BRAC UNIVERSITY

Inspiring Excellence

Project for

CSE721:

Introduction to Cryptography

CryptoSuite:

Encryption/Decryption Tool + Hill Known-Plaintext Cracker

Prepared For :

Dr. Md Sadek Ferdous, Professor, CSE, BRAC University

Prepared By :

Abdullah Khondoker [20301065]

A.S.M Mahabub Siddiqui [20301040]

github.com/abdullah1065/CryptoSuite

1. Overview

In this project, we have implemented the encryption and decryption processes of four different classic cryptographic systems.

These are:

- *Caesar Cipher*
- *Affine Cipher*
- *Playfair Cipher*
- *Hill Cipher* (2×2)

We then implemented a cryptographic cracking tool that performs a known-plaintext attack on the *Hill Cypher* (2×2). Here, the plaintext and ciphertext of a *Hill Cipher* are given, and the crypto cracker cracked the key.

2. Architecture

The codebase is organized as follows:

- `cryptoSuite.py` implemented the encryption and decryption process of four different classic cryptographic systems and the crypto cracker for the *HillCipher*. All cipher implementations are class-based. There are four classes of cryptographic systems. Under those classes, we have implemented the encryption and decryption functions. Under the class of *HillCipher*, the crypto-cracking tool is implemented.
- In `app_gui.py`, we implemented a Tkinter GUI front-end. Here, we are assisted by LLM applications [*ChatGPT and Gemini*].
- In `app_cli.py`, we implemented the console interface for input/output.

3. Libraries Used

Only Python standard libraries are used:

- tkinter and re (key parsing) for Graphical User Interface.
- Built-in modular arithmetic "pow" is used.
- From the *future* library we imported *annotations* and from the *typing* library we imported *list, tuple* and *optional*.
- Built-in Functions *isalpha() , join() , ord() , chr() , upper() , lower(), remove()* are used.

4. Cipher Operations

Each cipher supports encryption and decryption via the GUI/CLI.

4.1 *Caesar Cipher*

Key: integer shift k

Encryption:

- Convert ‘a’..‘z’ to 0..25
- shifts letters forward, $(x + k)\ mod\ 26$
- Convert back to letter and output as uppercase

Decryption:

- Convert ‘a’..‘z’ to 0..25
- shifts backward by $(x - k)\ mod\ 26$
- Convert back to letter and output as lowercase

4.2 *Affine Cipher*

Key: (α, β)

Computes the modular inverse of a modulo 26

Encryption:

- Checks input is alphabet or not
- Convert ‘a’..‘z’ to 0..25
- Implements $E(x) = \alpha x + \beta\ (mod\ 26)$
- Convert back to letter and output as uppercase

Decryption:

- Checks input is alphabet or not
- Convert ‘a’..‘z’ to 0..25
- $D(x) = \alpha^{-1}(x - \beta)\ (mod\ 26)$. α must be invertible $mod\ 26$ $(gcd(\alpha, 26) = 1)$
- Convert back to letter and output as lowercase

4.3 *Playfair Cipher*

Key: keyword builds a (5×5) key matrix (I/J combined).

Create digrams: Splits plaintext into digrams, inserting ‘X’ between repeated letters or at the end if needed.

Find position: Finds and returns the (row, column) position of a character in the key matrix.

Remove inserted x: Removes padding ‘X’ characters that were inserted during encryption.

Encryption:

- Saves non-letter symbols with their indices
- Does letter only operation
- Creates diagrams
- Encrypts digrams using Find position
- Reinserts non-letter symbols and gives output

Decryption:

- Saves non-letter symbols with their indices
- Does letter only operation
- Creates diagrams
- Decrypts digrams using Find position
- Reinserts non-letter symbols and gives output

4.4 *Hill Cipher*

Key: a 2×2 matrix K with entries taken $(mod\ 26)$.

$$K = \begin{bmatrix} a & b \\ c & d \end{bmatrix} (mod\ 26) \quad Letters\ are\ mapped\ as{:}\ A = 0, B = 1, ..., Z = 25$$

Encryption:

- Convert plaintext letters to numbers 0-25
- Take plaintext in blocks of **2 letters**: $P = \begin{bmatrix} p_1 & p_2 \end{bmatrix}$
- Multiply by the key matrix and take mod 26: $C = E(K, P) = PK\ (mod\ 26)$
- Convert the resulting numbers back to letters (ciphertext). If the plaintext length is odd, pad with X to make pairs.

