CSIT 6000Q - Blockchain and Smart Contracts
Assignment 1 with Solutions

Problem 1: Transposition Ciphers (15 points)

Transposition ciphers rearrange the characters of the plaintext to form the ciphertext. Use the given key to encrypt and decrypt the following text using the columnar transposition cipher method.

You can use the following online tool for this question: https://crypto.interactivemaths.com/columnar-transposition-cipher.html

Problem 1.1 (7.5 points)

Encrypt the following message using a 5-column transposition cipher:

`"Blockchain technology provides a decentralized ledger for secure transactions."`

Solution 1.1

If we use the key: `zmuxv`, we get the following ciphertext:
`LHELRECADESESOOACOOSELLRETANKNNYIDTZDOUATCIHGVANIEFCRCSBCTOPDEREGRRNI`

Problem 1.2 (7.5 points)

Decrypt the following ciphertext that was encrypted using the following 4-column transposition cipher key: `pmwa`

`"GTGYIGSIVANTIGRMFNONAHESONIIONRLBNIVTWLLMETRELIVLNNELGISEYEAENDEATRLNETEFIURNEIELSAL"`

Solution 1.2

By using the given key, the given ciphertext decrypts to:
`thegreatestgloryinlivingliesnotinneverfallingbutinrisingeverytimewefallnelsonmandela`

If we clean it up by introducing spaces appropriately, we get the following message:

"The greatest glory in living lies not in never falling but in rising every time we fall." -Nelson Mandela

Problem 2: Frequency Analysis of Substitution Cipher (20 points)

The following ciphertext has been encrypted using a substitution cipher. Using the frequency distribution of English letters and common digraphs, decrypt the message and explain your method:

```
"Z OLMT GRNV ZTL, RM Z TZOZCB UZI, UZI ZDZB...  RG RH Z WZIP GRNV ULI
GSV IVYVOORLM. ZOGSLFTS GSV WVZGS HGZI SZH YVVM WVHGILBVW, RNKVIRZO
GILLKH SZEV WIREVM GSV IVYVO ULIXVH UILN GSVRI SRWWVM YZHV ZMW KFIHFVW
GSVN ZXILHH GSV TZOZCB. VEZWRMT GSV WIVZWVW RNKVIRZO HGZIUOVVG, Z TILFK
LU UIVVWLN URTSGVIH OVW YB OFPV HPBDZOPVI SZH VHGZYORHSVW Z MVD HVXIVG
YZHV LM GSV IVNLGV RXV DLIOW LU SLGS. GSV VERO OLIW WZIGS EZWVI, LYHVHHVW
DRGS URMWRMT BLFMT HPBDZOPVI, SZH WRHKZGXSVW GSLFHZMWH LU IVNLGV KILYVH
RMGL GSV UZI IVZXSVH LU HKZXV..."
```

You can use the following online tool for this question: https://www.101computing.net/mono-alphabetic-substitution-cipher/

Solution 2

Using the following key:
`zyxwvutsrqponmlkjihgfedcba`

We get the plaintext as:
"A long time ago, in a galaxy far, far away... It is a dark time for the Rebellion. Although the Death Star has been destroyed, Imperial troops have driven the Rebel forces from their hidden base and pursued them across the galaxy. Evading the dreaded Imperial Starfleet, a group of freedom fighters led by Luke Skywalker has established a new secret base on the remote ice world of Hoth. The evil lord Darth Vader, obsessed with finding young Skywalker, has dispatched thousands of remote probes into the far reaches of space..."

Problem 3: Affine Cipher Encryption and Decryption (25 points)

Affine ciphers are a type of substitution cipher with the encryption function:

$$E(x) = (a \cdot x + b) \mod 26$$

where a and b are the keys and x is the numeric equivalent of a letter in the alphabet.

You can use the following online tool for this question: https://cryptii.com/pipes/affine-cipher

Problem 3.1 (7.5 points)

Encrypt the message "SECUREHASH" using the key $(a = 7, b = 3)$.

