



Royal University of Bhutan



རྒྱུག་རྒྱལ་འཛིན་གཞུག་ལག་སློབ་མེ།

College of Science and Technology  
Rinchending: Bhutan



---

# SWS101

## Introduction to Cybersecurity (SS2025)

### *Cryptographic Cyphers Assignment 4*

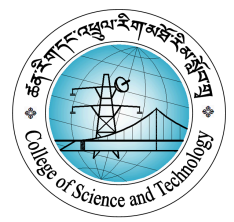
Submitted By;

Student Name: Thinley Dorji

Enrollment No.: 02230308

Programme: BESWE

Date: 17/05/2025



འབྲུག་རྒྱལ་ཁའི་གཞུག་ལག་སློབ་མེ།  
College of Science and Technology  
Rinchending: Bhutan

---

## RUB Wheel of Academic Law: Academic Dishonesty

Section H2 of the Royal University of Bhutan's *Wheel of Academic Law* provides the following definition of academic dishonesty:

Academic dishonesty may be defined as any attempt by a student to gain an unfair advantage in any assessment. It may be demonstrated by one of the following:

1. **Collusion:** the representation of a piece of unauthorized group work as the work of a single candidate.
2. **Commissioning:** submitting an assignment done by another person as the student's own work.
3. **Duplication:** the inclusion in coursework of material identical or substantially similar to material which has already been submitted for any other assessment within the University.
4. **False declaration:** making a false declaration in order to receive special consideration by an Examination Board or to obtain extensions to deadlines or exemption from work.
5. **Falsification of data:** presentation of data in laboratory reports, projects, etc., based on work purported to have been carried out by the student, which has been invented, altered or copied by the student.
6. **Plagiarism:** the unacknowledged use of another's work as if it were one's own.

Examples are:

- verbatim copying of another's work without acknowledgement.
- paraphrasing of another's work by simply changing a few words or altering the order of presentation, without acknowledgement.
- ideas or intellectual data in any form presented as one's own without acknowledging the source(s).
- making significant use of unattributed digital images such as graphs, tables, photographs, etc. taken from test books, articles, films, plays, handouts, internet, or any other source, whether published or unpublished.
- submission of a piece of work which has previously been assessed for a different award or module or at a different institution as if it were new work.
- use of any material without prior permission of copyright from appropriate authority or owner of the materials used".

Font: Lato

Font Size: 12

འབྲུག་རྒྱལ་ཁོངས་གཞི་རིག་ལག་སློབ་མཁུ་  
College of Science and Technology  
Rinchending: Bhutan

---

## AUTOKEY CIPHER

### Introduction

#### Historical Context

The Auto Key Cipher is a type of code that improves on the Vigenère Cipher. While the Vigenère uses a repeating keyword, the Auto Key Cipher uses the original message to make the key longer, which makes it harder to crack. Developed in the 16th century, this method was seen as stronger against certain code-breaking techniques because the key doesn't repeat.

#### Significance in Cryptographic Evolution

The Auto Key Cipher was a big improvement in code-making. Unlike the Vigenère Cipher, which reused its key, this cipher avoided repetition, making it tougher to crack with common attacks. While better ciphers came later, the ideas behind the Auto Key Cipher helped shape modern encryption methods.

### Cipher Fundamentals

#### Mechanism and Functioning

The Auto Key Cipher works as follows:

1. Key Setup: A short initial key is chosen (e.g., "KEY").
2. Encryption:
  - The initial key is used to encrypt the first few letters.
  - The remaining plaintext is appended to the key to continue encryption.
  - Each plaintext character is shifted according to the corresponding key character (using modular arithmetic).

འབྲུག་རྒྱལ་ཁོན་གཙུག་ལག་སློབ་མཁེ།  
College of Science and Technology  
Rinchending: Bhutan

---

## Formula:

$$C_i = (P_i + K_i) \bmod 26$$

$$P_i = (C_i - K_i) \bmod 26$$

where:

- $C_i$  = Ciphertext character
- $P_i$  = Plaintext character
- $K_i$  = Key character

## Glossary

- Polyalphabetic Cipher: A cipher that uses multiple substitution alphabets.
- Modular Arithmetic: Arithmetic operations performed within a fixed range (e.g., mod 26 for letters).
- Kasiski Examination: A cryptanalysis method to detect repeated sequences in ciphertext.

