but are never revealed to A.

2. A formulates two messages m_1 and m_2. These messages may be any two messages of A's choosing as long as the messages are in the message space *and are of the same length.* (The length may be different for subsequent oracle queries, but for each query $|m_1|$ must equal $|m_2|$.)

3. A calls $O(m_1, m_2)$. O returns ciphertext c to A. c is computed using the encryption scheme in question, the bit b, and the key, k. If $b = 0$, m_1 is encrypted. If $b = 1$, m_2 is encrypted. Remember that A's job is to guess b based on c.

4. A may return to step two any reasonable number of times. Remember that every call to O uses the same k and b.

5. After some reasonable running time, A's job is to guess b, *i.e.* A is trying to determine if O always encrypts the first input or the second. If we can demonstrate that A can find b with a probability notably better than 50% (better than random guessing), we have proven that the encryption scheme fails to be secure under an adaptive chosen plaintext attack.

(We have omitted some formalities from our definitions above, which is a very dangerous thing to do in cryptography. You can find the details and a proof of the equivalence to perfect secrecy in the Katz & Lindell text referred to in the syllabus. For the purposes of this homework assignment, we will allow some relaxation in formalities so that you can learn the general idea. The experiment above uses the terms *reasonable* and *notably better* informally, but these terms allow us to relax information theoretical perfect secrecy to "computationally good enough" secrecy.)

Recall that ECB mode encryption applies a block cipher block-by-block. Write the pseudocode of an efficient adversarial algorithm, A, that can always find b when an otherwise good block cipher is run in ECB mode. (Hint: I can answer this question in about three lines of pseudocode. Also, there is no constraint that says that m_1 must be different from m_2 and there is no constraint that says A can't send the same message(s) to O multiple times or in different positions.)

Algorithm A:

Fix $m \in$ Msg Space

$C_0 \leftarrow O(m, m)$

$C_1 \leftarrow O(m, \bar{m})$ $\bar{m} \neq m$,

if $C_0 = C_1$ return 0, else return 1.

Problem Nine (3 Points)

This problem uses the adversarial indistinguishability experiment of the previous question and underscores the critical importance of cryptographically secure random number generators in encryption implementations. When implemented correctly, CBC mode is provably secure against adaptive chosen plaintext attacks. Suppose that I implement CBC mode with a poor random number generator such that *A* can often guess (with 1% probability of guessing correctly) the IV that *O* will use when it encrypts. Write pseudocode for an adversarial algorithm, *A*, that can find *b* using a reasonable amount of computing resources. (Hint: My solution has six lines of pseudocode.)

Algorithm A:

get $C_0 \leftarrow O(|0|^{|IV|}, |0|^{|IV|})$

Store IV from C_0 as i.

Guess next IV as g ←

get $C \leftarrow (g \oplus i, \overline{g \oplus i})$

if IV from $C \neq g$, go back

if $C = C_0$ return 0, else return 1

In this way correctly guessing the IV will cause the oracle to run $E_k(i \oplus g \oplus IV)$ where $g \oplus IV = 0$, which is equivalent to what it encrypted to find C_0 $(E_k(i = IV \oplus 0))$.

Compare ciphertexts for (in) equality to determine b