Solution 3.1

The encrypted message is: `ZFRNSFADZA`

The affine cipher encryption formula is:

$$E(x) = (a \cdot x + b) \mod 26$$

where:

- $a = 7$
- $b = 3$
- x is the numeric position of a letter in the alphabet $(A = 0, B = 1, C = 2, \ldots, Z = 25)$.

Let's encrypt the message `"SECUREHASH"`.

1. Convert each letter to its numeric equivalent:

   $S = 18$, $E = 4$, $C = 2$, $U = 20$, $R = 17$, $E = 4$, $H = 7$, $A = 0$, $S = 18$, $H = 7$

2. Apply the encryption formula:

   $$E(x) = (7 \cdot x + 3) \mod 26$$

   - $S = (7 \cdot 18 + 3) \mod 26 = (126 + 3) \mod 26 = 129 \mod 26 = 25 \quad (Z)$
   - $E = (7 \cdot 4 + 3) \mod 26 = (28 + 3) \mod 26 = 31 \mod 26 = 5 \quad (F)$
   - $C = (7 \cdot 2 + 3) \mod 26 = (14 + 3) \mod 26 = 17 \mod 26 = 17 \quad (R)$
   - $U = (7 \cdot 20 + 3) \mod 26 = (140 + 3) \mod 26 = 143 \mod 26 = 13 \quad (N)$
   - $R = (7 \cdot 17 + 3) \mod 26 = (119 + 3) \mod 26 = 122 \mod 26 = 18 \quad (S)$
   - $E = (7 \cdot 4 + 3) \mod 26 = (28 + 3) \mod 26 = 31 \mod 26 = 5 \quad (F)$
   - $H = (7 \cdot 7 + 3) \mod 26 = (49 + 3) \mod 26 = 52 \mod 26 = 0 \quad (A)$
   - $A = (7 \cdot 0 + 3) \mod 26 = (0 + 3) \mod 26 = 3 \mod 26 = 3 \quad (D)$

- $S = (7 \cdot 18 + 3) \mod 26 = (126 + 3) \mod 26 = 129 \mod 26 = 25 \quad (Z)$
- $H = (7 \cdot 7 + 3) \mod 26 = (49 + 3) \mod 26 = 52 \mod 26 = 0 \quad (A)$

3. Convert back to letters:
   `"ZFRNSFADZA"`

Encrypted Message: `"ZFRNSFADZA"`

Problem 3.2 (7.5 points)

Decrypt the ciphertext "AFCCXAXBDSFPXN" using the same key.

Solution 3.2

The decrypted message is: `HELLOHOWAREYOU`

Let's decrypt the ciphertext `"AFCCXAXBDSFPXN"` using the key values $a = 7$ and $b = 3$.

Step 1: Decryption formula

The decryption formula for an affine cipher is:

$$D(y) = a^{-1} \cdot (y - b) \mod 26$$

where:

- y is the numeric value of the ciphertext letter ($A = 0, B = 1, C = 2, \ldots, Z = 25$).
- a^{-1} is the modular multiplicative inverse of a modulo 26.
- b is the key value, which in this case is $b = 3$.

Step 2: Find a^{-1} (mod 26)

To find a^{-1}, we need a number such that:

$$7 \cdot a^{-1} \equiv 1 \mod 26$$

The modular inverse of 7 modulo 26 is 15 because:

$$7 \cdot 15 = 105 \equiv 1 \mod 26$$

Thus, $a^{-1} = 15$.

Step 3: Convert the ciphertext into numeric values

Let's convert each letter of the ciphertext `"AFCCXAXBDSFPXN"` into its corresponding numeric value:

- A = 0
- F = 5
- C = 2
- C = 2
- X = 23
- A = 0
- X = 23
- B = 1
- D = 3
- S = 18
- F = 5
- P = 15
- X = 23
- N = 13

So, the numeric representation of the ciphertext is:

$$[0, 5, 2, 2, 23, 0, 23, 1, 3, 18, 5, 15, 23, 13]$$

Step 4: Apply the decryption formula

For each numeric value y, calculate:

$$D(y) = 15 \cdot (y - 3) \mod 26$$

Let's apply the formula to each value:

1. **A = 0**:

   $$D(0) = 15 \cdot (0 - 3) \mod 26 = 15 \cdot (-3) \mod 26 = -45 \mod 26 = 7$$

   7 corresponds to H.

