**Information Security IA-1**

**CRYPTII**

Gauri Charpe 16010122027

Ashish Ghatkar 16010122057

Soumil Dhywershetty 16010122048

**Introduction**

Cryptii.com is a versatile and user-friendly online cryptography tool that enables users to encode, decode, and analyse a wide variety of ciphers. The platform is particularly valuable for students, cryptography enthusiasts, and professionals who seek to understand encryption techniques and their applications. By providing a dynamic and interactive environment, Cryptii.com helps users explore both historical and modern cryptographic methods.

With the increasing need for secure communication in the digital age, understanding encryption mechanisms has become crucial. Cryptii.com bridges the gap between theoretical knowledge and practical application by offering an accessible and visually engaging way to experiment with encryption and decryption processes. From classical ciphers used in historical contexts to advanced modern encryption algorithms, Cryptii.com serves as a comprehensive educational tool that fosters a deeper understanding of cryptography.

**Features/Characteristics**

Cryptii.com stands out due to its broad range of functionalities, designed to cater to various learning and analytical needs. The platform offers a number of significant features:

1. **Web-Based Platform**: Cryptii.com is an entirely web-based application, meaning users do not need to install any additional software. It is accessible from any device with an internet connection, making it highly convenient.
2. **Support for Various Ciphers**: The tool includes a comprehensive library of encryption techniques, ranging from classical substitution ciphers to advanced cryptographic functions.
3. **Interactive User Interface**: The website provides a user-friendly interface that visually demonstrates how plaintext is transformed into ciphertext and vice versa.
4. **Customizable Parameters**: Many ciphers available on Cryptii.com allow users to modify parameters such as key length, shift values, and encryption settings, providing a deeper understanding of how encryption works.
5. **Educational Resource**: Cryptii.com serves as a hands-on learning platform for students and professionals who want to explore encryption and decryption methodologies in an intuitive manner.
6. **Open-Source Nature**: The platform is built using open-source technologies, which allows for community contributions and continuous improvements.
7. **Real-Time Encoding and Decoding**: Users can instantly see the effect of encryption and decryption processes as they enter text and select different ciphers.
8. **Supports Multiple Cryptographic Disciplines**: The platform is not limited to simple encryption techniques but also includes hashing functions, message authentication codes, and stream ciphers.

**Methodology**

Cryptii.com employs various cryptographic methodologies, categorized into classical ciphers and modern encryption techniques. Below is a detailed breakdown:

**Classical Ciphers:**

Classical ciphers represent early encryption methods that, while no longer used for secure communications, remain fundamental in cryptographic education.

**1. Enigma Machine**

The **Enigma Machine** was a mechanical cipher device used by Nazi Germany during World War II. It encrypted messages using a complex system of rotating rotors, plugboards, and reflectors.

**History:**

* Invented by Arthur Scherbius in 1918.
* Used extensively by the German military.
* Broken by Alan Turing and his team at Bletchley Park, leading to major Allied victories.

**Working:**

1. A letter is pressed on the keyboard.
2. The signal passes through a plugboard, where some letters are swapped.
3. The signal moves through multiple rotors, each performing a different substitution.
4. A reflector bounces the signal back through the rotors in reverse.
5. The final encrypted letter lights up on a display.
6. After each key press, the rotors rotate, changing the encryption pattern dynamically.

**Decryption:**

* The receiver needed the exact rotor settings to decrypt the message.
* Without the correct settings, breaking the encryption was nearly impossible.

**Example:**

Assume an Enigma machine with simple rotor settings:  
Plaintext: **HELLO**  
Ciphertext: **MFNCZ**  
(Exact encryption varied due to rotor positions.)

**2. Caesar Cipher**

The **Caesar Cipher** is a simple substitution cipher where each letter is shifted by a fixed number.

**History:**

* Named after Julius Caesar, who used it to send secure messages.
* One of the earliest known ciphers.

**Working:**

1. Choose a shift value (e.g., shift by 3).
2. Replace each letter with another letter that is the chosen number of places forward in the alphabet.
3. Wrap around from Z to A if necessary.

**Decryption:**

* Simply shift the letters back by the same amount.

**Example:**

Shift = 3  
Plaintext: **HELLO**  
Ciphertext: **KHOOR**  
Decryption: Shift **back** by 3 → **HELLO**

**Weakness:**

* Easy to break using frequency analysis.

**3. Affine Cipher**

The **Affine Cipher** is a type of monoalphabetic substitution cipher that applies a mathematical function to encrypt letters.

**History:**

* Used in cryptography due to its mathematical approach.
* Provides stronger security than a simple Caesar cipher.

**Working:**

1. Convert each letter to a number (A=0, B=1, ..., Z=25).
2. Use the encryption formula: C=(aP+b)mod  26C = (aP + b) \mod 26C=(aP+b)mod26 where **P** is the plaintext letter’s numerical value, **a** and **b** are keys, and **C** is the ciphertext numerical value.
3. Convert numbers back to letters.

**Decryption:**

* Uses the modular inverse of **a** to reverse the transformation.

**Example:**

Let **a = 5, b = 8**  
Plaintext: **HELLO**  
H(7) → (5×7 + 8) mod 26 = 17 → **R**  
E(4) → (5×4 + 8) mod 26 = 2 → **C**  
L(11) → (5×11 + 8) mod 26 = 7 → **H**  
Ciphertext: **RCHH**

**Weakness:**

* If **a** is not chosen properly, decryption is difficult.
* Susceptible to brute-force attacks.

**4. Vigenère Cipher**

The **Vigenère Cipher** is a polyalphabetic substitution cipher that uses a repeating keyword to determine shifts.

**History:**

* Invented by Giovan Battista Bellaso in 1553.
* Often misattributed to Blaise de Vigenère.
* Was considered unbreakable for centuries until Charles Babbage and Friedrich Kasiski cracked it.

**Working:**

1. Choose a keyword (e.g., **KEY**).
2. Repeat the keyword until it matches the plaintext length.
3. Shift each plaintext letter by the corresponding keyword letter.
4. Use the formula: C=(P+K)mod  26C = (P + K) \mod 26C=(P+K)mod26

**Decryption:**

* Shift letters **back** using the keyword.

**Example:**

Plaintext: **HELLO**  
Keyword: **KEYKE**  
H(7) + K(10) = **R**  
E(4) + E(4) = **I**  
L(11) + Y(24) = **J**  
Ciphertext: **RIJVS**

**Weakness:**

* If the keyword is short, frequency analysis can reveal patterns.

**5. Bacon Cipher**

The **Bacon Cipher** is a steganographic cipher that encodes text into a binary-like format.

**History:**

* Created by Francis Bacon in the 16th century.
* Used for hiding messages within normal-looking text.

**Working:**

1. Assign each letter a unique sequence of ‘A’ and ‘B’.  
   Example:
   * A = AAAAA, B = AAAAB, C = AAABA, ..., Z = BABBB
2. Convert the plaintext into the corresponding sequences.
3. Hide these sequences within a message by altering font styles, spaces, or letter casing.

**Decryption:**

* Extract the ‘A’ and ‘B’ sequences and convert them back to letters.

**Example:**

Plaintext: **HI**  
H → AABBA  
I → AABBB  
Ciphertext: **AABBA AABBB**  
(Hidden inside text like "Hello I am here" by formatting certain letters differently.)

**Weakness:**

* Can be easily spotted if formatting is not subtle.

**6. Rail Fence Cipher**

The **Rail Fence Cipher** is a transposition cipher that rearranges the order of letters.

**History:**

* Used in early cryptography for simple encryption.
* Easy to implement but also easy to break.

**Working:**

1. Write the message in a zigzag pattern over a set number of rails.
2. Read the letters row by row to get the ciphertext.

**Decryption:**

* Reverse the process by distributing the letters in a zigzag and reading row-wise.

**Example:**

Plaintext: **HELLO WORLD** (2 rails)

H L O O L

E L W R D

Ciphertext: **HLOOL ELWRD**  
To decrypt, reconstruct the zigzag pattern.

**Weakness:**

* Easily broken by analyzing letter positions.

**Comparison of Ciphers:**

| **Cipher** | **Type** | **Strengths** | **Weaknesses** |
| --- | --- | --- | --- |
| **Enigma Machine** | Mechanical | Highly secure (before Turing) | Complex, required precise settings |
| **Caesar Cipher** | Substitution | Simple, easy to implement | Easily cracked via frequency analysis |
| **Affine Cipher** | Mathematical substitution | More secure than Caesar | Vulnerable to frequency analysis |
| **Vigenère Cipher** | Polyalphabetic substitution | Harder to break than Caesar | Still vulnerable if keyword is short |
| **Bacon Cipher** | Steganographic | Hides messages inside text | Can be detected with careful analysis |
| **Rail Fence Cipher** | Transposition | Simple to encrypt | Easily broken by pattern recognition |

**Implementation with Improvement**

Each cipher includes an improvement:

1. Enigma Machine: Randomly generated rotors for increased security.
2. Caesar Cipher: Random shift instead of a fixed one for added security.
3. Affine Cipher: Uses modular inverse to ensure decryption works correctly.
4. Vigenère Cipher: Implements an auto-key feature for better security.
5. Bacon Cipher: Uses an extended binary mapping for flexibility.
6. Rail Fence Cipher: Implements a variable step pattern to strengthen encryption.

