# INTE2401/2402 Lab 2

Student ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In this week’s lab, we attempt to use JavaScript to implement the affine cipher. We can use Notepad(or other text editor) to write a JavaScript file with the extension name **.html** and then open it with Internet Explore(or other web browser).

## Implementation of the Affine Cipher

In the lecture slides, we introduce the affine cipher and give the algorithm as

P=C=Z26

K={(k1,k2)| k1,k2 ∈ Z26, gcd(k1,26)=1}

E: c=E(p)=k1p+k2 (mod 26)

D: p=D(c)=k1-1(c-k2) (mod 26)

D(c)=k1-1(c-k2)=k1-1(k1p+k2-k2)=p

Note 1: gcd means greatest common divisor

Note 2: gcd(k1,26)=1 🡨🡪 k1 ∈{1,3,5,7,9,11,15,17,19,21,23,25}

Note 3: k1k1-1 =1 (mod 26), for example, 3\*9=27=1 (mod 26)

(1,1), (3,9), (5,21), (7,15), (11,19), (17,23), (25,25)

Note 4: k1=1, k2=0 should be avoided because p=c in this case.

The algorithm has 6 components, plaintext P, ciphertext C, encryption keys k1, k2, encryption method E, and decryption method C. It also defines restrictions on k1 as it has the inverse number k1-1, and it coprimes of 26.

Q1. Implement the encryption and decryption algorithms of Affine Cipher with JavaScript.

Hints:

When implementing, you may

1) check whether the input k1 satisfies the conditions that:

k1<26 && isNaN(k1) == false && k1%2 != 0 && k1 != 13 .

2) find the inverse of k1.

3) convert the input plaintext/ciphertext to number according to the alphabetic order, e.g. ‘a’/’A’ 🡪 0.

Remember in the Q3, Lab1, we use *str.charCodeAt(i)* to convert the ith character in string str to the ASCII code, e.g. ‘A’🡪65? In this task, you may first unify all of the letters to lower case or upper case. Then, convert them to ASCII code, and minus the offset to get the alphabetic orders.

For example, the input plaintext is “Ac”. You may use the method *plaintext.toLowerCase()* mentioned in Q5 to convert it to “ac”. Afterwards, you may use *order\_i = plaintext.charCodeAt(i) – 97* to convert it to (0, 2). Note that the ASCII code in decimal format of ‘a’ is 97.

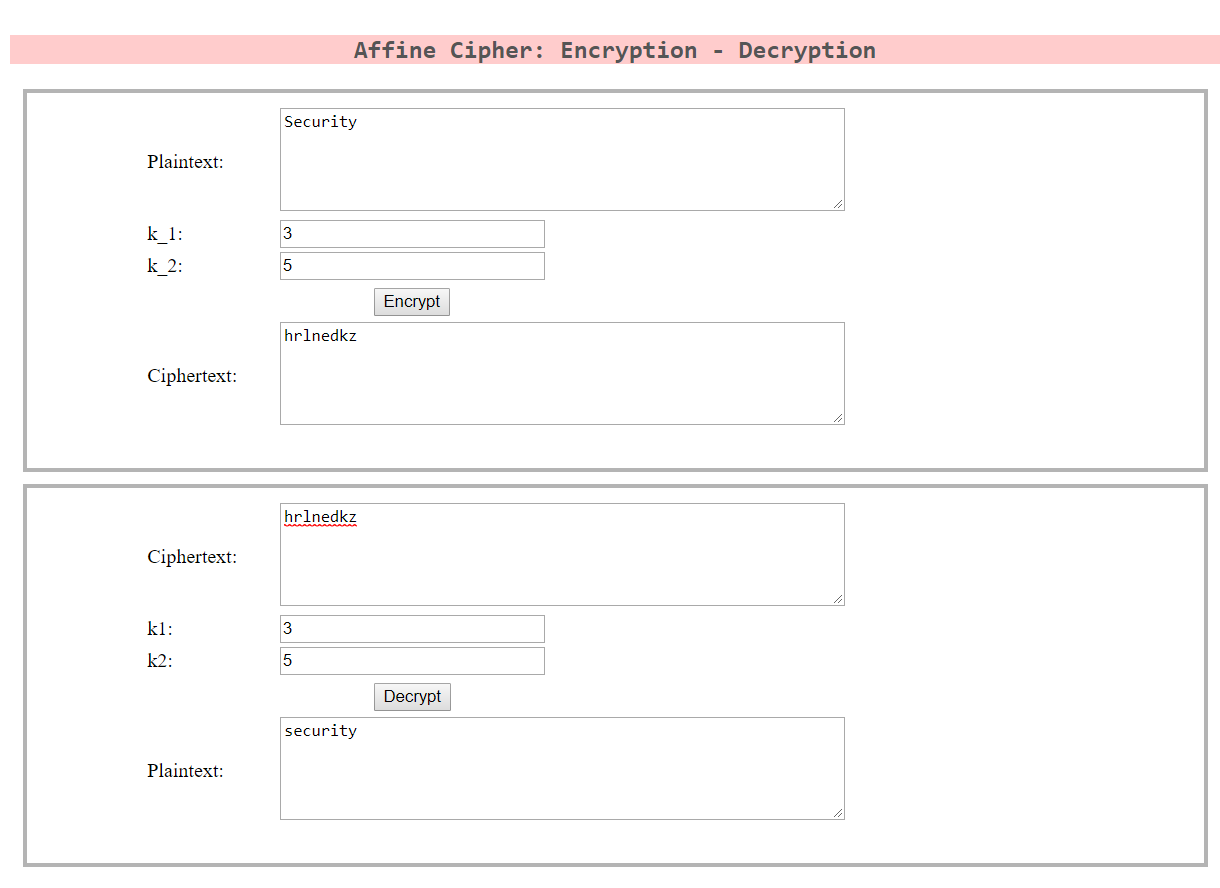
4) encrypt the *order\_i* to get *encrypted\_order* according to the encryption function.

5) convert it back to string. You may use the method

*String.fromCharCode(encrypted\_order + 97)* .

More about this method can be found in this link (<https://www.w3schools.com/jsref/jsref_fromcharcode.asp>).

Sample form:



Here is the table of ASCII code