2. **F = 5**:

$$D(5) = 15 \cdot (5 - 3) \mod 26 = 15 \cdot 2 \mod 26 = 30 \mod 26 = 4$$

4 corresponds to E.

3. **C = 2**:

$$D(2) = 15 \cdot (2 - 3) \mod 26 = 15 \cdot (-1) \mod 26 = -15 \mod 26 = 11$$

11 corresponds to L.

4. **C = 2** (again):

$$D(2) = 15 \cdot (2 - 3) \mod 26 = 15 \cdot (-1) \mod 26 = -15 \mod 26 = 11$$

11 corresponds to L.

5. **X = 23**:

$$D(23) = 15 \cdot (23 - 3) \mod 26 = 15 \cdot 20 \mod 26 = 300 \mod 26 = 14$$

14 corresponds to O.

6. **A = 0** (again):

$$D(0) = 15 \cdot (0 - 3) \mod 26 = 15 \cdot (-3) \mod 26 = -45 \mod 26 = 7$$

7 corresponds to H.

7. **X = 23** (again):

$$D(23) = 15 \cdot (23 - 3) \mod 26 = 15 \cdot 20 \mod 26 = 300 \mod 26 = 14$$

14 corresponds to O.

8. **B = 1**:

$$D(1) = 15 \cdot (1 - 3) \mod 26 = 15 \cdot (-2) \mod 26 = -30 \mod 26 = 22$$

22 corresponds to W.

9. **D = 3**:

$$D(3) = 15 \cdot (3 - 3) \mod 26 = 15 \cdot 0 \mod 26 = 0$$

0 corresponds to A.

10. **S = 18**:

$$D(18) = 15 \cdot (18 - 3) \mod 26 = 15 \cdot 15 \mod 26 = 225 \mod 26 = 17$$

17 corresponds to R.

11. **F = 5** (again):

$$D(5) = 15 \cdot (5 - 3) \mod 26 = 15 \cdot 2 \mod 26 = 30 \mod 26 = 4$$

4 corresponds to E.

12. **P = 15**:

$$D(15) = 15 \cdot (15 - 3) \mod 26 = 15 \cdot 12 \mod 26 = 180 \mod 26 = 24$$

24 corresponds to Y.

13. **X = 23** (again):

$$D(23) = 15 \cdot (23 - 3) \mod 26 = 15 \cdot 20 \mod 26 = 300 \mod 26 = 14$$

14 corresponds to O.

14. **N = 13**:

$$D(13) = 15 \cdot (13 - 3) \mod 26 = 15 \cdot 10 \mod 26 = 150 \mod 26 = 20$$

20 corresponds to U.

Step 5: Convert numeric values back to letters

The decrypted message is:

`HELLOHOWAREYOU`

Final Decrypted Message

The final decrypted message is:
"HELLOHOWAREYOU"

Problem 3.3 (10 points)

Provide the pseudocode for the affine cipher's encryption and decryption algorithms.

Solution 3.3

Algorithm 1 AffineEncrypt(plaintext, a, b)

Input: plaintext, a, b
Output: encrypted_text

Function AffineEncrypt(plaintext, a, b):
  # Initialize an empty string to hold the encrypted message
  encrypted_text = ""

  # Iterate through each character in the plaintext
for each character **in** plaintext **do**
  if character is an alphabetic letter **then**
    # Convert the letter to its numeric equivalent (0-25)
    x = Numeric value of character ($A = 0, B = 1, \ldots, Z = 25$)