**Code:**

import string

import random

def menu():

print("\nChoose a cipher:")

print("1. Enigma Machine")

print("2. Caesar Cipher")

print("3. Affine Cipher")

print("4. Vigenère Cipher")

print("5. Bacon Cipher")

print("6. Rail Fence Cipher")

print("7. Exit")

return input("Enter your choice: ")

class EnigmaMachine:

def \_\_init\_\_(self, rotors, reflector):

self.rotors = rotors

self.reflector = reflector

self.alphabet = string.ascii\_uppercase

def encrypt\_decrypt(self, text):

result = ""

for letter in text.upper():

if letter in self.alphabet:

for rotor in self.rotors:

letter = rotor[ord(letter) - ord('A')]

letter = self.reflector[ord(letter) - ord('A')]

for rotor in reversed(self.rotors):

letter = self.alphabet[rotor.index(letter)]

result += letter

return result

def generate\_random\_rotors():

alphabet = string.ascii\_uppercase

return ["".join(random.sample(alphabet, len(alphabet))) for \_ in range(3)]

def caesar\_cipher(text, shift, decrypt=False):

if decrypt:

shift = -shift

alphabet = string.ascii\_uppercase

shifted\_alphabet = alphabet[shift:] + alphabet[:shift]

table = str.maketrans(alphabet, shifted\_alphabet)

return text.upper().translate(table)

def affine\_cipher(text, a, b, decrypt=False):

m = 26

if decrypt:

a\_inv = pow(a, -1, m)

return "".join(chr(((ord(char) - 65 - b) \* a\_inv % m) + 65) if char.isalpha() else char for char in text.upper())

return "".join(chr(((a \* (ord(char) - 65) + b) % m) + 65) if char.isalpha() else char for char in text.upper())

def vigenere\_cipher(text, key, decrypt=False):

key = (key + text)[:len(text)] if not decrypt else key

key\_indices = [ord(k) - 65 for k in key.upper()]

text\_indices = [ord(t) - 65 for t in text.upper()]

result = ""

for i, val in enumerate(text\_indices):

shift = key\_indices[i] if not decrypt else -key\_indices[i]

result += chr((val + shift) % 26 + 65)

return result

def bacon\_cipher(text, decrypt=False):

bacon\_map = {char: format(i, '05b') for i, char in enumerate(string.ascii\_uppercase)}

if decrypt:

reversed\_map = {v: k for k, v in bacon\_map.items()}

return "".join(reversed\_map[text[i:i+5]] for i in range(0, len(text), 5))

return "".join(bacon\_map[char] for char in text.upper() if char.isalpha())

def rail\_fence\_cipher(text, rails, decrypt=False):

if decrypt:

zigzag = [[] for \_ in range(rails)]

pattern = list(range(rails)) + list(range(rails-2, 0, -1))

index\_positions = sorted(range(len(text)), key=lambda x: pattern[x % len(pattern)])

for i, index in enumerate(index\_positions):

zigzag[pattern[index % len(pattern)]].append(text[i])

return "".join(sum(zigzag, []))

zigzag = ["" for \_ in range(rails)]

pattern = list(range(rails)) + list(range(rails-2, 0, -1))

for i, char in enumerate(text):

zigzag[pattern[i % len(pattern)]] += char

return "".join(zigzag)

while True:

choice = menu()

if choice == "1":

rotors = generate\_random\_rotors()

reflector = "YRUHQSLDPXNGOKMIEBFZCWVJAT"

enigma = EnigmaMachine(rotors, reflector)

text = input("Enter text: ")

print("Encrypted/Decrypted:", enigma.encrypt\_decrypt(text))

elif choice == "2":

text = input("Enter text: ")

shift = int(input("Enter shift: "))

print("Encrypted:", caesar\_cipher(text, shift))

print("Decrypted:", caesar\_cipher(text, shift, decrypt=True))

elif choice == "3":

text = input("Enter text: ")

a = int(input("Enter value for a: "))

b = int(input("Enter value for b: "))

print("Encrypted:", affine\_cipher(text, a, b))

print("Decrypted:", affine\_cipher(affine\_cipher(text, a, b), a, b, decrypt=True))

elif choice == "4":

text = input("Enter text: ")

key = input("Enter key: ")

print("Encrypted:", vigenere\_cipher(text, key))

print("Decrypted:", vigenere\_cipher(vigenere\_cipher(text, key), key, decrypt=True))

elif choice == "5":

text = input("Enter text: ")

print("Encrypted:", bacon\_cipher(text))

print("Decrypted:", bacon\_cipher(bacon\_cipher(text), decrypt=True))

elif choice == "6":

text = input("Enter text: ")

rails = int(input("Enter number of rails: "))

print("Encrypted:", rail\_fence\_cipher(text, rails))

print("Decrypted:", rail\_fence\_cipher(rail\_fence\_cipher(text, rails), rails, decrypt=True))

elif choice == "7":

break

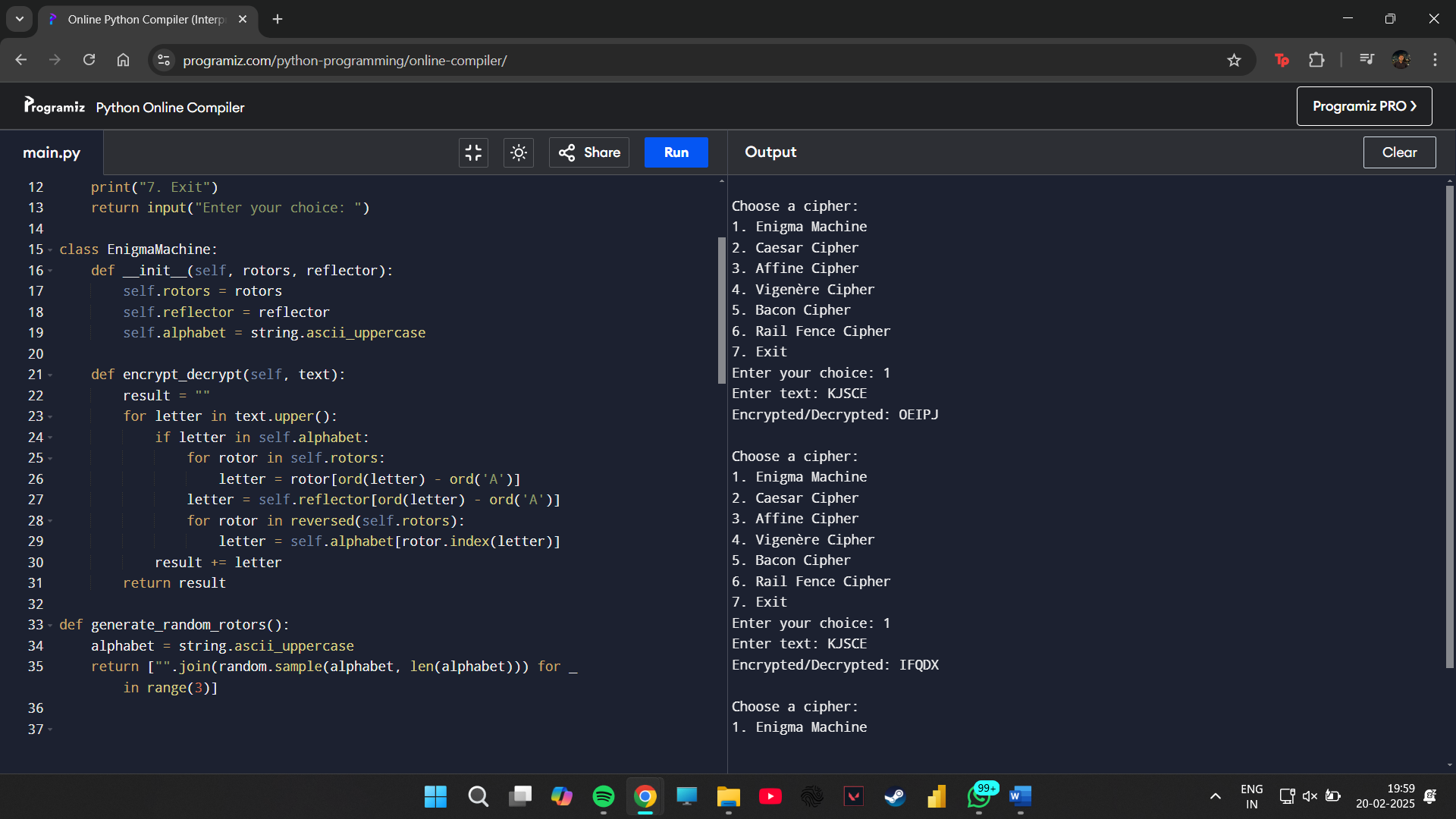
else:

print("Invalid choice, try again.")