## Diagram

Below is a diagram illustrating the encryption process:

Plaintext: H E L L O

Key: K E Y H E

-----

Shift: (H+K) (E+E) (L+Y) (L+H) (O+E)

Ciphertext: R I J A S

འབྲུག་རྒྱལ་ཁོལ་གཞུག་ལག་སློབ་ཁྲུང་།  
College of Science and Technology  
Rinchending: Bhutan

---

## Security Analysis

### Cryptanalysis Techniques

The Auto Key Cipher is vulnerable to:

- Known Plaintext Attacks: If an attacker knows part of the plaintext, they can deduce the key.
- Frequency Analysis (with limitations): Since the key is not fully repeating, frequency analysis is less effective but still possible with long messages.

## Strengths and Weaknesses

Strengths:

- No repeating key pattern
- More secure than Vigenère for short messages

Weaknesses:

- Vulnerable if initial key is short
- Known plaintext reveals key

## Practical Demonstration of Breaking the Cipher

Given Ciphertext: "RIJAS" (assuming initial key length = 3)

1. Guess possible keys (e.g., "KEY").
2. Decrypt using the key:
  - a.  $R - K = H$
  - b.  $I - E = E$
  - c.  $J - Y = L$

འབྲུག་རྒྱལ་ཁོངས་གཞི་རིག་སློབ་ཐང་།  
College of Science and Technology  
Rinchending: Bhutan

---

- d.  $A - H = L$
  - e.  $S - E = O$
3. Recovered Plaintext: "HELLO"

## Practical Implementation

```
def autokey_encrypt(plaintext, key):  
    ciphertext = []  
    key = key.upper()  
    plaintext = plaintext.upper()  
    key_stream = list(key) + [c for c in plaintext[-len(key):]]
```

```
    for p, k in zip(plaintext, key_stream):  
        if p.isalpha():  
            shift = (ord(p) + ord(k) - 2 * ord('A')) % 26  
            ciphertext.append(chr(shift + ord('A')))  
        else:  
            ciphertext.append(p)  
    return ''.join(ciphertext)
```

```
def autokey_decrypt(ciphertext, key):  
    plaintext = []  
    key = key.upper()  
    ciphertext = ciphertext.upper()  
    key_stream = list(key)
```

```
    for i, c in enumerate(ciphertext):  
        if c.isalpha():  
            p = (ord(c) - ord(key_stream[i]) + 26) % 26  
            plaintext.append(chr(p + ord('A')))  
            key_stream.append(plaintext[-1])  
        else:  
            plaintext.append(c)
```

འབྲུག་རྒྱལ་ཁོན་གཞུག་ལག་སློབ་མེ།  
College of Science and Technology  
Rinchending: Bhutan

```
return ''.join(plaintext)
```

