Irving Reyes Bravo
02/07/2025
CS 4920 Spring 2025
HW 1

Problem 1. Security Objectives

For the following scenarios, give examples of confidentiality, integrity, availability, authenticity, and accountability requirements associated with the system. In each case, indicate the degree of importance of the requirements and explain why.

a. An automated teller machine (ATM) in which user provide a personal identification number (PIN) and a card for account access.
   a Confidentiality: [HIGH] Accounts numbers & PINS must be encrypted to stop intrusion.
   b Integrity: [HIGH] Transactions must be accurate, ensuring correct amounts each time.
   c Availability: [MID] ATM should be operational, with enough downtime for maintenance.
   d Authenticity: [HIGH] Users must be authenticated via their PIN & card to prevent fraud.
   e Accountability: [HIGH] Transactions must be logged to track fraudulent, user activities.
b. An implanted defibrillator that uses a remote programmer to perform diagnostics, read/write data, and adjust the treatment settings.
   a Confidentiality: [HIGH] Remote command & patient data are protected against intrusion.
   b Integrity: [HIGH] Incorrect modifications could lead to life-threatening conditions.
   c Availability: [CRITICAL] It must function at all times, as failure could be fatal.
   d Authenticity: [HIGH] Only authorized healthcare providers should be able to modify it.
   e Accountability: [HIGH] Logs should track changes to settings and all access attempts.
c. A domain name system (DNS) which resolves and translates the domain names to IP addresses.
   a Confidentiality: [LOW] DNS queries are generally public, but DoH can improve privacy.
   b Integrity: [HIGH] DNS spoofing can lead to redirection to malicious websites.
   c Availability: [HIGH] DNS outages can offline many services so redundancy is crucial.
   d Authenticity: [HIGH] DNSSEC can verify that responses come from legitimate servers.
   e Accountability: [MID] Logs should track query requests and balance privacy concerns.
d. Read a security news on your own. Describe the scenario and the security incident (1-2 paragraphs), cite the news sources, and solve the problem.
   Novaes Neto, N., Madnick, S., de Paula, M. G., & Malara Borges, N. (2020, January 1). A Case Study of the Capital One Data Breach. MIT Sloan.
   https://www.capitalone.com/digital/facts2019/
   In "A Case Study of the Capital One Data Breach", the authors provide an in-depth analysis, focusing on the technical aspects of the attacks and the effectiveness of existing compliance measures. Despite Capital One's adherence to various security frameworks and regulations, the authors assess the limitations of the requirements and emphasize that mere compliance dies not guarantee security. They detail how the attacker exploited a misconfigured WAF to perform a Server-Side Request Forgery attack. This allowed unauthorized access to sensitive data stored in Amazon Web Service (AWS) S3 buckets. The authors concluded that organizations should go beyond compliance by implementing continuous monitoring and regular security assessments. A vigilant organizational mindset is prudent for fostering a security-centric culture that effectively mitigates cyber threats.
   Data breaches like this demonstrate vulnerabilities in Cloud Security, misconfigured access controls, and insider threats. Besides increased regulatory scrutiny and expensive remediation

efforts, organizations may find that customer confidence, once data is exposed, erodes, leading to losses in business. This loss of trust can reduce market share and consumer confidence but also induce physiological stress to the victims of the breach. Capital One was fined $80 million by U.S. regulators and spent millions on security upgrades and settlements. The breach damaged the company's reputation, leading to customer skepticism about its ability to protect sensitive data. Their preventative measures focused on technical defenses (zero-trust architecture), security awareness/training (automated compliance monitoring), and governance policies (incident response plans).

a Confidentiality: The breach compromised personal information, like Social Security numbers and bank account details, which led to identity theft and financial fraud.

b Integrity: While no indication of data alteration was found, the unauthorized access undermined the integrity of Capital one's data security measures.

c Availability: This breach was so severe because it did not appear to affect the availability of Capital One's services; most if not all operations continued without interruption.

d Authenticity: Compromised data can be used in phishing attacks impersonating the bank.

e Accountability: Capital One faced legal consequences, including an $80 million fine.

To mitigate the risk of similar incidents in the future, organizations should implement robust security configurations while conducting regular security audits to enhance monitoring and detection methods. Additionally, it would help to provide employee training on safeguarding sensitive data and develop incident response plans to address potential breaches effectively.