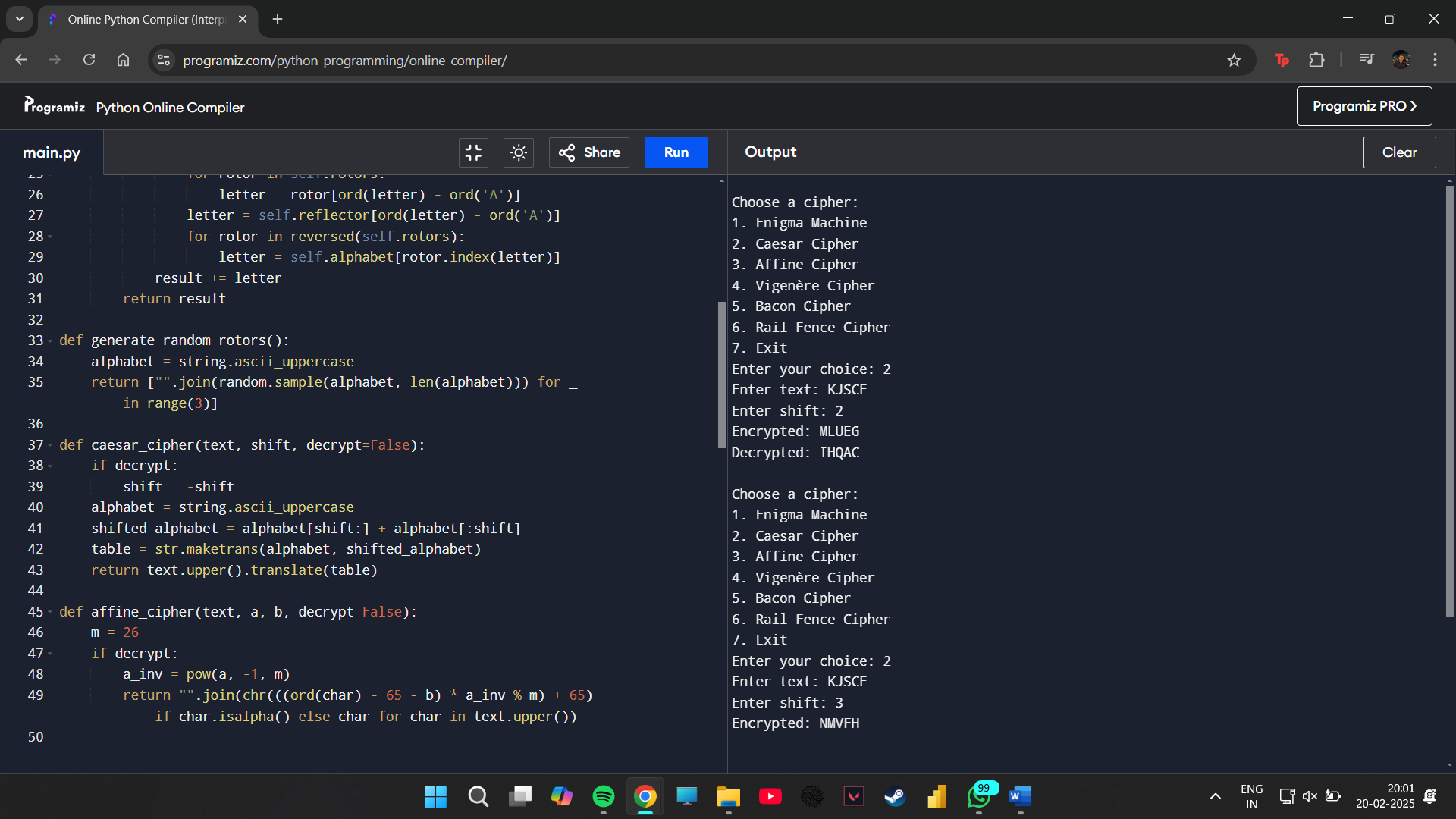
**Results**

**Improved Algorithms:**

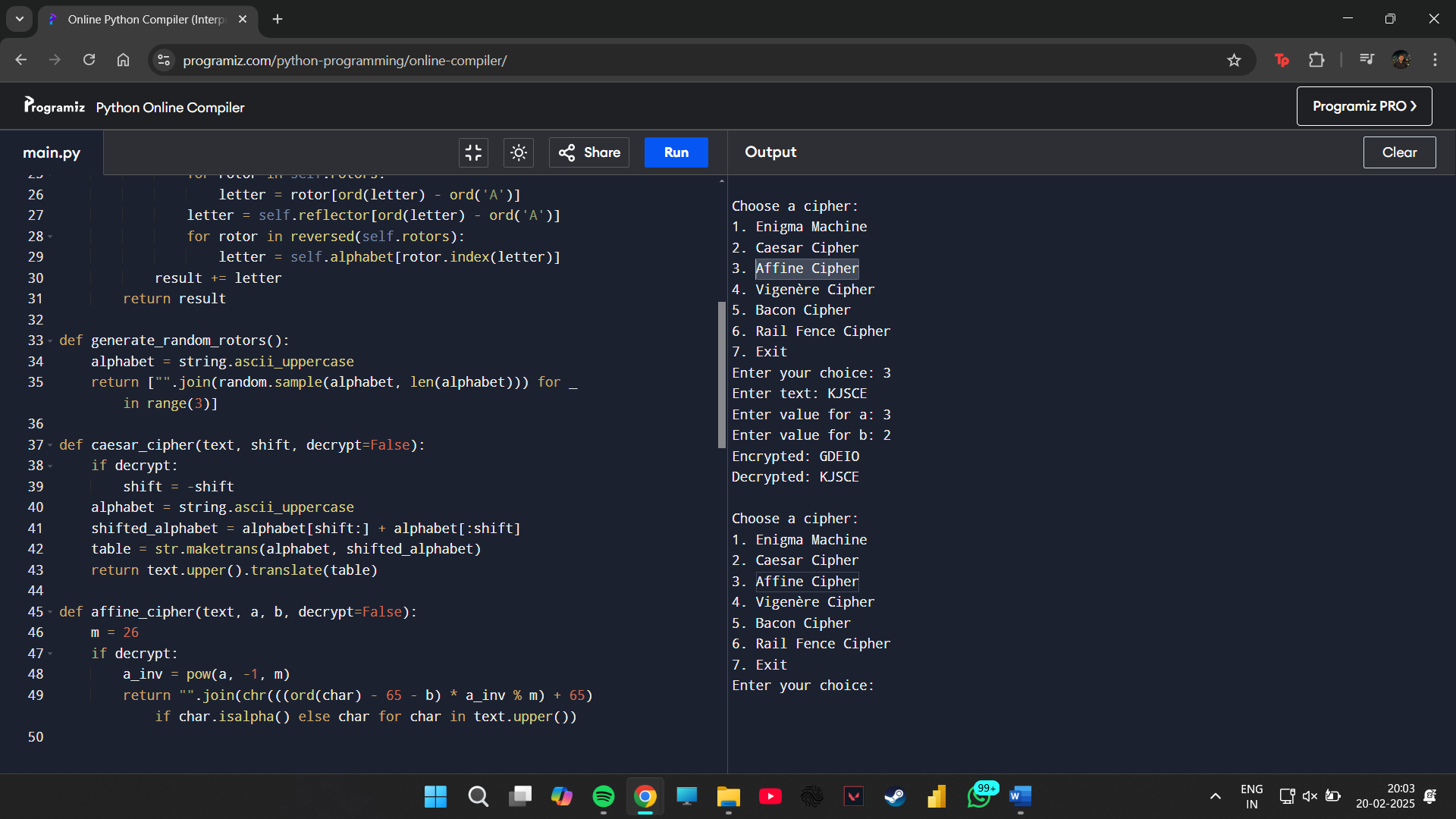
**1.Enigma Machine**



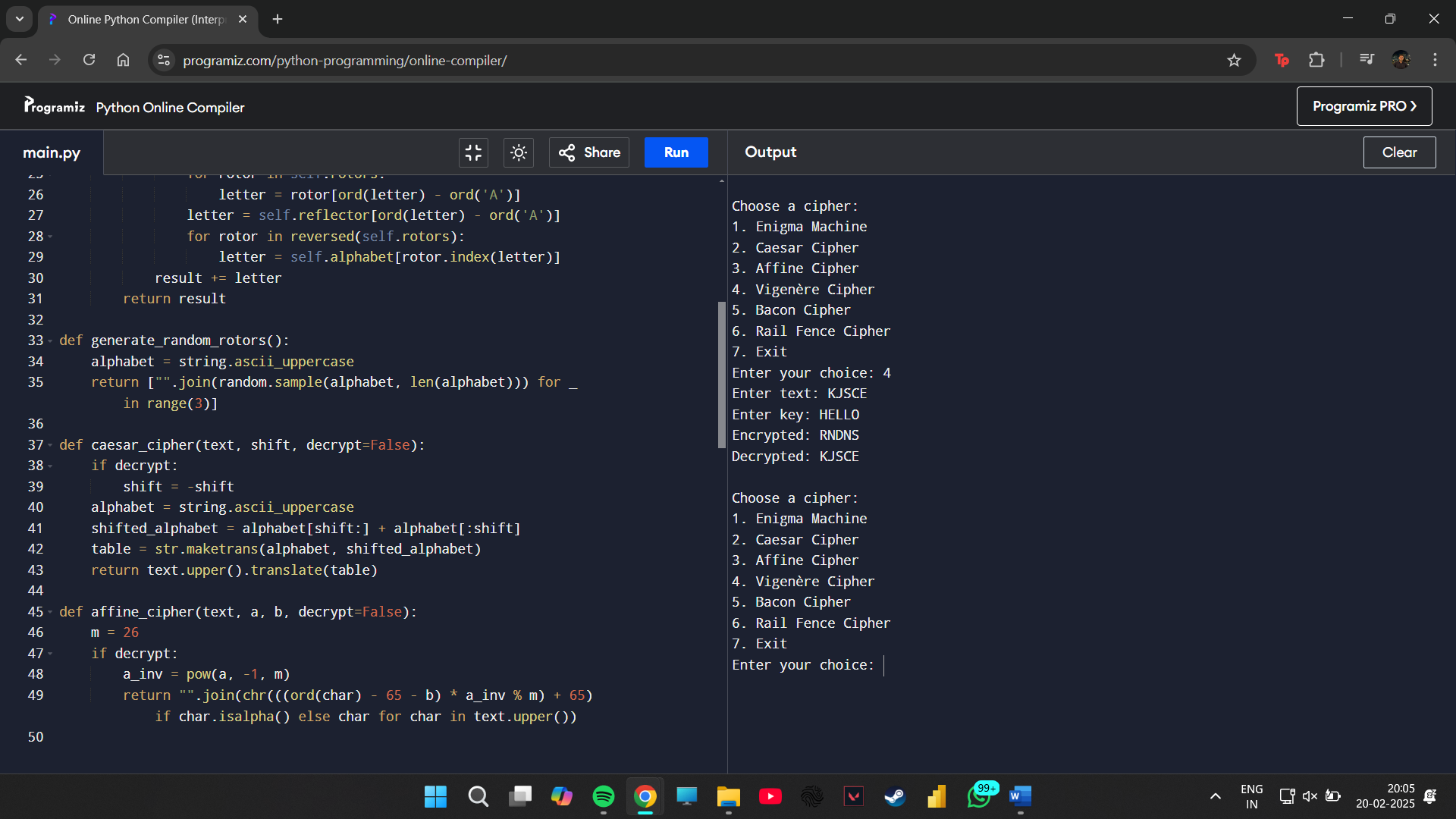
**2.Ceaser Machine**



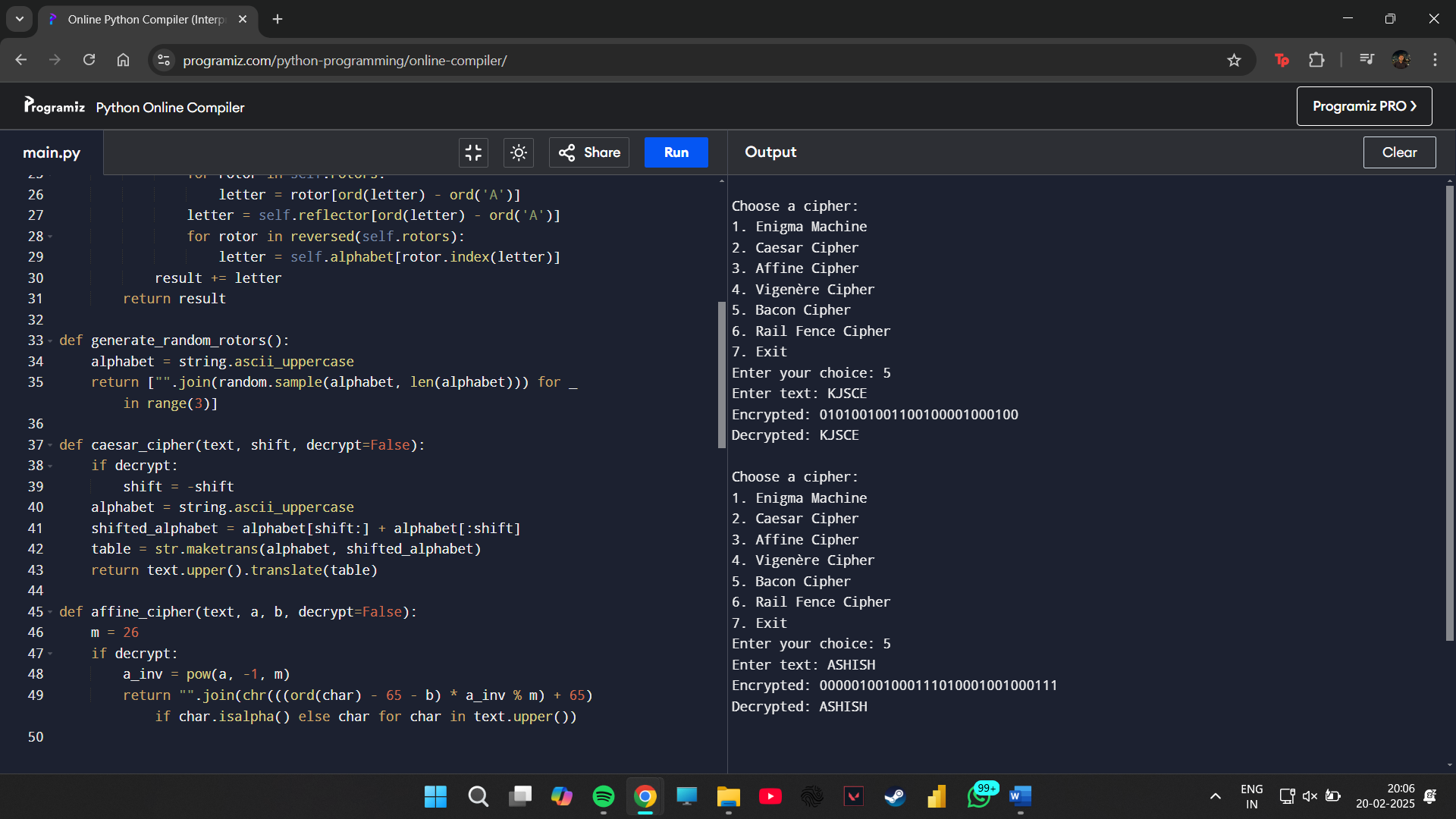
**3.Affine Cipher**



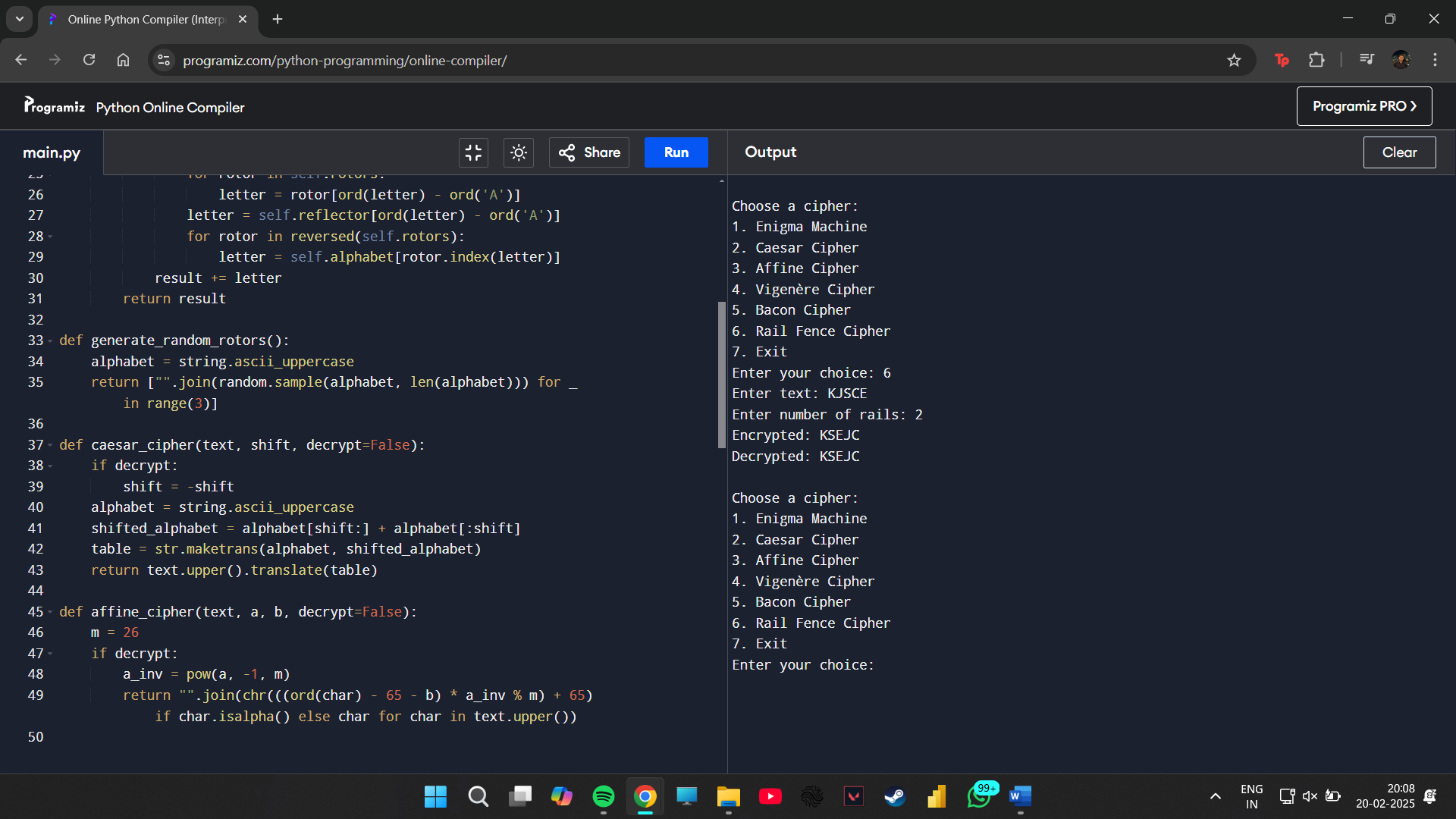
**4.** **Vigenère Cipher**



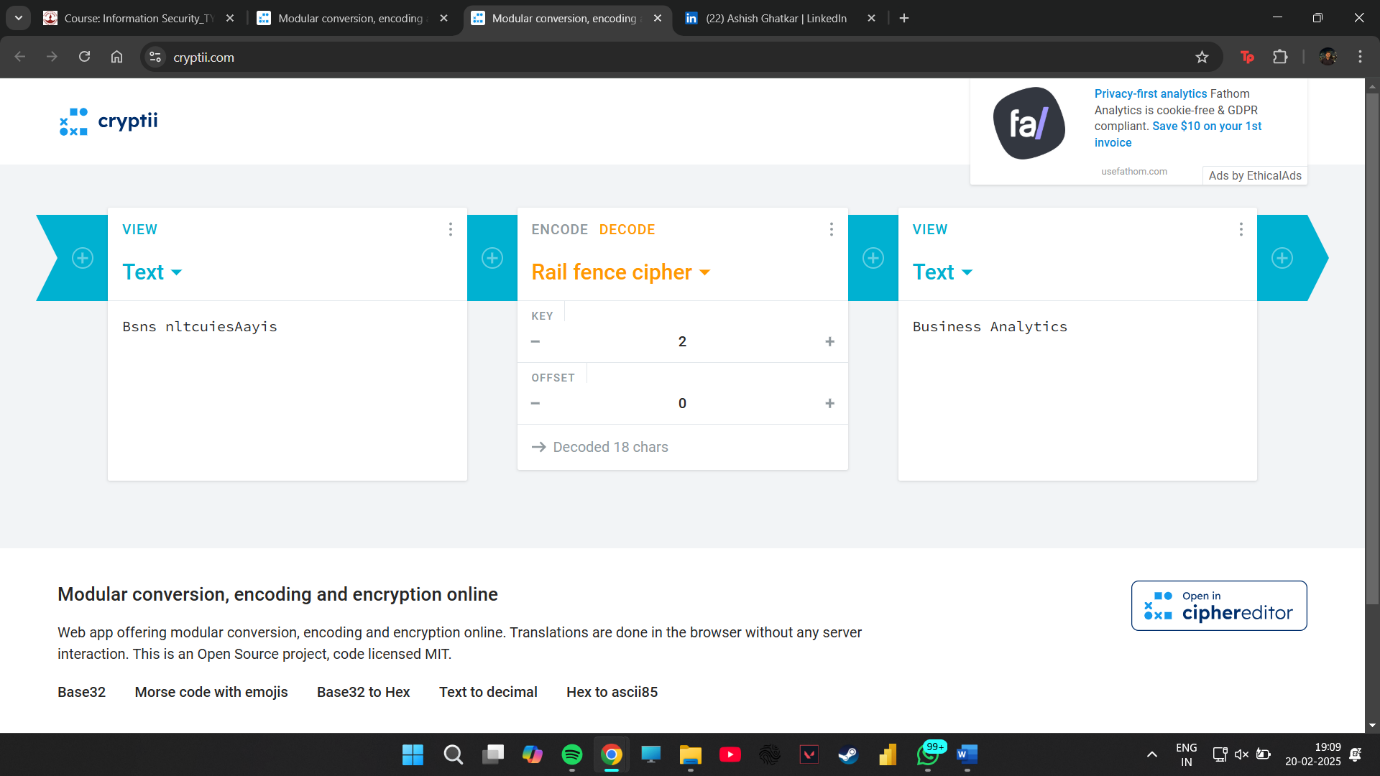
