CSIT 6000Q - Blockchain and Smart Contracts
Assignment 1

Due: 11:59 PM on Sunday, Oct. 13, 2024
Released: September 21, 2024

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.interactive-maths.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."
```

note : 5-column transposition cipher key: `zmuxv`

Problem 1.2 (7.5 points)

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

```
"GTGYIGSIVANTIGRMFNONAHESONIIONRLBNIVTWLLMETRELIVLNNELGISEYEAENDEATRLNETEFIURNEIELSAL"
```

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/

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 $(a = 7, b = 3)$.

Problem 3.2 (7.5 points)

Decrypt the ciphertext "AFCCXAXBDSFPXN" using the same key.

Problem 3.3 (10 points)

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

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.

Problem 4.2 (10 points)

Explain how a collision in this hash function could allow a malicious user to alter the blockchain ledger. Provide an example scenario.

Problem 5: Smart Contracts – Practical Application (20 points)

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.

```
// Smart Contract for Rental Payments
contract RentalAgreement {
    address tenant;   // Alice
    address landlord; // Bob
    uint rentAmount;
    uint dueDate;     // Monthly due date in Unix timestamp
    uint penaltyAmount;
    bool isPaid;
    constructor(address _tenant, address _landlord, uint _rentAmount,
                uint _dueDate, uint _penaltyAmount) {
        tenant = _tenant;
        landlord = _landlord;
        rentAmount = _rentAmount;
        dueDate = _dueDate;
        penaltyAmount = _penaltyAmount;
        isPaid = false;
    }

    // Function to make rent payment
    function payRent() public {
        require(msg.sender == tenant, "Only the tenant can pay the rent.");
        require( /* Fill in the missing condition to ensure rent is paid only once per month */ );

        if ( /* Fill in the condition to check if payment is late */ ) {
            // Apply penalty
            rentAmount += penaltyAmount;
        }

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

    // Reset payment status at the start of each month
    function resetPaymentStatus() public {
        require( /* Fill in the missing condition */ );
        isPaid = false;
    }

    // Function to check if rent is paid
    function checkPaymentStatus() public view returns (bool) {
        return isPaid;
    }
}
```

Problem 5.1 (4 points)

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

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.

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.)