Decryption:

- To decrypt, the key matrix must be invertible mod 26, i.e.,$(gcd(det(K), 26) = 1)$
- Compute the modular inverse of the key:

$$K^{-1} = (det(K))^{-1}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}(mod\ 26)$$

where $det(K) = ad - bc\ (mod\ 26)$

- Take ciphertext in 2-letter blocks: $C = \begin{bmatrix} c_1 & c_2 \end{bmatrix}$
- Recover plaintext: $P = D(K, C) = CK^{-1}\ (mod\ 26) = PKK^{-1} = P$
- Convert numbers back to letters

5. *Hill Cipher* Known-Plaintext Attack (2 x 2)

Input: take a piece of known plaintext and its matching ciphertext (both produced using the same unknown Hill 2×2 key)

Preprocess: remove all non-letter characters, convert remaining letters to uppercase, and map letters to numbers $A = 0$ to $Z = 25$

Need minimum data: ensure there are at least 4 letters in both the plaintext and the ciphertext (because a 2×2 matrix requires four values)

Form matrices from blocks: take two consecutive plaintext pairs (4 letters) to build a 2×2 plaintext matrix P, and take the corresponding two ciphertext pairs to construct a 2×2 ciphertext matrix C

Check invertibility: compute whether P is invertible mod 26 (its determinant has a modular inverse). If P is not invertible, skip this block

Recover the key: when an invertible P is found, compute the key matrix:

$$K = P^{-1}C(mod\ 26)$$

Sliding search: if the first block fails, the program slides forward by one block (2 letters) and repeats steps 4-6 until a valid key is found

Output: return the recovered 2×2 key matrix K; if no invertible plaintext block exists in the provided sample, return failure / no key found.

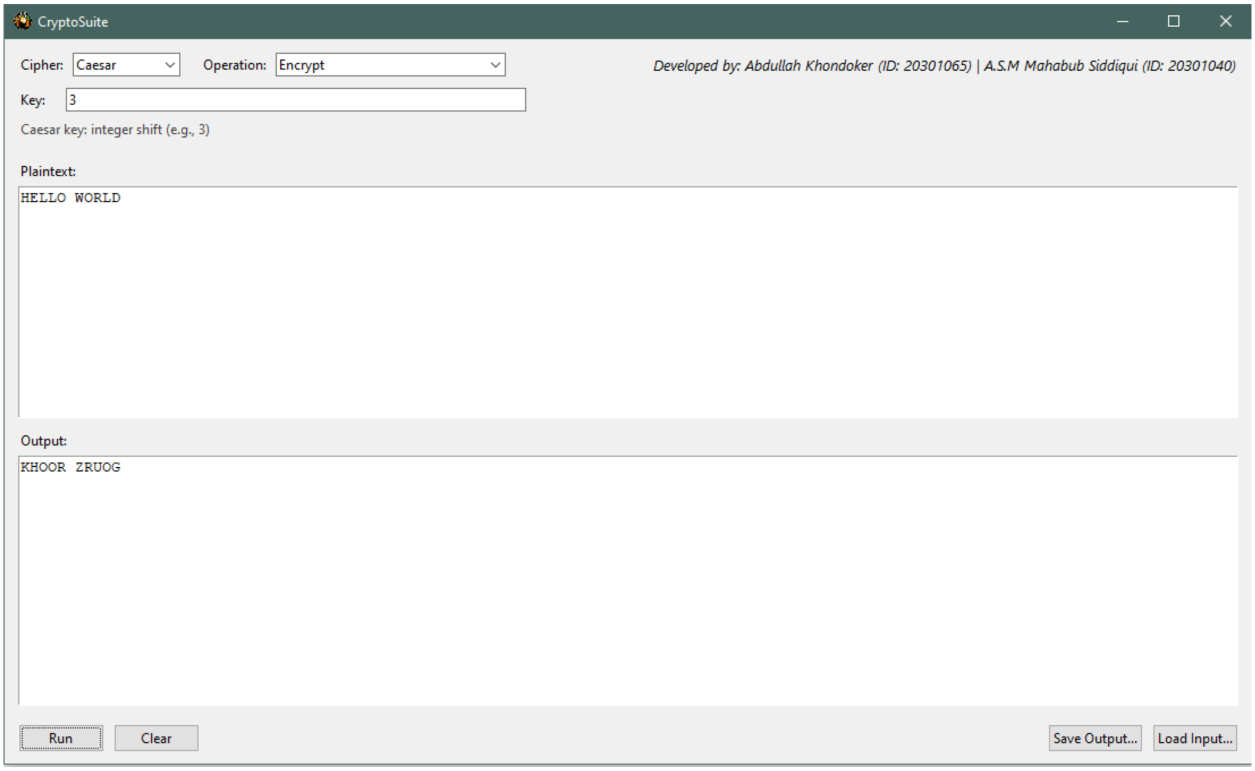
6. How to Run

1. To run the code block Visual Studio Code and Python needs to be installed. [How to Run Python in Visual Studio Code on Windows]
2. Then download the zipped codeblock from GitHub repository.
3. Unzip the folder.
4. Open the folder with Visual Studio Code.
5. Open TERMINAL :
   - For GUI: In TERMINAL, write *python app_gui.py* [it will open Graphical User Interface, use the input-output shown in section 7].
   - For CLI: In TERMINAL, write *python app_cli.py* [It will open the command based Interface, use the only if GUI doesn't work.].
   - In TERMINAL, write *python main.py* for the original demo runner (file-based).
6. After proving required inputs (shown in section 7) click Run button at the bottom left of the GUI.