# Example Usage

```
plaintext = "HELLO"
```

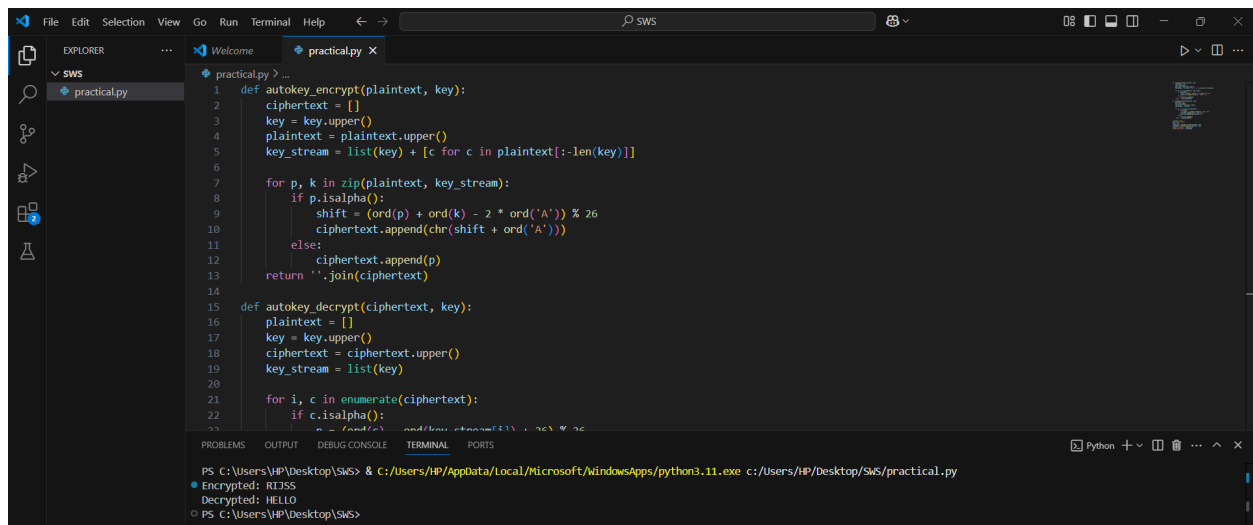
```
key = "KEY"
```

```
ciphertext = autokey_encrypt(plaintext, key)
```

```
decrypted = autokey_decrypt(ciphertext, key)
```

```
print("Encrypted:", ciphertext)
```

```
print("Decrypted:", decrypted)
```

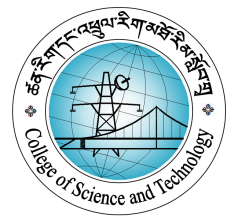


```
1 def autokey_encrypt(plaintext, key):
2     ciphertext = []
3     key = key.upper()
4     plaintext = plaintext.upper()
5     key_stream = list(key) + [c for c in plaintext[:-len(key)]]
6
7     for p, k in zip(plaintext, key_stream):
8         if p.isalpha():
9             shift = (ord(p) + ord(k) - 2 * ord('A')) % 26
10            ciphertext.append(chr(shift + ord('A')))
11        else:
12            ciphertext.append(p)
13    return ''.join(ciphertext)
14
15 def autokey_decrypt(ciphertext, key):
16     plaintext = []
17     key = key.upper()
18     ciphertext = ciphertext.upper()
19     key_stream = list(key)
20
21     for i, c in enumerate(ciphertext):
22         if c.isalpha():
23             n = (ord(c) - ord(key_stream[i]) + 26) % 26
24             plaintext.append(chr(n + ord('A')))
25             key_stream.append(ciphertext[i])
26         else:
27             plaintext.append(c)
28
29     return ''.join(plaintext)
```

PS C:\Users\HP\Desktop\SWS> & C:/Users/HP/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/HP/Desktop/SWS/practical.py  
Encrypted: RIJSS  
Decrypted: HELLO  
PS C:\Users\HP\Desktop\SWS>

## Implementation Challenges

- Key Management: Ensuring the key does not become predictable.
- Error Propagation: A single decryption error affects subsequent characters.
- Non-Alphabet Handling: Handling spaces and punctuation requires additional logic.



འབྲུག་རྒྱལ་ཁའི་གཞུག་ལག་སློབ་མེ།  
College of Science and Technology  
Rinchending: Bhutan

---

## Conclusion

The Auto Key Cipher was an important step forward in the history of cryptography. Unlike the Vigenère Cipher, which used a repeating key, the Auto Key Cipher made encryption stronger by using the plaintext itself to extend the key. This made it harder for attackers to break the code using simple methods like frequency analysis or the Kasiski examination.

However, despite its advantages, the Auto Key Cipher is not secure enough for modern use. If an attacker knows even a small part of the original message, they can figure out the rest of the key and decrypt the entire ciphertext. Also, since it still relies on basic letter shifts, it doesn't provide the level of security needed today, where computers can crack such codes easily.

Studying the Auto Key Cipher is still valuable because it helps us understand how encryption techniques evolved. It shows how early cryptographers tried to improve security by making keys less predictable. Even though we now use much stronger methods like AES and RSA, learning about ciphers like this one helps us appreciate the foundations of modern cryptography.

In short, the Auto Key Cipher was a clever idea for its time, but today, it serves more as a lesson in the history of code-making rather than a practical encryption tool.