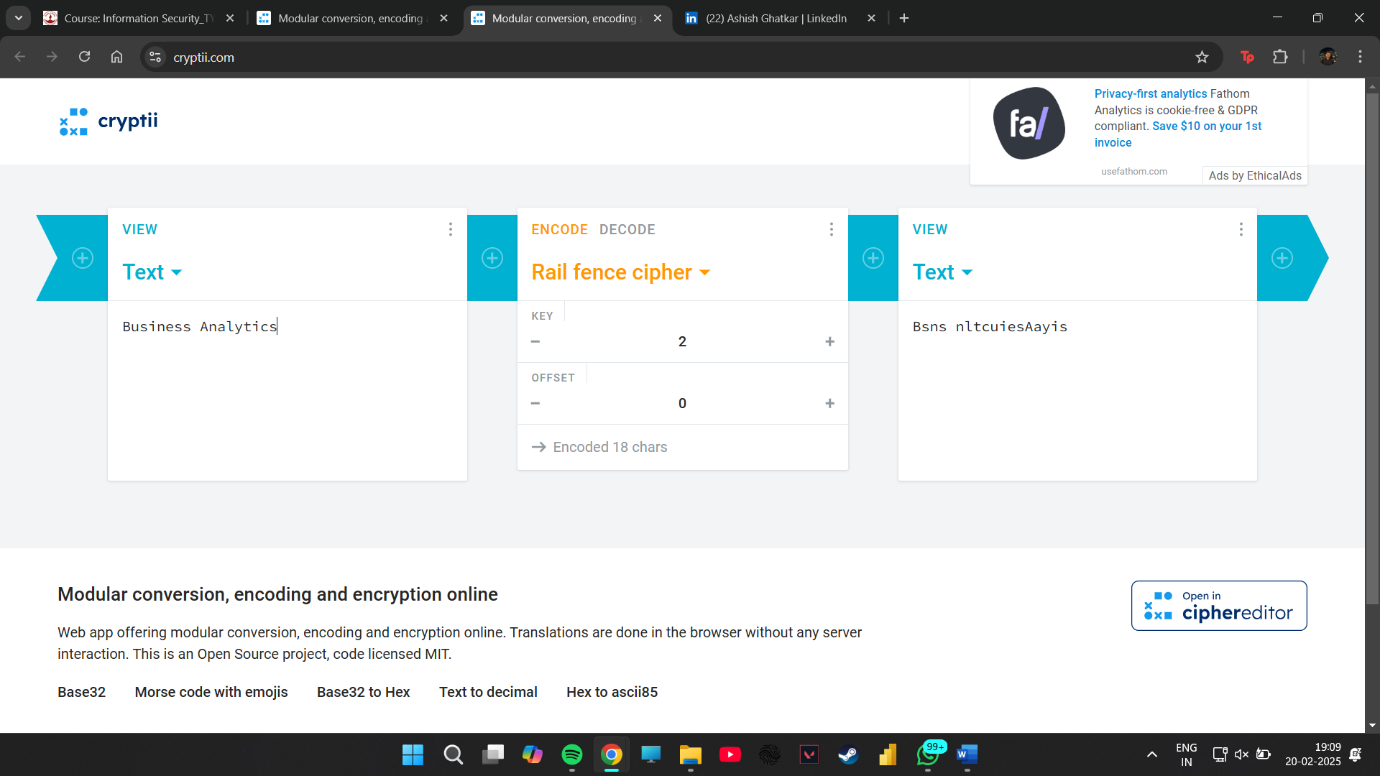
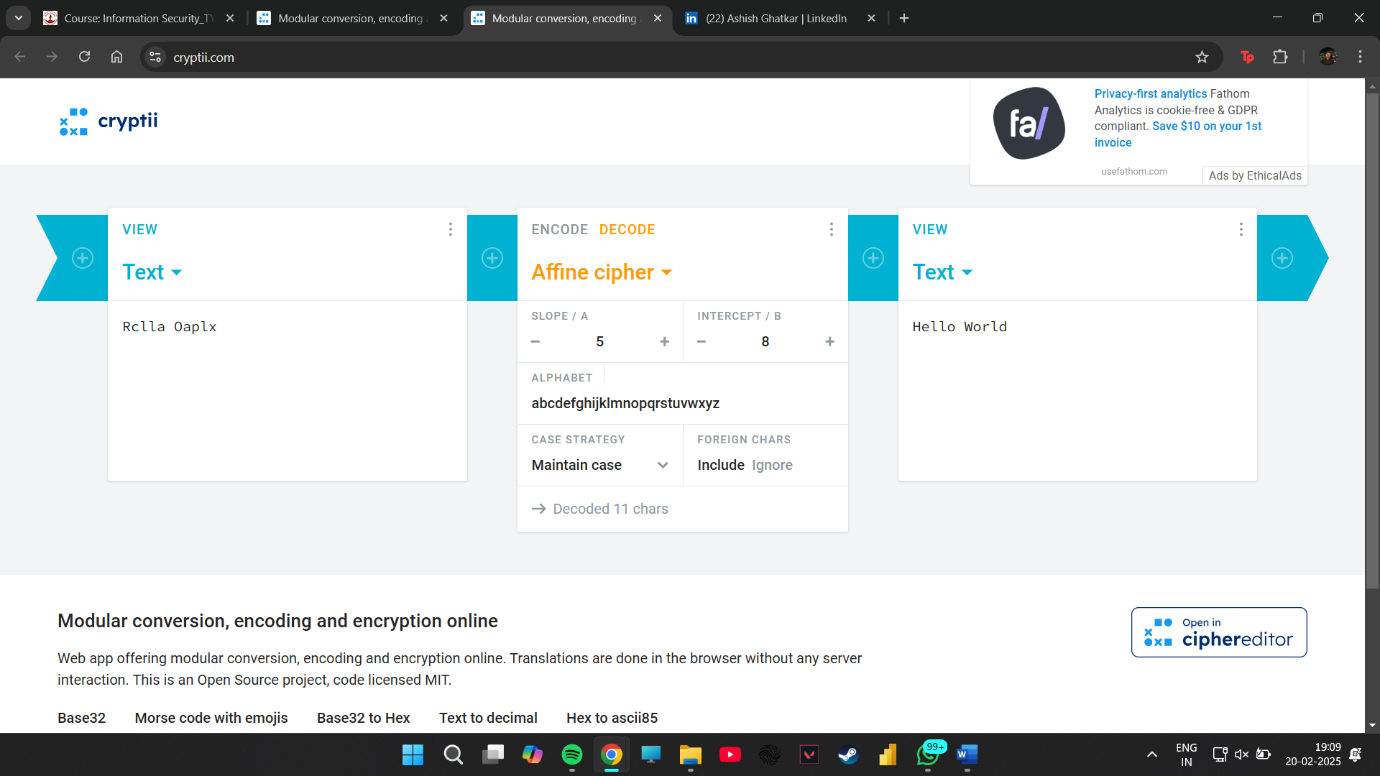
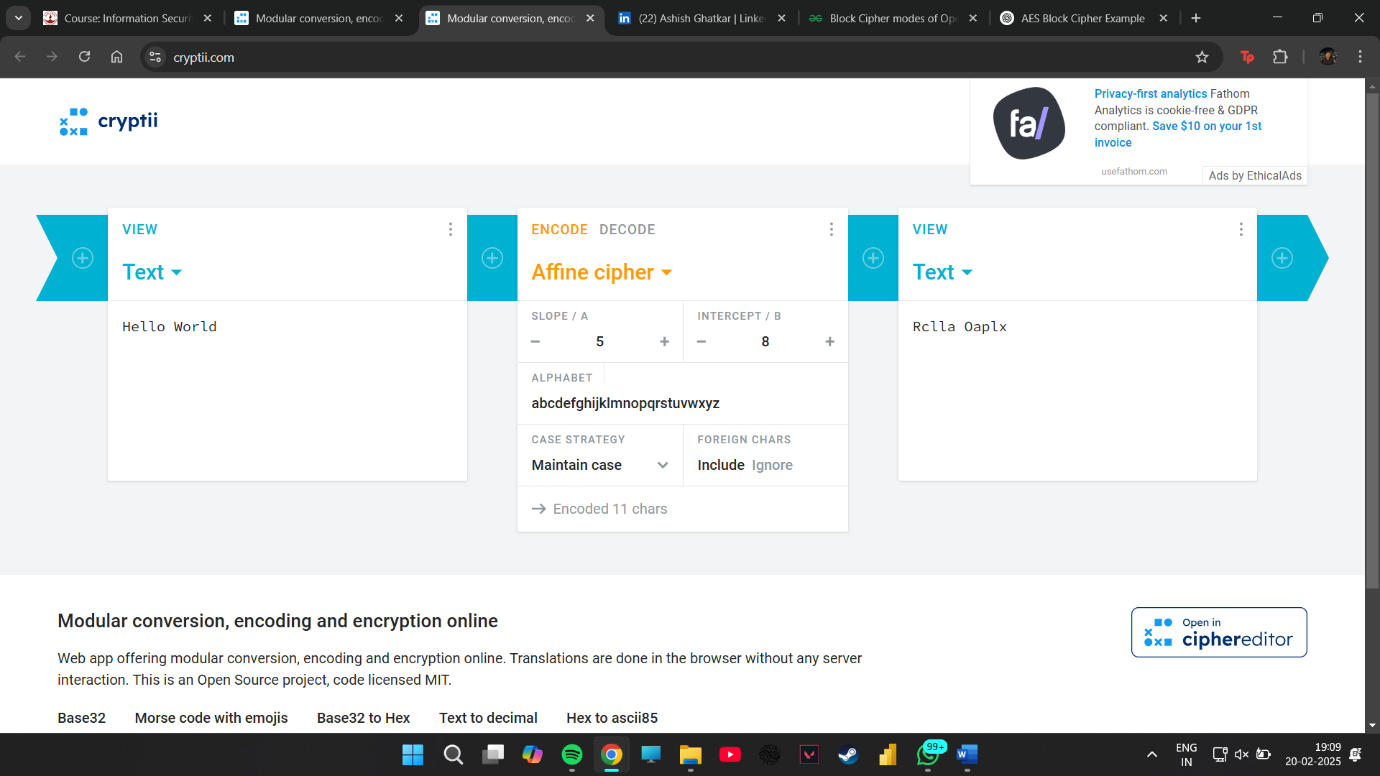
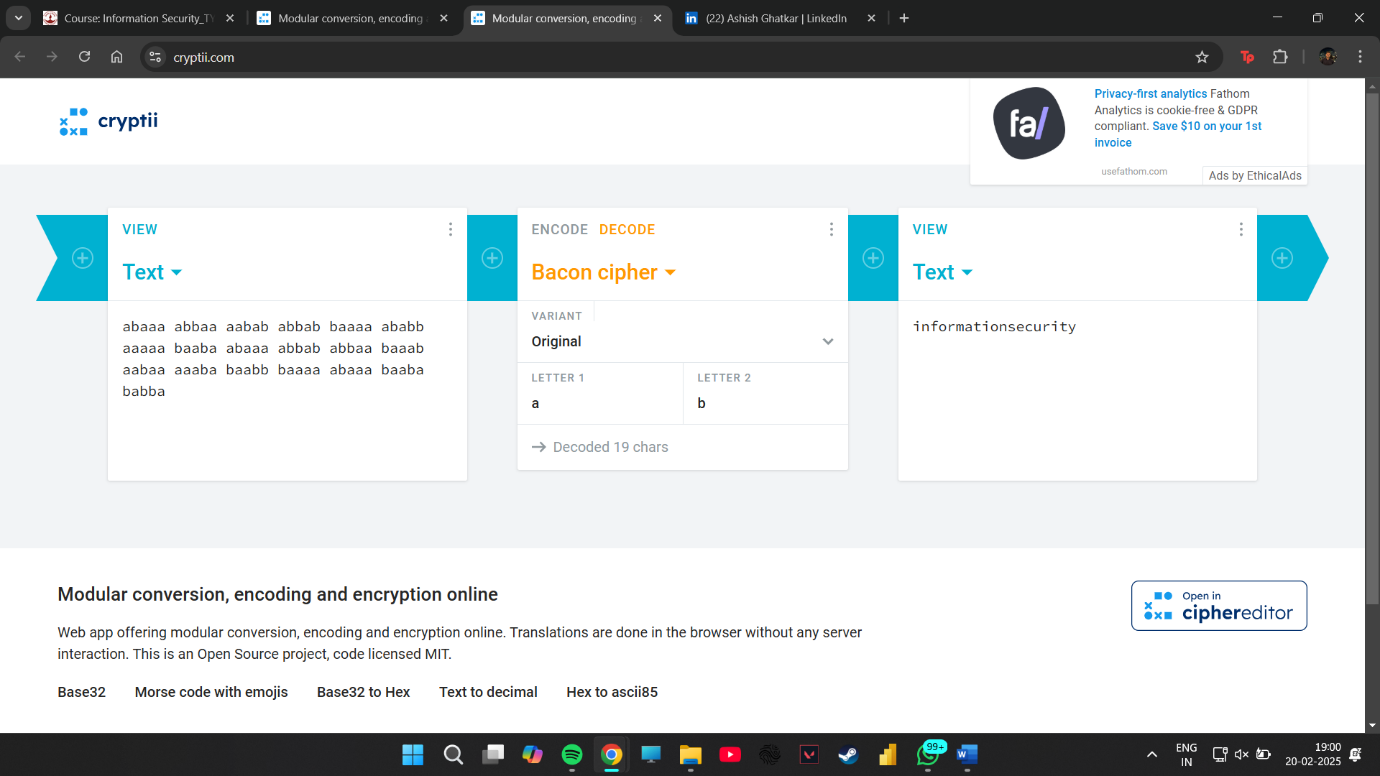
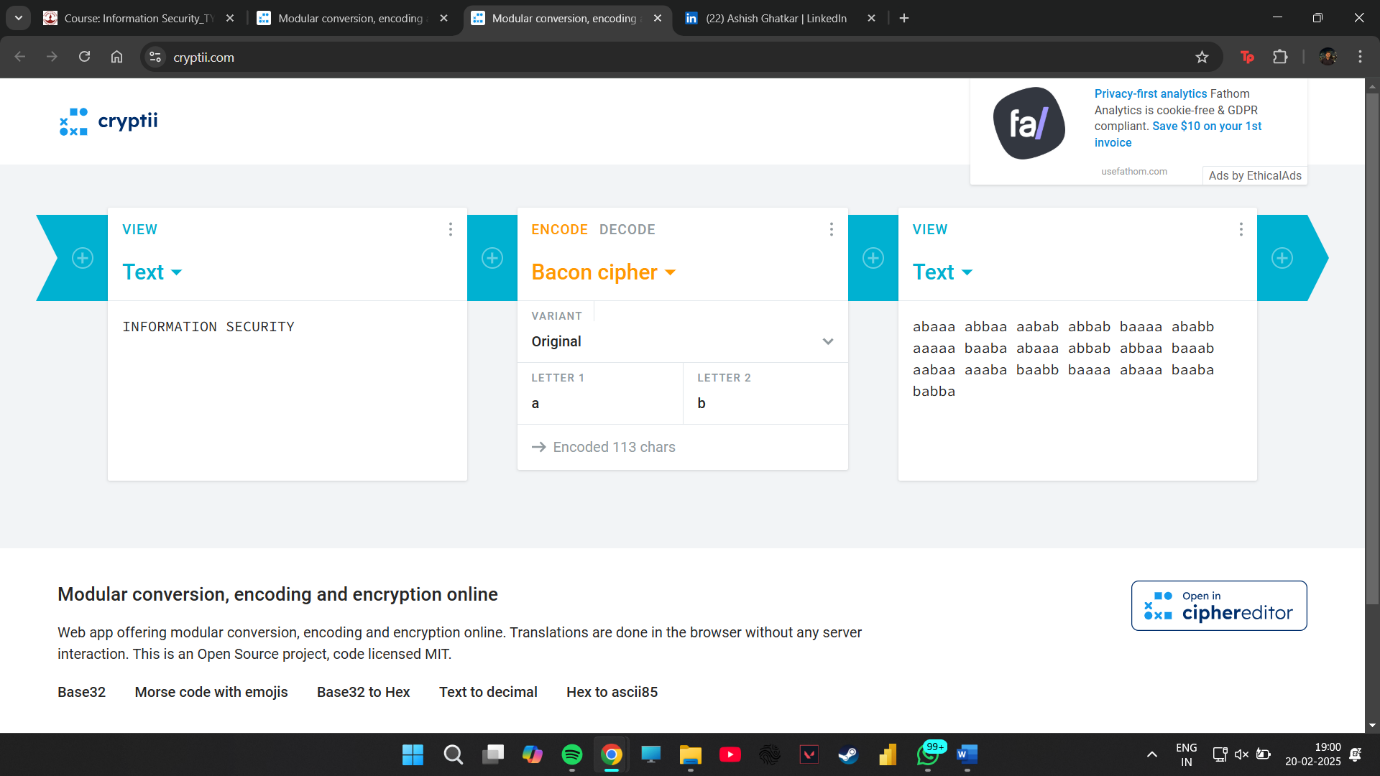
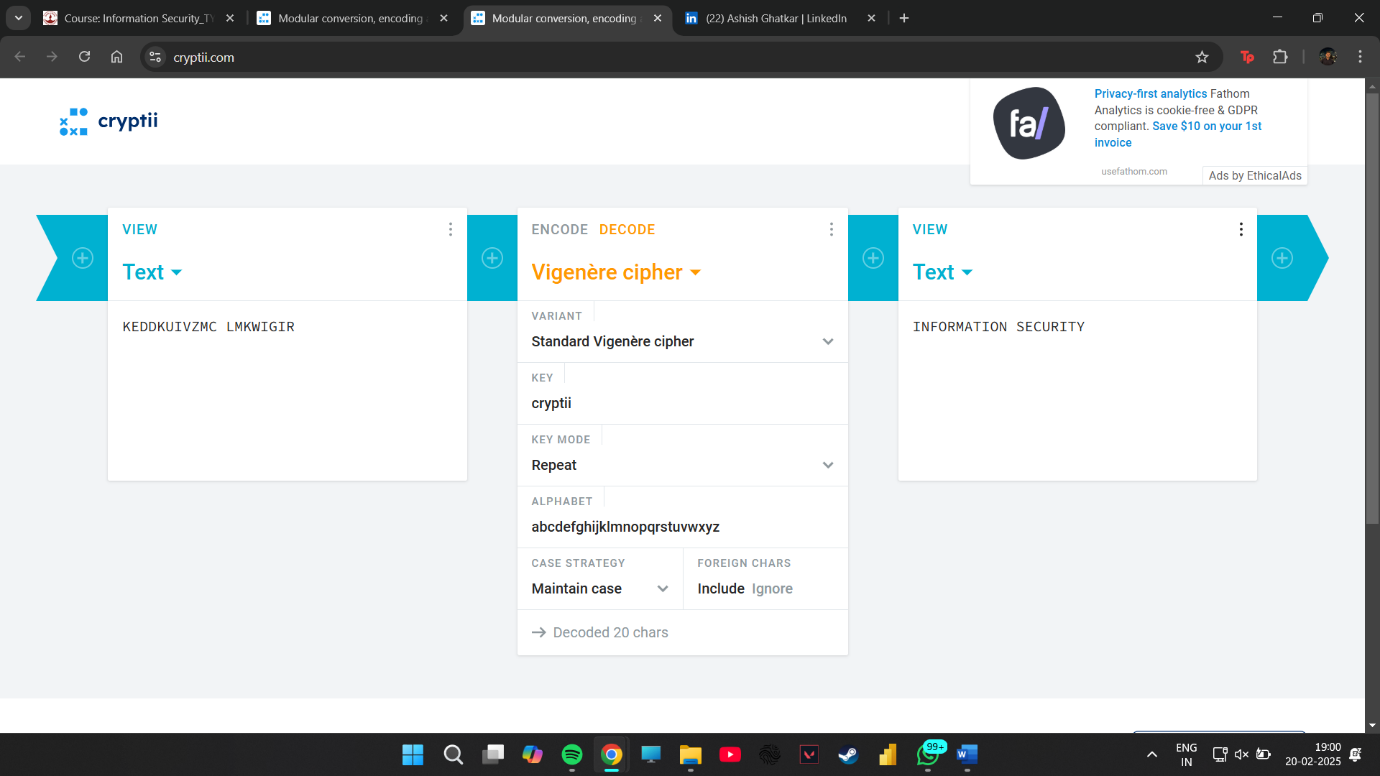
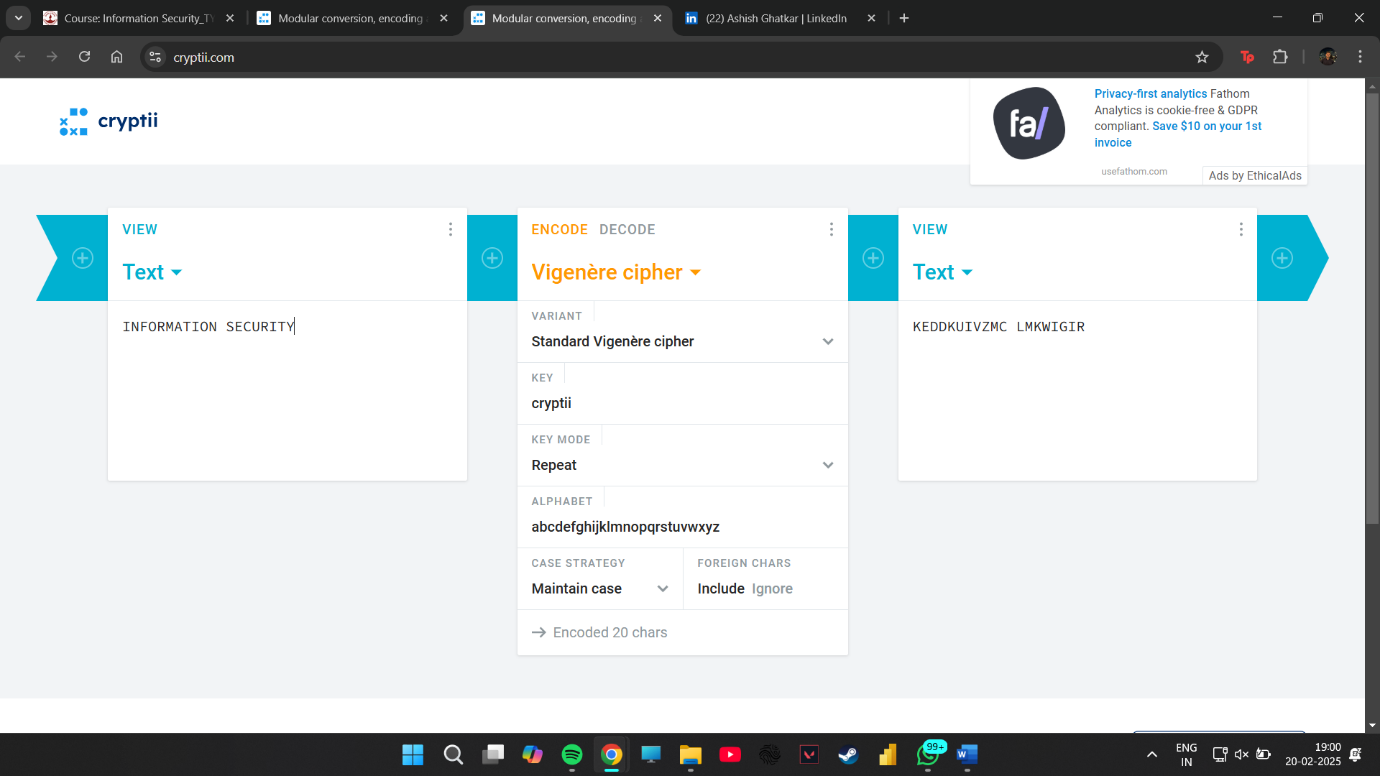
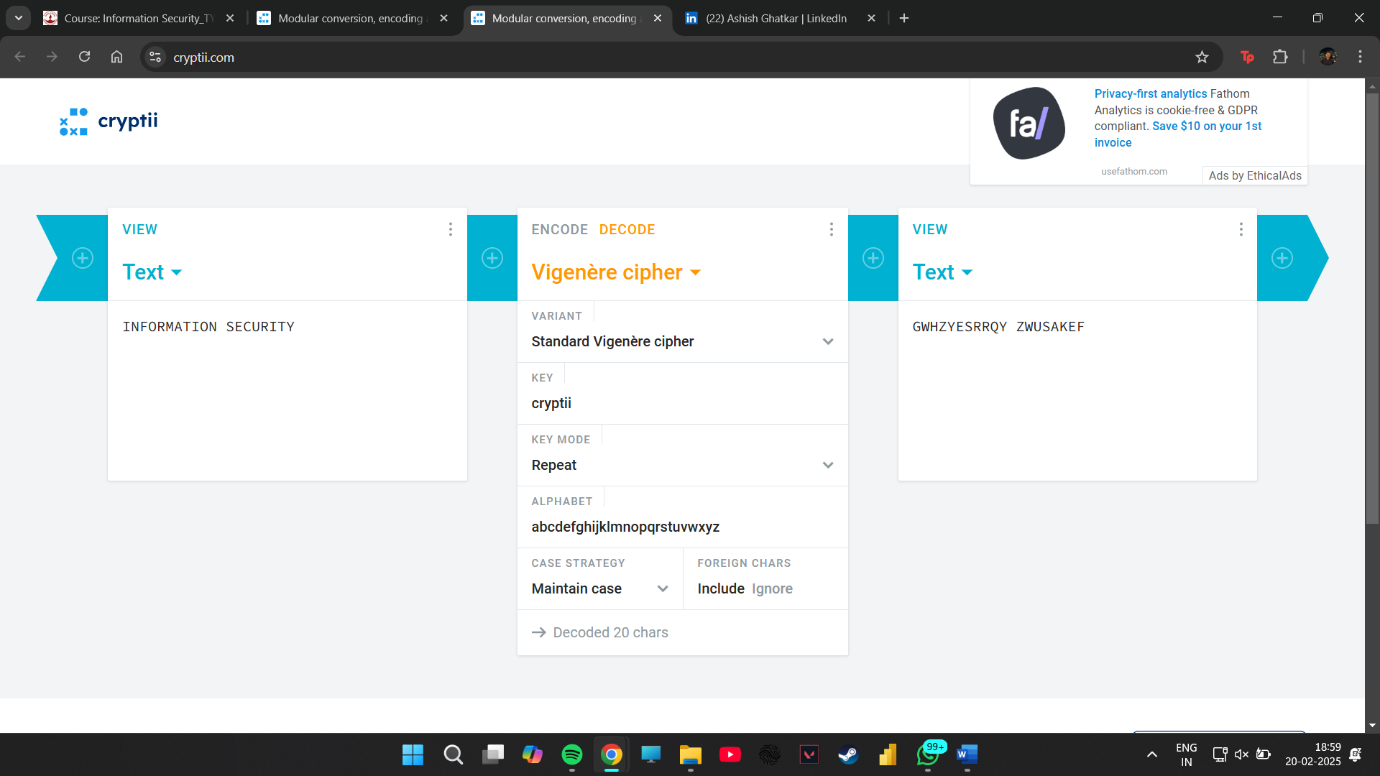
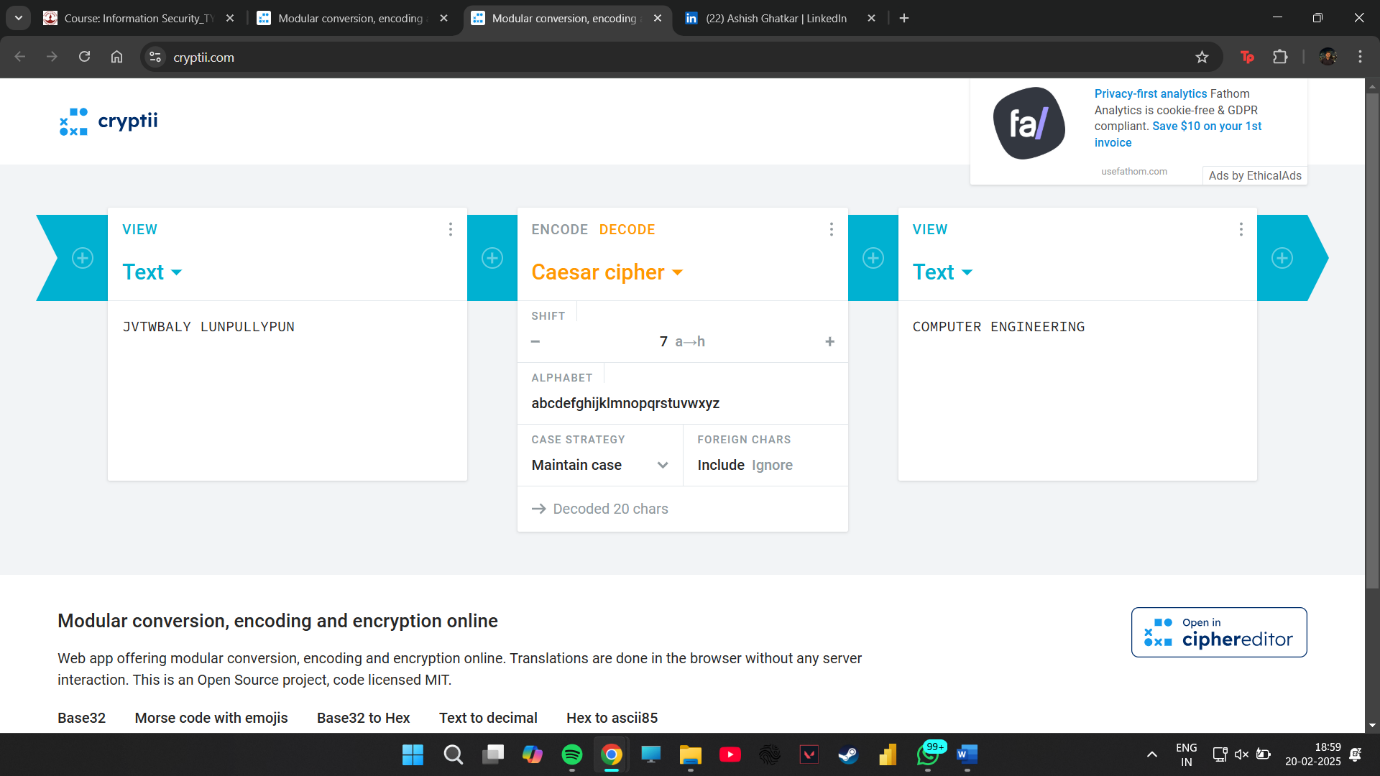
