MODERN CRYPTOGRAPHIC ALGORITHMS

LAB HANDOUT

Summer 2022

Lab Instructions

1. There are 4 labs for this course. Each lab is 20% of your final grade.
2. Each lab is expected to be completed within 2 lab sessions (i.e., 6 hours). However, extra time is given to submit the final lab report, allowing you to complete any pending lab tasks at your own time.
3. All lab work must be your own, unless specified. There are labs that could be completed as a team (more on this during the lab sessions).
4. All lab reports must be in pdf format and submitted in Dropbox. Each one of you will be given access to your own Dropbox folder. It is important to email me (ioanna.dionysiou@gmail.com) your name and student ID to proceed with the folder creation by Tuesday, May 31, 2022, the latest.

Lab Tentative Schedule

Lab	Title	Topics	Deadlines
1 (Sessions 1-2)	Comprehensive Exercise Set Lab	• Exercises and problems covering cryptographic topics discussed during the lectures	Out: 2/6 Due: 7/6
2 (Sessions 3-4)	Secret Key Encryption Lab https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_Encryption/	• Secret-key encryption • Substitution cipher and frequency analysis • Encryption modes, IV, and paddings • Common mistakes in using encryption algorithms • Programming using the crypto library	Out: 13/6 Due:17/6
3 (Sessions 5-6)	MD5 Hash Collision Lab https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_MD5_Collision/	• One-way hash function, MD5 • The collision-resistance property • Collision attacks	Out: 17/6 Due:22/6
4 (Sessions 7-8)	Student Choice	• Lab from the Cryptography Labs category (SeedLabs 2.0) • Student proposal to implement cryptographic algorithms or use cryptographic libraries in a program	Out: 22/6 Due:26/6

LAB 1
Comprehensive Exercise Set

Lab Description

Your task is to provide solutions to selected exercises[1] and problems from two textbooks:

- Wenliang Du, Computer & Internet Security: A Hands-on Approach, 2nd edition, 2019
- William Stallings, Cryptography and Network Security: Principles and Practices, 7th edition, Global edition, 2017

Exercise Set from Wenliang Du Book
(5 points each – total 35 points possible)

21.15 Alice encrypts a message using AES and sends the ciphertext to Bob. Unfortunately, during the transmission, the 2nd bit of the third block in the ciphertext is corrupted. How much of the plaintext can Bob still recover if the mode of encryption is one of the following: CBC, CFB, OFB, or CTR?

22.4 An attacker gets a copy of the shadow file, and he tries to guess Bob's password. It was said that the salt value makes it much more difficult for the attacker to do so. Do you agree or not?

22.5 Currently, the salt used in the shadow file is saved in the file in plaintext. Isn't it better not to save the salt in plaintext? Please explain.

22.7 In Linux, the password hash is produced by applying a hash function for many rounds (e.g., 5000 rounds for SHA-512). This seems to waste time. Why does Linux do this?

22.13 Charlie has arranged a blind date for Alice and Bob, who are both cryptographers, and they do not know each other before. Charlie also gave Alice and Bob a secret number K (nobody else knows K). Bob wants to make sure that the person he is dating is Alice, not somebody else. Please describe how Bob can ask Alice to securely prove that she is Alice (Alice will not reveal the secret number K to anybody).

23.1 In the Diffie-Hellman key exchange, Alice sends $g^x \bmod p$ to Bob, and Bob sends $g^y \bmod p$ to Alice. How do they get a common secret?

23.4 Let $n = 2419 = 41 * 59$ and $e = 7$. (1) Find the private key d. (2) Encrypt the message 6. (3) Sign the message 10. Assume RSA is used. You only need to show the formula; there is no need to calculate the final results.

[1] Note that the exercise number is as shown in the book.

Exercise Set from William Stallings Book
(10 points each – total 60 points possible)

3.1 A generalization of the Caesar cipher, known as the affine Caesar cipher, has the following form:

For each plaintext letter p, substitute the ciphertext letter C as shown below

$$C = E([a, b], p) = (ap + b) \bmod 26$$

A basic requirement of any encryption algorithm is that it be one-to-one. That is, if $p \neq q$, then $E(k, p) \neq E(k, q)$. Otherwise, decryption is impossible, because more than one plaintext character maps into the same ciphertext character.