7. Screenshots

Caesar Cipher:

- **Encryption:**
  1. Provide a key as shown in the interface.
  2. Enter Plaintext
  3. Click Run



Cipher: Caesar Operation: Encrypt

Developed by: Abdullah Khondoker (ID: 20301065) | A.S.M Mahabub Siddiqui (ID: 20301040)

Key: 3

Caesar key: integer shift (e.g., 3)

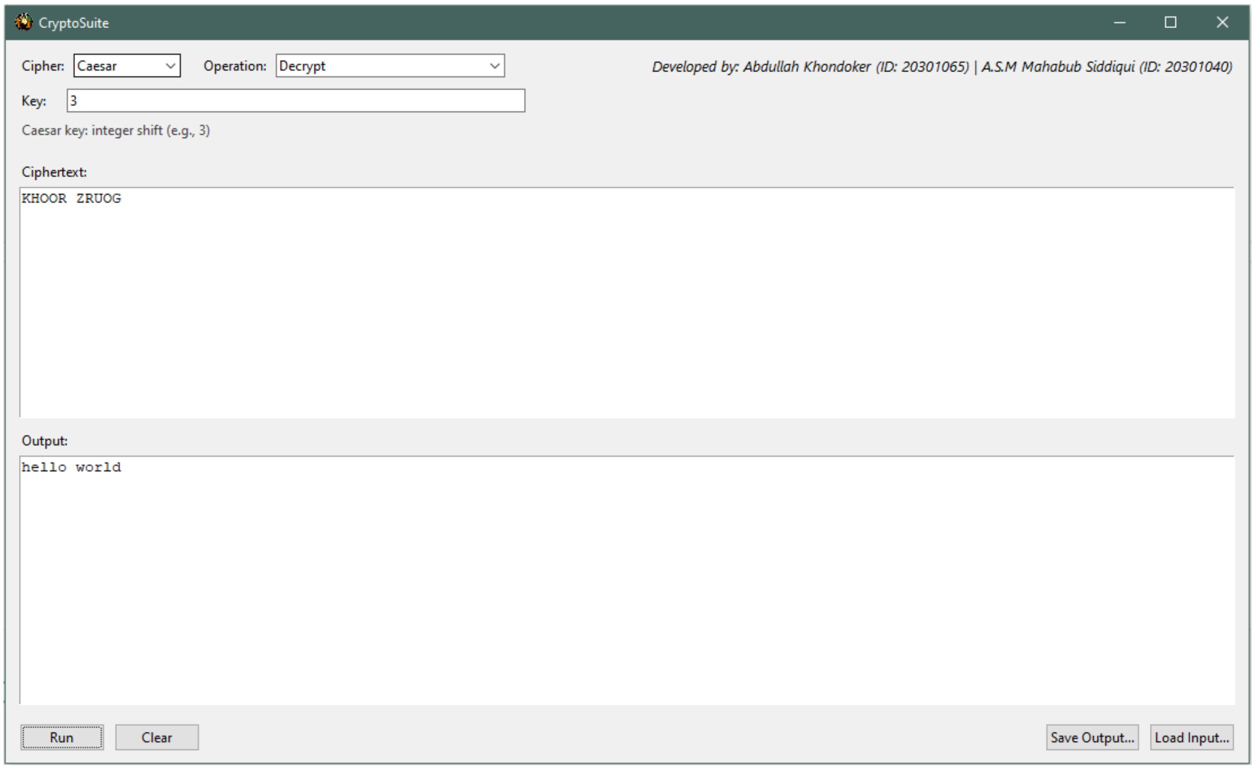
Plaintext:

```
HELLO WORLD
```

Output:

```
KHOOR ZRUOG
```

- **Decryption:**
    1. Provide a key as shown in the interface.
    2. Enter Ciphertext
    3. Click Run



Cipher: Caesar Operation: Decrypt

Developed by: Abdullah Khondoker (ID: 20301065) | A.S.M Mahabub Siddiqui (ID: 20301040)

Key: 3

Caesar key: integer shift (e.g., 3)