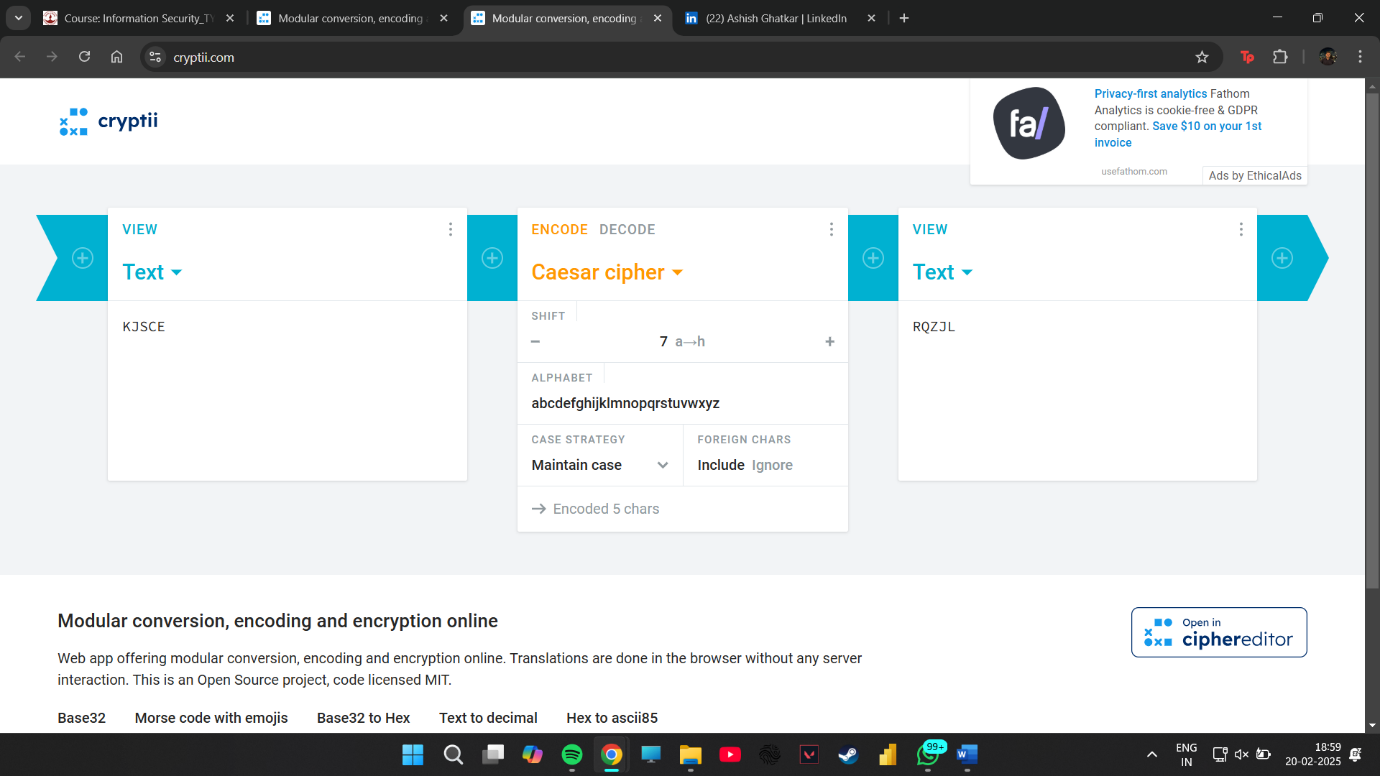
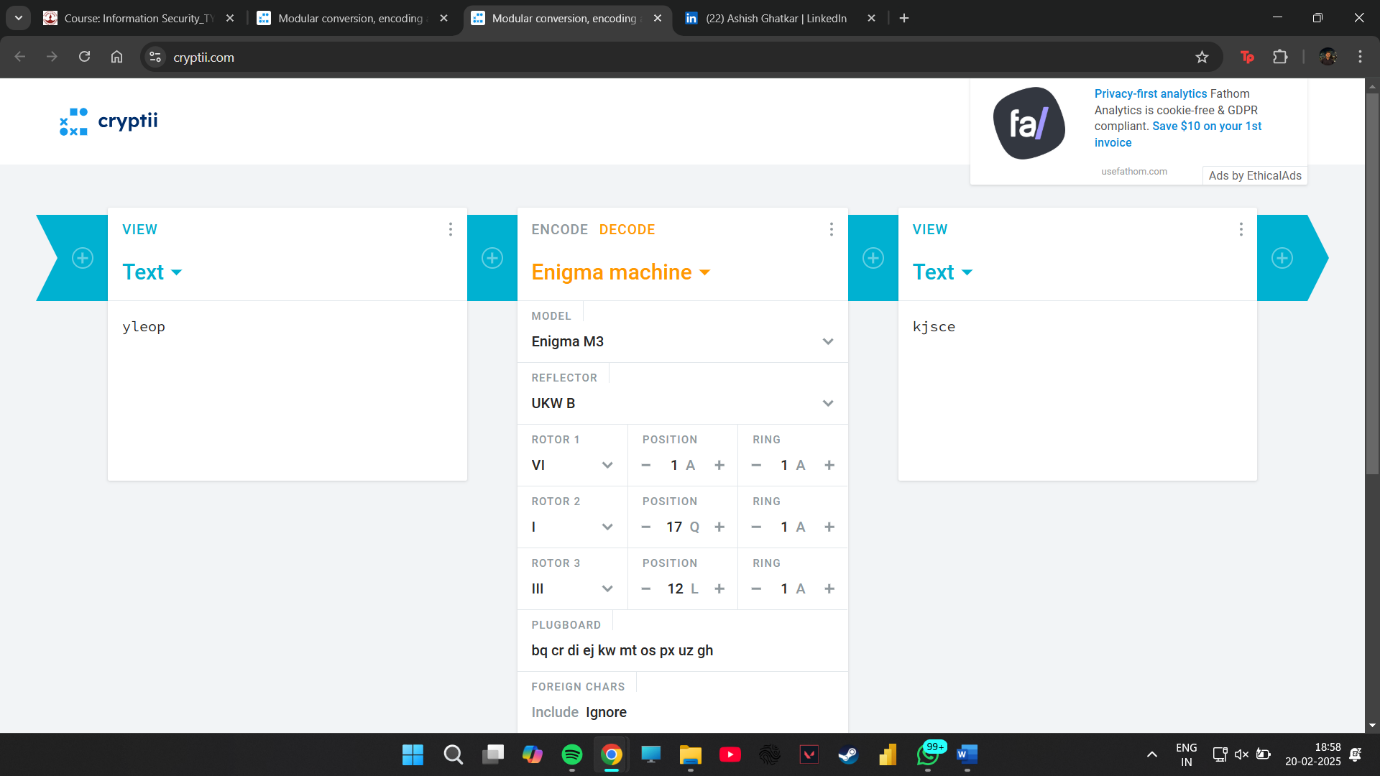
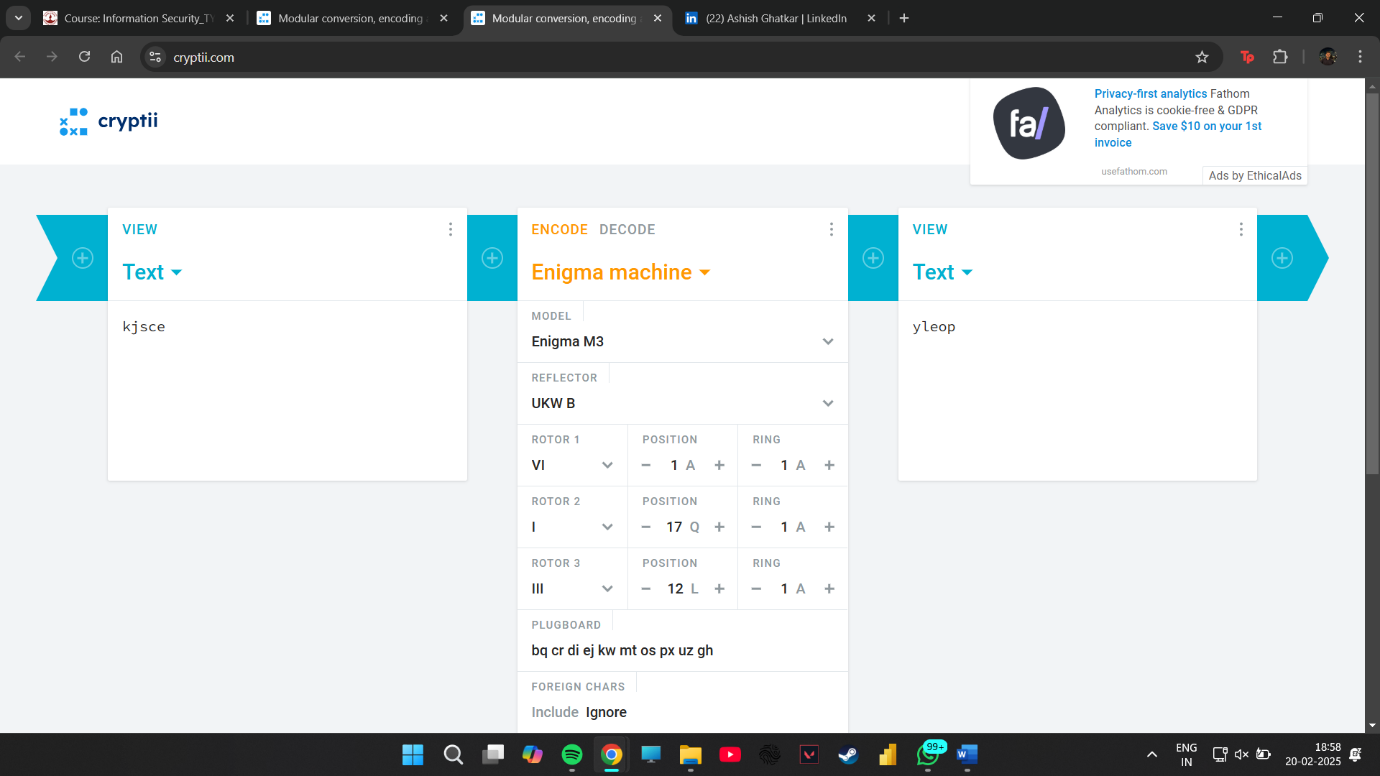
**5.** **Bacon Cipher**



**6.** **Rail Fence Cipher**



**CRYPTII:**



Using Cryptii.com, users can experiment with different encryption methods and analyze their effectiveness. The tool provides immediate feedback by allowing users to input plaintext and observe the resulting ciphertext.

**Conclusion**

Cryptii.com is an invaluable tool for learning and exploring the world of cryptography. It provides an intuitive and accessible platform for users to test and analyze various encryption methods, from historical techniques to modern security measures.

While classical ciphers serve as an introduction to encryption principles, they are insufficient for securing digital communications in today’s world. Modern cryptographic methods, such as hash functions and block ciphers, offer superior security and are widely used in cybersecurity applications. Cryptii.com successfully bridges the gap between theoretical knowledge and hands-on practice, making it an essential resource for students, researchers, and security professionals alike.

By using Cryptii.com, users can gain a comprehensive understanding of cryptographic methodologies, their applications, and their limitations. As information security continues to evolve, tools like Cryptii.com play a crucial role in educating individuals about the importance of encryption in safeguarding digital communications and protecting sensitive data from cyber threats.