The affine Caesar cipher is not one-to-one for all values of a. For example, for a = 2 and b = 3, then E([a , b], 0) = E([a , b], 13) = 3.

(a) Are there any limitations on the value of b ? Explain why or why not.

(b) Determine which values of a are not allowed.

3.20 This problem explores the use of a one-time pad version of the Vigenère cipher. In this scheme, the key is a stream of random numbers between 0 and 26. For example, if the key is 3 19 5 . . . , then the first letter of plaintext is encrypted with a shift of 3 letters, the second with a shift of 19 letters, the third with a shift of 5 letters, and so on.

(a) Encrypt the plaintext *sendmoremoney* with the key stream
3 11 5 7 17 21 0 11 14 8 7 13 9

(b) Using the ciphertext produced in part (a), find a key so that the ciphertext decrypts to the plaintext *cashnotneeded*

4.2 Consider a Feistel cipher composed of sixteen rounds with a block length of 128 bits and a key length of 128 bits.

Suppose that, for a given k, the key scheduling algorithm determines values for the first eight round keys, k1 , k2 ,…,k8 , and then sets k9 = k8 , k10 = k7 , k11 = k6 , …, k16 = k1.

Suppose you have a ciphertext c. Explain how, with access to an encryption oracle, you can decrypt c and determine m using just a single oracle query. This shows that such a cipher is vulnerable to a chosen plaintext attack. (An encryption oracle can be thought of as a device that, when given a plaintext, returns the corresponding ciphertext. The internal details of the device are not known to you, and you cannot break open the device. You can only gain information from the oracle by making queries to it and observing its responses.

7.7 For the ECB, CBC, and CFB modes, the plaintext must be a sequence of one or more complete data blocks (or, for CFB mode, data segments). In other words, for these three modes, the total number of bits in the plaintext must be a positive multiple of the block (or segment) size. One common method of padding, if needed, consists of a 1 bit followed by as few zero bits, possibly none, as are necessary to complete the final block. It is considered good practice for the sender to pad every message, including messages in which the final message block is already complete. What is the motivation for including a padding block when padding is not needed?

7.11 "*This is a very interesting case, Watson*," Holmes said. "*The young man loves a girl, and she loves him too. However, her father is a strange fellow who insists that his would-be son-in-law must design a simple and secure protocol for an appropriate public-key cryptosystem he could use in his company's computer network*."

The young man came up with the following protocol for communication between two parties. For example, user A wishing to send message M to user B: messages exchanged are in the format (sender's name, text, receiver's name)

1. A sends B the following block: (A, E(PU_B , [M , A]), B)
2. B acknowledges receipt by sending to A the following block: (B , E(PU_A , [M , B]), A)

"*You can see that the protocol is really simple. But the girl's father claims that the young man has not satisfied his call for a simple protocol, because the proposal contains a certain redundancy and can be further simplified to the following*:"

1. A sends B the block: (A , E(PU_B , M), B)
2. B acknowledges receipt by sending to A the block: (B , E(PU_A , M), A)

"*On the basis of that, the girl's father refuses to allow his daughter to marry the young man, thus making them both unhappy. The young man was just here to ask me for help*."
"*Hmm, I don't see how you can help him*." Watson was visibly unhappy with the idea that the sympathetic young man has to lose his love.
"*Well, I think I could help. You know, Watson, redundancy is sometimes good to ensure the security of protocol. Thus, the simplification the girl's father has proposed could make the new protocol vulnerable to an attack the original protocol was able to resist*," mused Holmes.
"*Yes, it is so, Watson. Look, all an adversary needs is to be one of the users of the network and to be able to intercept messages exchanged between A and B. Being a user of the network, he has his own public encryption key and is able to send his own messages to A or to B and to receive theirs. With the help of the simplified protocol, he could then obtain message M user A has previously sent to B using the following procedure:*"

Complete the description.