Ciphertext:

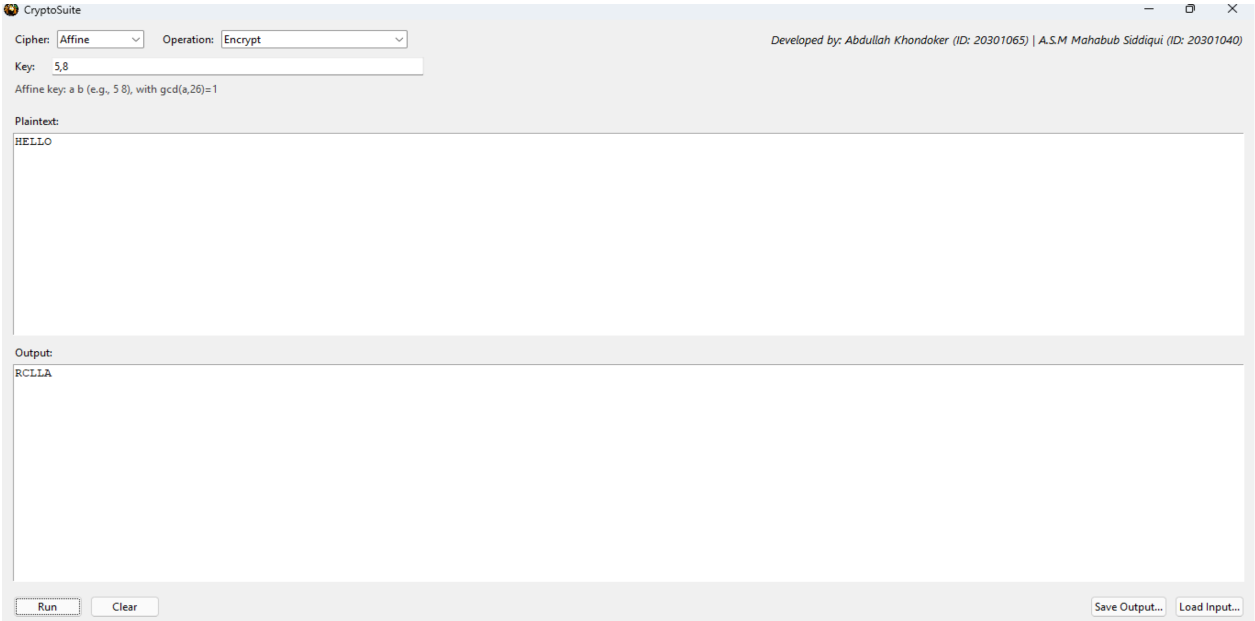
```
KHOOR ZRUOG
```

Output:

```
hello world
```

Affine Cipher:

- **Encryption:**
    1. Provide a key as shown in the interface.
    2. Enter Plaintext
    3. Click Run



Cipher: Affine Operation: Encrypt

Developed by: Abdullah Khondoker (ID: 20301065) | A.S.M Mahabub Siddiqui (ID: 20301040)