Problem 2. Affine Caesar Cipher

A generalization of the Caesar cipher, known as the affine Caesar cipher, has the following form: for each plaintext letter p (where p can be an integer between 0 and 25 inclusive), substitute the ciphertext letter C:

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

A basic requirement of any encryption algorithm is that it needs to be one-to-one, i.e., if $p \neq q$, then $\mathrm{E}(k,p) \neq \mathrm{E}(k,q)$ where the key $k = [a,b]$. 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 $\mathrm{E}([a,b],0) = \mathrm{E}([a,b],13) = 3$.

a. Are there any limitations on the value of b for the affine Caesar cipher to be one-to-one? Explain.
The function C = (ap + b) mod 26 is a bijection transformation, where a determines the scaling (or the multiplication) and b determines the shift (or the addition). For the function to be one-on-one, the multiplication factor a must have an inverse modulo 26 so that the description function, p = a^{-1} (C – b) mod 26 is well-defined. Since addition is always reversible, any value of b from 0 to 25 is valid. Thus, there are no limitations on b but on a, which must be comprise ton26 (i.e., gcd(a, 26) = 1) to ensure that every plaintext letter maps uniquely to a cipher text letter.

b. What are the values of b providing distinct mappings for the affine Caesar cipher? <u>Hint:</u> Because of the mod-26 operation, some b provide equal mapping outputs for the affine Caesar cipher.
The valid and distinct values of b are all integers from 0 to 25. Since b acts as a shift, and shifts wrap around after 26, any values of b that differ by 26 (or multiples of 26) are equivalent. Hence, there are exactly 26 distinct values of b, one for each possible shift.

c. Determine which values of a are not allowed. Provide a general statement of which values of a are and are not allowed. Justify your statement.
Since the affine Caesar Cipher is a bijection transformation, requiring it to be invertible, meaning there must exist a multiplicative inverse for a modulo 26. For this to be true, a must be co-prime to 26: GCD(a, 26) = 1. Otherwise, some plaintext letters will map to the same ciphertext letter.

Since 26 factors as (2 * 13), any *a* that shares a common factor with 26 is not allowed. The numbers in the range [0 – 25] that are divisible by 2 or 13 are: 0, 2, 4, 6, 8, 10, 12, 13, 14, 16, 18, 20, 22, and 24. These 14 values are not allowed because they don't have a modular inverse modulo 26. Thus, the remaining 12 values (1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, and 25) are valid because they are co-prime to 26 (i.e., GCD(a, 26) = 1.

d. How many keys are there for distinct affine Caesar cipher? Ordinary Caesar cipher has 26 keys. Each valid choice of *a* can be paired with any choice of *b*, so the total number of distinct keys is:
Total Keys = (# of valid a) * (# of valid b)
= 12 * 26
= 312 distinct keys
The ordinary Caesar Cipher is a special case where *a* = 1, so the only key component is *b*, giving us 26 keys. The affine Caesar Cipher allows for multiple values of *a*, leading to 312 unique keys, making it significantly more complex than the ordinary Caesar Cipher.

e. A long English book has been encrypted using an affine Caesar cipher. The most frequent letter of the ciphertext is "C", and the second most frequent letter of the ciphertext is "D". Break this code to find the key.
Given: The most frequent letters in the ciphertext are "C" followed by "D".
The most common letters in English text are "E" followed by "T" according to UND.
Assume: That the affine Cipher maps E to C and T to D.
Steps:
1) Convert Letters to Numbers
Using A = 0, …, Z = 25, I map E to 4, T to 19, C to 2, and D to 3. Thus my cipher is:
(4a + b) mod 26 = 2
(19a + b) mod 26 = 3
2) Solve for *a* and *b*
I now subtract the two equations to solve for both variables:
(19a + b) – (4a + b) = 3 – 2 mod 26
15a = 1 mod 26
3) Find Modular Inverse of 15 Modulo 26
GCD(15, 26) = 1
Using the Extended Euclidean algorithm, the modular inverse of 15 mod 26 is 7.
Thus: a = 7(1) mod 26
4) Solve for *b*
Substituting *a* into the first equation:
(4 * 7 + b) mod 26 = 2
(28 + b) mod 26 = 2
Thus: b = 2 – 28 = -26 = 0 mod 26

Final Answer: The Key is (a, b) = (7, 0)
The encryption function used is: C = (7p + 0) mod 26
The decryption function used is: p = (15(C – 0)) mod 26
p = (15C) mod 26

Problem 3.

a. Use Vigenere cipher to encrypt the plaintext "sendmoremoney" with the given key stream.
Steps:
1) Convert Letters to Numbers
Using A = 0, B = 1, …, Z = 25, I know that:

s	e	n	d	m	o	r	e	m	o	n	e	y

18	4	13	3	12	14	17	4	12	14	13	4	24

2) Apply the Keystream
   The keystream values given are:

9	0	1	7	23	15	21	14	11	11	2	8	9

   I can now compute the ciphertext:
   C = (P + K) mod 26

18	4	13	3	12	14	17	4	12	14	13	4	24
9	0	1	7	23	15	21	14	11	11	2	8	9
1	**4**	**14**	**10**	**9**	**3**	**12**	**18**	**23**	**25**	**15**	**12**	**7**
B	**E**	**O**	**K**	**J**	**D**	**M**	**S**	**X**	**Z**	**P**	**M**	**H**

3) Final Ciphertext
   The encrypted text is: "BEOKJDMSXZPMH".