11.12 This problem introduces a hash function similar in spirit to SHA that operates on letters instead of binary data. It is called the *toy tetragraph hash* (tth).[6] Given a message consisting of a sequence of letters, tth produces a hash value consisting of four letters. First, tth divides the message into blocks of 16 letters, ignoring spaces, punctuation, and capitalization. If the message length is not divisible by 16, it is padded out with nulls. A four-number running total is maintained that starts out with the value (0, 0, 0, 0); this is input to the compression function for processing the first block. The compression function consists of two rounds.

Round 1 Get the next block of text and arrange it as a row-wise 4×4 block of text and convert it to numbers ($A = 0, B = 1$, etc.). For example, for the block ABCDEFGHIJKLMNOP, we have

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Then, add each column mod 26 and add the result to the running total, mod 26. In this example, the running total is (24, 2, 6, 10).

Round 2 Using the matrix from round 1, rotate the first row left by 1, second row left by 2, third row left by 3, and reverse the order of the fourth row.
In our example:

B	C	D	A
G	H	E	F
L	I	J	K
P	O	N	M

1	2	3	0
6	7	4	5
11	8	9	10
15	14	13	12

Now, add each column mod 26 and add the result to the running total. The new running total is (5, 7, 9, 11). This running total is now the input into the first round of the compression function for the next block of text. After the final block is processed, convert the final running total to letters. For example, if the message is ABCDEFGHIJKLMNOP, then the hash is FHJL.

(a) Draw figures to depict the overall *tth* logic and the compression function logic.
(b) Calculate the hash function for the 22-letter message "*Practice makes us perfect*"
(c) To demonstrate the weakness of tth, find a message of length 32-letter to produce the same hash.

4.11 ***(optional)*** This problem provides a numerical example of encryption using a one-round version of DES. We start with the same bit pattern for the key K and the plaintext, namely:

Hexadecimal notation: 0123456789ABCDEF

Binary notation: 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

(a) Derive K1 , the first-round subkey
(b) Derive L0 , R0

(c) Expand R0 to get E[R 0], where E[#] is the expansion function of Table S1
(d) Calculate A = E[R 0] $\oplus$ K 1
(e) Group the 48-bit result of (d) into sets of 6 bits and evaluate the corresponding S-box substitutions.
(f) Concatenate the results of (e) to get a 32-bit result, B .

Exercise (5 points)

Send me your email to create your Dropbox folder!

LAB 2
Secret Key Encryption Lab

Lab Description

This lab belongs to the SEED Labs 2.0 (https://seedsecuritylabs.org/labs.html) and its main objective is to get familiar with the concepts in the secret-key encryption. The site for this lab is https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_Encryption/

The lab tasks are described in great detail in the pdf document https://seedsecuritylabs.org/Labs_20.04/Files/Crypto_Encryption/Crypto_Encryption.pdf. The deliverables are also indicated in the lab description.

Note that you need to create a SEED VM to conduct this lab. Instructions on how to set up the lab environment could be found in https://seedsecuritylabs.org/labsetup.html . It is best to create a SEED VM on your local machine by using the pre-built SEED VM.

LAB 3
MD5 Collision Attack Lab

Lab Description

This lab belongs to the SEED Labs 2.0 (https://seedsecuritylabs.org/labs.html) and its main objective is to understand the impact of collision attacks in hash functions. The site for this lab is https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_MD5_Collision/ .

The lab tasks are described in great detail in the pdf document https://seedsecuritylabs.org/Labs_20.04/Files/Crypto_MD5_Collision/Crypto_MD5_Collision.pdf . The deliverables are also indicated in the lab description.

Note that you need to create a SEED VM to conduct this lab. Instructions on how to set up the lab environment could be found in https://seedsecuritylabs.org/labsetup.html . It is best to create a SEED VM on your local machine by using the pre-built SEED VM.

LAB 4
Student Choice

Lab Description

For this lab, you get to choose what you want to do! You could either complete one more lab from the Cryptography Labs category (https://seedsecuritylabs.org/Labs_20.04/Crypto/) or propose the development of an application that will use a cryptographic library of the programming language of your choice. In the latter case, your proposal needs to be approved by me (send an email with the proposal).