CS 458 – Fall 2024
Introduction To Information Security
Assignment #1

1 Source Code

```python
# Caesar cipher

# Function to encrypt plaintext using the Caesar cipher
def encrypt(plaintext, key):
    ciphertext = ""
    for char in plaintext:
        if char.isalpha():
            shift = 65 if char.isupper() else 97
            ciphertext += chr((ord(char) - shift + key) % 26 + shift)
        else:
            ciphertext += char
    return ciphertext

# Function to decrypt ciphertext using the Caesar cipher
def decrypt(ciphertext, key):
    plaintext = ""
    for char in ciphertext:
        if char.isalpha():
            shift = 65 if char.isupper() else 97
            plaintext += chr((ord(char) - shift - key) % 26 + shift)
        else:
            plaintext += char
    return plaintext

# Function to perform brute force attack on the ciphertext
def brute_force_attack(ciphertext):
    print("Brute Force Attack Results:")
    for key in range(26):
        possible_plaintext = decrypt(ciphertext, key)
        print(f"Key {key}: {possible_plaintext}")

# Main menu and user interaction
def main():
```

```
    while True:
        print("\nChoose an option:")
        print("1. Encryption")
        print("2. Decryption")
        print("3. Brute Force Attack")
        print("4. Exit")
        choice = input("Enter your choice (1/2/3/4): ")

        if choice == '1':  # Encryption
            plaintext = input("Enter plaintext: ")
            try:
                key = int(input("Enter key (numeric): "))
                ciphertext = encrypt(plaintext, key)
                print(f"Ciphertext: {ciphertext}")
            except ValueError:
                print("Invalid key! Please enter a numeric value.")

        elif choice == '2':  # Decryption
            ciphertext = input("Enter ciphertext: ")
            try:
                key = int(input("Enter key (numeric): "))
                plaintext = decrypt(ciphertext, key)
                print(f"Decrypted Plaintext: {plaintext}")
            except ValueError:
                print("Invalid key! Please enter a numeric value.")

        elif choice == '3':  # Brute Force Attack
            ciphertext = input("Enter ciphertext: ")
            brute_force_attack(ciphertext)

        elif choice == '4':  # Exit
            print("Exiting the program.")
            break

        else:
            print("Invalid choice! Please enter 1, 2, 3, or 4.")

# Run the program
if __name__ == "__main__":
```

```
    main()
```

2 Detailed Explanation

```
encrypt(plaintext, key)
```

This function takes the plaintext (original message) and a numeric key as inputs and returns the ciphertext (encrypted message) using the Caesar cipher. The Caesar cipher works by shifting each letter in the plaintext by the key value.

How it works:

- The function loops through each character in the plaintext.
- If the character is an alphabetic character, it is shifted by the key value using the ord() function, which converts a character to its ASCII value. The chr() function converts the resulting shifted value back into a character.
- It handles both uppercase and lowercase letters:
    - For uppercase letters, the shift starts from the ASCII value of 'A' (65).
    - For lowercase letters, the shift starts from the ASCII value of 'a' (97).
- Non-alphabetic characters (e.g., spaces, punctuation) are added to the ciphertext without modification.
- The final encrypted string is returned as the ciphertext.

```
decrypt(ciphertext, key)
```

This function decrypts the ciphertext back to the original plaintext using the same key that was used for encryption.

How it works:

- The process is similar to the encrypt() function but in reverse. Instead of shifting the characters forward, it shifts them backward by the key value.
- Like the encrypt() function, it handles both uppercase and lowercase letters, and non-alphabetic characters remain unchanged.
- It loops through each character in the ciphertext, checks if it's a letter, shifts it back by the key, and builds the decrypted plaintext.
- The final decrypted string is returned as the plaintext.

```
brute_force_attack(ciphertext)
```

This function performs a brute force attack on the ciphertext, which means it tries every possible key (from 0 to 25) to decrypt the message. It's helpful when the user does not know the key used to encrypt the ciphertext.

How it works:

- The function loops through all possible keys (0 to 25) and attempts to decrypt the ciphertext with each key using the decrypt() function.
- For each key, the function prints the possible plaintext result.
- This allows the user to see all possible decrypted messages and choose the one that makes sense.

```
main()
```

This function handles user interaction, presents a menu of options (encryption, decryption, brute force attack, or exit), and calls the appropriate function based on the user's choice.

How it works:

- A while-loop ensures the program continues to run until the user wants to exit (option 4).
- The program prompts the user for input, such as plaintext, ciphertext, and key values, and performs basic error handling for invalid inputs (like non-numeric keys or invalid menu options).
- Based on the user's choice, it calls either:
    - encrypt() for encryption,
    - decrypt() for decryption,
    - brute_force_attack() for brute force decryption.
- If the user chooses to exit, the program breaks out of the loop and ends.

Additional Notes:

- Error Handling:
  The program uses a try-except block to catch invalid numeric inputs for the key. This ensures the program doesn't crash and instead prompts the user to enter valid input.

- Menu System:
  The menu lets the user easily navigate between the options. The loop will keep running until the user chooses to exit.

3 Screenshots of Demonstration