b. Using the ciphertext produced, find a key so that it decrypts to the plaintext "cashnotneeded".
   Steps:
   1) Convert Letters to Numbers
      Using A = 0, B = 1, …, Z = 25, I know that:

B	E	O	K	J	D	M	S	X	Z	P	M	H
1	4	14	10	9	3	12	18	23	25	15	12	7

C	A	S	H	N	O	T	N	E	E	D	E	D
2	0	18	7	13	14	19	13	4	4	3	4	3

   2) Compute the Key
      I can now calculate the key:
      K = (C – P) mod 26

1	4	14	10	9	3	12	18	23	25	15	12	7
2	0	18	7	13	14	19	13	4	4	3	4	3
25	**4**	**22**	**3**	**22**	**15**	**19**	**5**	**19**	**21**	**12**	**8**	**4**
Z	**E**	**W**	**D**	**W**	**P**	**T**	**F**	**T**	**V**	**M**	**I**	**E**

   3) Final Key
      The key that decrypts the ciphertext into the given plaintext is: "ZEWDWPTFTVMIE".

c. Can a brute-force attacker decrypt the ciphertext from Part a. so that it knows the original message in a deterministic manner?
   Previously, the keystream was 13 characters long. Each key element is a shift value from 0 to 25, meaning there are 26^{13} possible key combinations.
   26^{13} ~= 2.56 (10^{18})
   This is a large key space, making exhaustive brute-force search infeasible for most attackers. If the keystream were short and repeating, an attacker could exploit Kasiski examination or index of coincidences techniques to determine the key length. However, if the key is as long as the plaintext (one-time pad scenario), brute-force decryption does not deterministically recover the original message. To counter this, the keystream should be long, random, and not reused.

Problem 4. Cipher Programming: Caesar Cipher

Write a program that can encrypt and decrypt using the general Caesar cipher described in the Textbook Section 3.2 (not the Affine Caesar Cipher). Ignore the non-letter symbols (such as "." And "").

a. Describe your program/implementation:
   The Caesar cipher program encrypts and decrypts text by shifting each letter by a given key using modular arithmetic, following Section 3.2 of Stallings' book, which defines classical substitution ciphers. The function caesar_cipher(text, key, mode="encrypt") processes each character in the

input string, applying the shift formula C=(P+k)mod26 for encryption and P=(C−k)mod26 for decryption. Case sensitivity is preserved by determining if a letter is uppercase (ord('A') = 65) or lowercase (ord('a') = 97), then applying the shift within the respective range. Non-letter symbols (such as spaces and punctuation) are ignored, maintaining their original form in the output. The implementation uses Python's built-in ord() and chr() functions for ASCII conversions and isalpha() to filter letters. By handling only alphabetic characters and using modular arithmetic to keep shifts within the 26-letter range, the code ensures proper encryption and decryption while aligning with the theoretical principles outlined in *Cryptography and Network Security*.

b. Encrypt the text in the class input file ("class_input_b.txt") provided for this assignment. Generate the ciphertext in txt file. (1 file)
c. Decrypt the text in the class input file ("class_input_c.txt") provided for this assignment. Generate the decrypted plaintext in txt file. (1 file)
d. Generate one input test file on your own and encrypt it to generate the corresponding output file. The input file should have at least 50 letters for the message plaintext. Also, verify that the decryption results back in to the plaintext message. (2 files)

Problem 5. Cipher Programming: Hill Cipher

Write a program that can encrypt and decrypt Hill cipher. For your hill cipher, append "x" to the end of the message if needed and ignore the non-letter symbols (such as "." and "").

a. Describe your program/implementation:
   This Python program implements the Hill Cipher encryption and decryption using **NumPy** for matrix operations. The program preprocesses the text by converting it to uppercase and removing non-alphabetic characters while storing the positions of spaces to maintain the original formatting after decryption. The encryption function (encrypt()) converts the text into numerical values, groups them into blocks matching the key matrix size, and performs matrix multiplication modulo 26. If necessary, "X" is appended as padding to ensure divisibility. The decryption function (decrypt()) computes the modular inverse of the key matrix and then applies it to the encrypted numerical values. Finally, it restores the spaces before returning the final text. This ensures case insensitivity, as the input is converted to uppercase and spaces are ignored during encryption but reinserted in decryption. The program validates the key matrix to ensure its determinant is invertible modulo 26, preventing decryption errors.
b. Encrypt the text in the class input file ("class_input_b.txt") provided for this assignment. Generate the ciphtertext in txt file. (1 file)
c. Decrypt the text in the class input file ("class_input_c.txt") provided for this assignment. Generate the decrypted plaintext in txt file. (1 file)
d. Generate one input test file on your own and encrypt it to generate the corresponding output file. The input file should have at least 50 letters for the message plaintext. Also, verify that the decryption results back in to the plaintext message. (2 files)