Key: 5,8

Affine key: a b (e.g., 5 8), with gcd(a,26)=1

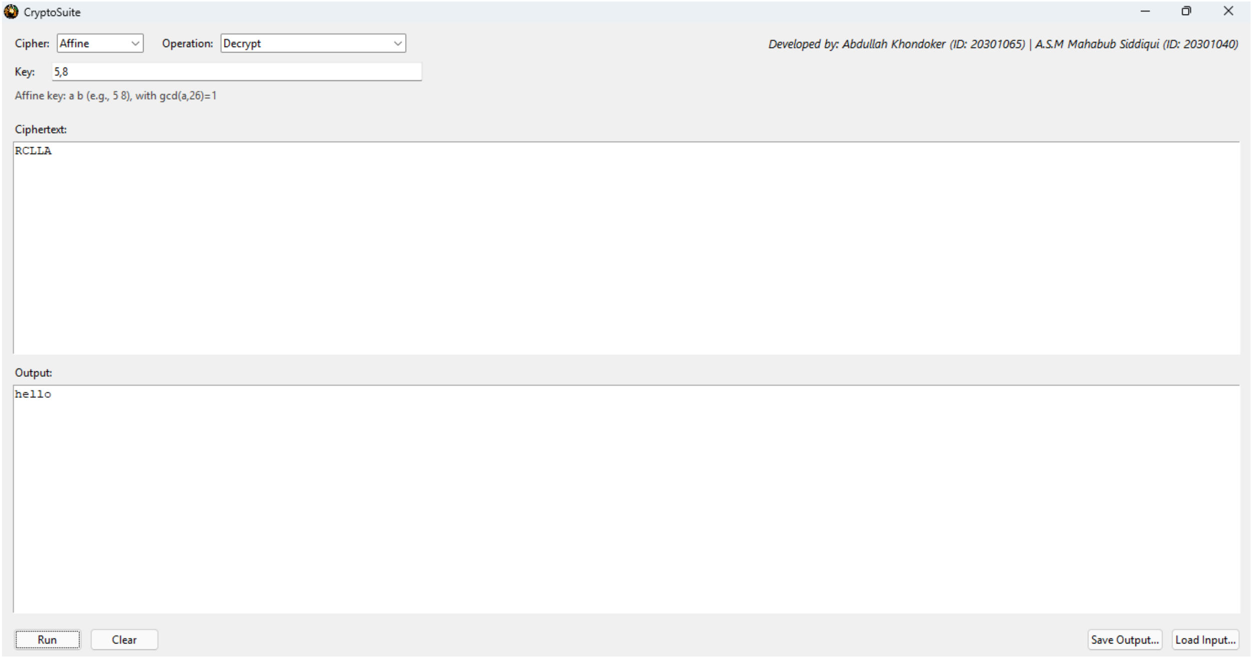
Plaintext:

```
HELLO
```

Output:

```
RCLLA
```

- **Decryption:**
    1. Provide a key as shown in the interface.
    2. Enter Ciphertext
    3. Click Run



Cipher: Affine Operation: Decrypt

Developed by: Abdullah Khondoker (ID: 20301065) | A.S.M Mahabub Siddiqui (ID: 20301040)

Key: 5,8

Affine key: a b (e.g., 5 8), with gcd(a,26)=1

Ciphertext:

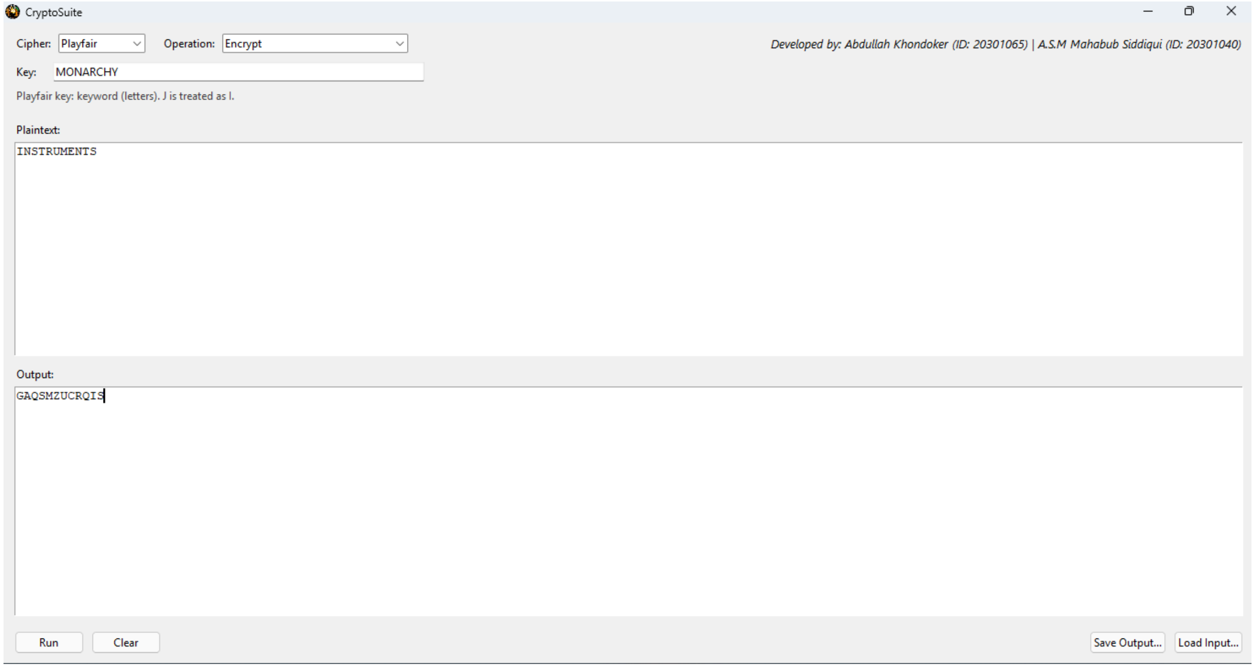
```
RCLLA
```

Output:

```
hello
```

Playfair Cipher:

- **Encryption:**
    1. Provide a key as shown in the interface.
    2. Enter Plaintext
    3. Click Run



Cipher: Playfair Operation: Encrypt

Developed by: Abdullah Khondoker (ID: 20301065) | A.S.M Mahabub Siddiqui (ID: 20301040)

Key: MONARCHY

Playfair key: keyword (letters). J is treated as I.