    # Apply the encryption formula
    $y = (a \cdot x + b) \mod 26$

    # Convert the numeric result back to a letter and add to encrypted_text
    encrypted_letter = Corresponding letter for y
    Append encrypted_letter to encrypted_text
  else
    # Non-alphabetic characters are added as they are
    Append character to encrypted_text
  end if
end for

  Return encrypted_text

Algorithm 2 AffineDecrypt(ciphertext, a, b)

Input: ciphertext, a, b
Output: decrypted_text

Function AffineDecrypt(ciphertext, a, b):
  # Find the modular multiplicative inverse of a modulo 26
  $a\_inverse$ = Modular multiplicative inverse of $a \mod 26$

  # Initialize an empty string to hold the decrypted message
  decrypted_text = ""

  # Iterate through each character in the ciphertext
for each character **in** ciphertext **do**
  if character is an alphabetic letter **then**
    # Convert the letter to its numeric equivalent (0-25)
    y = Numeric value of character ($A = 0, B = 1, \ldots, Z = 25$)

    # Apply the decryption formula
    $x = a\_inverse \cdot (y - b) \mod 26$

    # Convert the numeric result back to a letter and add to decrypted_text
    decrypted_letter = Corresponding letter for x
    Append decrypted_letter to decrypted_text
  else
    # Non-alphabetic characters are added as they are
    Append character to decrypted_text
  end if
end for

  Return decrypted_text

Problem 4: Cryptographic Hash Functions and Collisions (20 points)

Hash functions play a critical role in blockchain technology. Consider the following hash function:

$$H(x) = (x \mod p) + (x \mod q)$$

where p and q are large prime numbers.

Problem 4.1 (10 points)

Is this hash function collision-resistant? Justify your answer by providing an example where a collision could occur.

Solution 4.1

This function is not collision-resistant. Example:
For $x_1 = 12$ and $x_2 = 17$, we get the same hash value: $H(12) = H(17) = 7$.

Problem 4.2 (10 points)

Explain why collision resistance is important for secure blockchain transactions.

Solution 4.2

Collision resistance prevents double-spending attacks by ensuring that two different transactions cannot produce the same hash.

Problem 5: Smart Contracts – Practical Application

Consider the following incomplete pseudocode for a smart contract that handles rental payments between a tenant (Alice) and a landlord (Bob). The smart contract automatically transfers rental payments each month and can impose a penalty for late payments.

Problem 5.1 (4 points)

Complete the missing condition in the `payRent()` function that ensures the tenant can only pay rent once per month.

Solution 5.1:

```
require(isPaid == false, "Rent has already been paid this month
    .");
```

Problem 5.2 (4 points)

Fill in the missing condition in `payRent()` to check if the payment is late. Use the `dueDate` to determine if the tenant has missed the due date.

Solution 5.2:

```
if (block.timestamp > dueDate) {
    // Apply penalty
    rentAmount += penaltyAmount;
}
```

Problem 5.3 (4 points)

Fill in the missing condition in the `resetPaymentStatus()` function to ensure it resets the payment status only at the start of a new month.

Solution 5.3:

```
require(block.timestamp > dueDate, "Cannot reset payment status
    until the new month starts.");
```

Problem 5.4 (8 points)

Identify one logical flaw in the given pseudocode and explain how it could cause unintended behavior in the contract. (Hint: Think about the situation when a penalty is applied, and rent is paid late.)

Solution 5.4: One logical flaw is that when the tenant pays late, the penalty is added to the `rentAmount` directly. This causes an issue because the increased `rentAmount` (including the penalty) remains in effect for future months if it is not reset after payment. This means future rent payments could incorrectly include the penalty.
To fix this: After the payment is made, the `rentAmount` should be reset to its original value (without the penalty) so that subsequent payments are not affected by previous penalties.

Here's how we can fix it in the `payRent()` function:

```
// Transfer rent to landlord
landlord.transfer(rentAmount);
isPaid = true;

// Reset rent amount after payment, if penalty was applied
rentAmount -= penaltyAmount;
```

This ensures that the penalty is applied only for the late payment and does not persist into the next month.