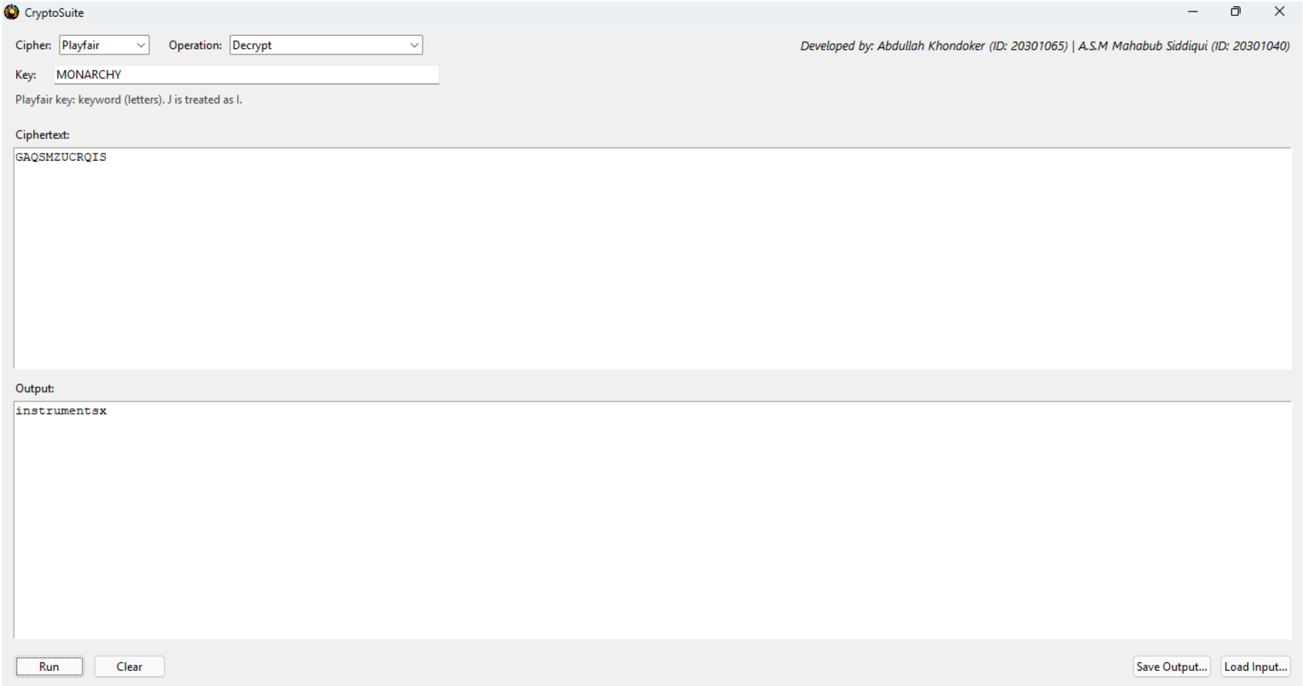
Plaintext:

```
INSTRUMENTS
```

Output:

```
GAQSMZUCRQIS
```

- **Decryption:**
    1. Provide a key as shown in the interface.
    2. Enter Ciphertext
    3. Click Run



Cipher: Playfair Operation: Decrypt

Developed by: Abdullah Khondoker (ID: 20301065) | A.S.M Mahabub Siddiqui (ID: 20301040)

Key: MONARCHY

Playfair key: keyword (letters). J is treated as I.

Ciphertext:

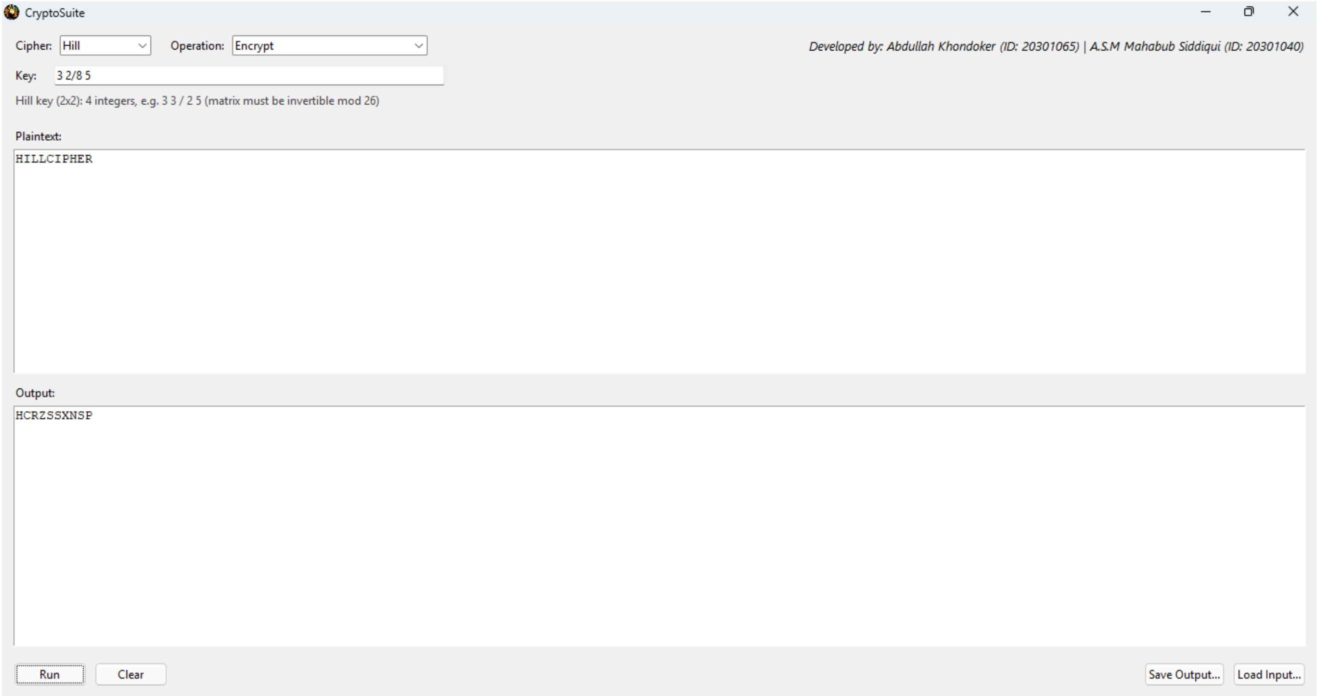
```
GAQSMZUCRQIS
```

Output:

```
instrumentsx
```

Hill Cipher:

- **Encryption:**
    1. Provide a key matrix as shown in the interface.
    2. Enter Plaintext
    3. Click Run



Cipher: Hill Operation: Encrypt

Developed by: Abdullah Khondoker (ID: 20301065) | A.S.M Mahabub Siddiqui (ID: 20301040)

Key: 3 2/8 5

Hill key (2x2): 4 integers, e.g. 3 3 / 2 5 (matrix must be invertible mod 26)

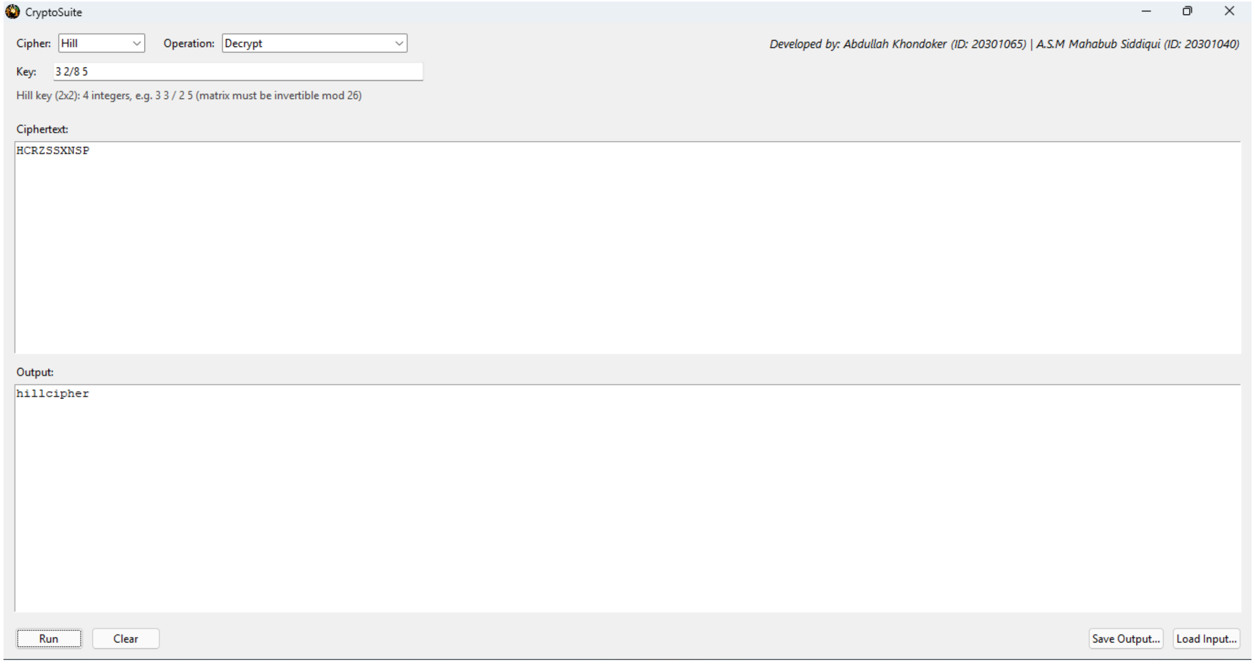
Plaintext:

```
HILLCIPHER
```

Output:

```
HCRZSSXNSP
```

- **Decryption:**
    1. Provide a key matrix as shown in the interface .
    2. Enter Ciphertext
    3. Click Run



Cipher: Hill Operation: Decrypt

Developed by: Abdullah Khondoker (ID: 20301065) | A.S.M Mahabub Siddiqui (ID: 20301040)

Key: 3 2/8 5

Hill key (2x2): 4 integers, e.g. 3 3 / 2 5 (matrix must be invertible mod 26)

Ciphertext:

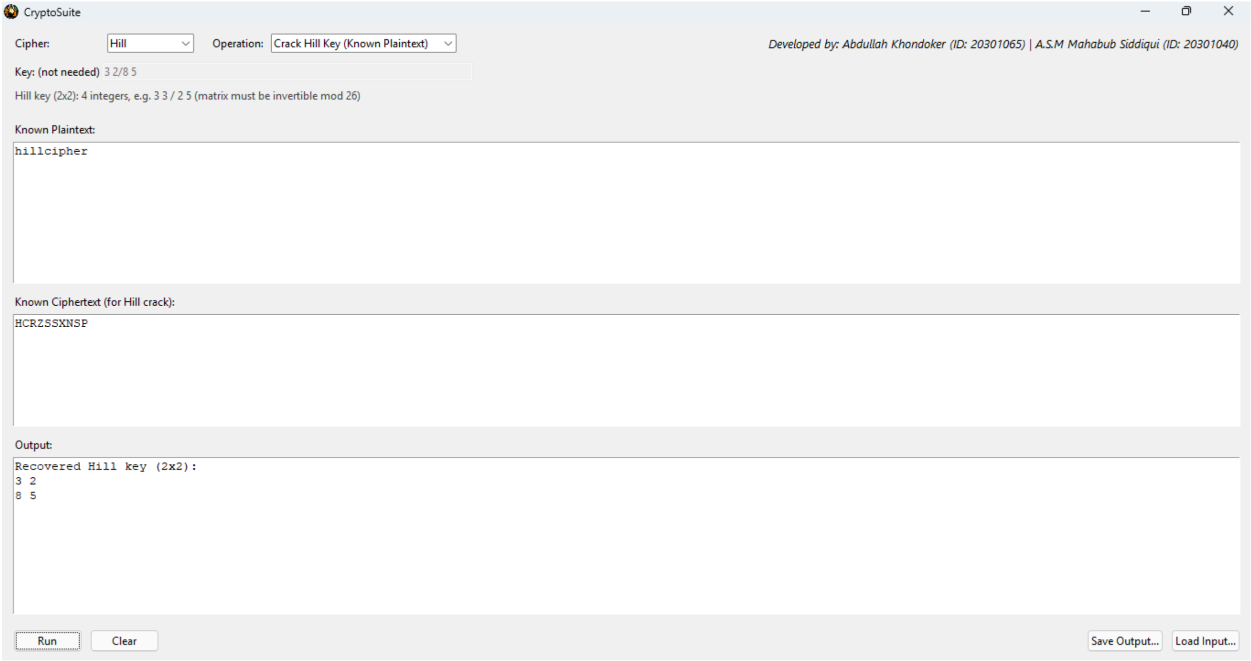
```
HCRZSSXNSP
```

Output:

```
hillcipher
```

Hill Cipher crack:

1. Enter the Plaintext
2. Enter the Ciphertext
3. Click Run



Cipher: Hill Operation: Crack Hill Key (Known Plaintext)

Developed by: Abdullah Khondoker (ID: 20301065) | A.S.M Mahabub Siddiqui (ID: 20301040)

Key: (not needed) 3 2/8 5

Hill key (2x2): 4 integers, e.g. 3 3 / 2 5 (matrix must be invertible mod 26)

Known Plaintext:

```
hillcipher
```

Known Ciphertext (for Hill crack):

```
HCRZSSXNSP
```

Output:

```
Recovered Hill key (2x2):
3 2
8 5
```

8. GitHub Link + Credits

GitHub repository link

Credits: ChatGPT and Gemini were used for refactoring and packaging assistance; all final code was reviewed and tested by us.