VIVEKANANDA
GLOBAL
UNIVERSITY

BACHLOR OF COMPUTER APPLICATION

CRYPTOGRAPHY & NETWORK SECURITY (UGCSA215)

Project Title: Creating a Python Program to Perform Simple Cryptanalysis on Ciphers

PROJECT GUIDE:
Mr. NARAYAN VYAS

SUBMITTED BY:
SAHIL KR SINGH (23CSA2BC295)
VIRAJ KHATRI (23CSA2BC241)
HARSHIT KR VERMA (23CSA2BC185)
RAJEEV KUMAR (23CSA2BC266)
VIJAY KUMAR (23CSA2BC319)

ACKNOWLEDGEMENT

I have taken this opportunity to express my gratitude and humble regards to the Vivekananda Global University to provide an opportunity to present a project on the "Creating a Python Program to Perform Simple Cryptanalysis on Ciphers," which is a Cryptography & Network Security A6 Project.

I would also be thankful to my project guide **Mr. Naryan Vyas** to help me in the completion of my project and the documentation. I have taken efforts in this project, but the success of this project would not be possible without their support and encouragement.

I would like to thanks our Dean sir "**Dr. R C Tripathi**" to help us in providing all the necessary books and other stuffs as and when required. I show my gratitude to the authors whose books has been proved as the guide in the completion of my project I am also thankful to my classmates and friends who have encouraged me in the course of completion of the project.

Thanks

SAHIL KR SINGH (23CSA2BC295)
VIRAJ KHATRI (23CSA2BC241)
HARSHIT KR VERMA (23CSA2BC185)
RAJEEV KUMAR (23CSA2BC266)
VIJAY KUMAR (23CSA2BC319)

Place : Jaipur
Date : 01-04-2025

DECLARATION

We hereby declare that this Project Report titled “**Creating a python program to perform a simple cryptanalysis on ciphers”** submitted by us and approved by our project guide, to the Vivekananda Global University, Jaipur is a bonafide work undertaken by us and it is not submitted to any other University or Institution for the award of any degree diploma / certificate or published any time before.

Student Name:

SAHIL KR SINGH (23CSA2BC295)
VIRAJ KHATRI (23CSA2BC241)
HARSHIT KR VERMA (23CSA2BC185)
RAJEEV KUMAR (23CSA2BC266)
VIJAY KUMAR (23CSA2BC319)

Project Guide: Mr. Narayan Vyas

CONTENTS

1.	**Project Title……………………………………………….**	5
2.	**Problem Statement…………………………………….**	5-6
3.	**Project Description……………………………………..**	6
3.1	**Scope of Work …………………………………………..**	6-7
3.2	**Project Modules………………………………………..**	7-8
3.3	**Context Diagram……………………………………….**	8-10
4.	**Implementation of Methodology……………………….**	10-11
5.	**Technologies to be used……………………………….**	11-12
6.	**Software Platform……………………………………….**	12-13
7.	**Hardware Platform…………………………………….**	13
8.	**Advantages of this Project…………………………….**	13-14
9.	**Screenshots and Output…………………………………………**	14
10.	**Conclusion……………………………………………..**	15
11.	**References……………………………………………..**	16

1. Introduction

Cryptanalysis is the practice of breaking encryption systems to reveal hidden messages without the decryption key. It plays a crucial role in cybersecurity, helping to identify weaknesses in encryption techniques. This Python program aims to perform basic cryptanalysis on classical ciphers, including the **Caesar cipher, substitution cipher, and Vigenère cipher**.

The program will use the following techniques:

Frequency Analysis – Identifying letter patterns in ciphertexts to guess plaintext.

Brute Force Attack – Trying all possible keys to decrypt a message.

Dictionary Attack – Using a predefined list of words to find potential plaintext matches.

For example, the **Caesar cipher** can be broken by shifting letters back systematically, while a **substitution cipher** can be cracked using letter frequency comparisons. The **Vigenère cipher**, which uses a repeated keyword, can be decrypted using statistical methods like the Kasiski examination.

This program is useful for security professionals, ethical hackers, and encryption enthusiasts to understand how ciphers work and improve modern encryption methods. By automating cryptanalysis with Python, users can analyze encrypted texts and uncover vulnerabilities in outdated cryptographic systems.

2 Problem Statement

Encryption plays a crucial role in securing sensitive data, but weak encryption methods can be exploited through cryptanalysis. Classical ciphers like the Caesar cipher, substitution cipher, and Vigenère cipher are commonly used in basic encryption but are vulnerable to various attacks. The challenge is to analyze and break such ciphers using automated cryptanalysis techniques.

This project aims to develop a Python program that can decrypt encrypted messages without prior knowledge of the key. The program will implement cryptanalysis techniques such as:

Frequency Analysis – Analyzing letter distribution in ciphertexts to predict plaintext.

Brute Force Attack – Trying all possible decryption keys to find readable text.

Dictionary Attack – Using a list of common words to identify patterns in ciphertext.

3 Project Description

This project focuses on developing a Python-based cryptanalysis tool to analyze and break classical ciphers, including the Caesar cipher, substitution cipher, and Vigenère cipher. These ciphers, while historically significant, are vulnerable to various cryptanalysis techniques. The project will automate the process of decrypting encrypted messages without the decryption key, demonstrating weaknesses in outdated encryption methods.

3.1 Scope of Work

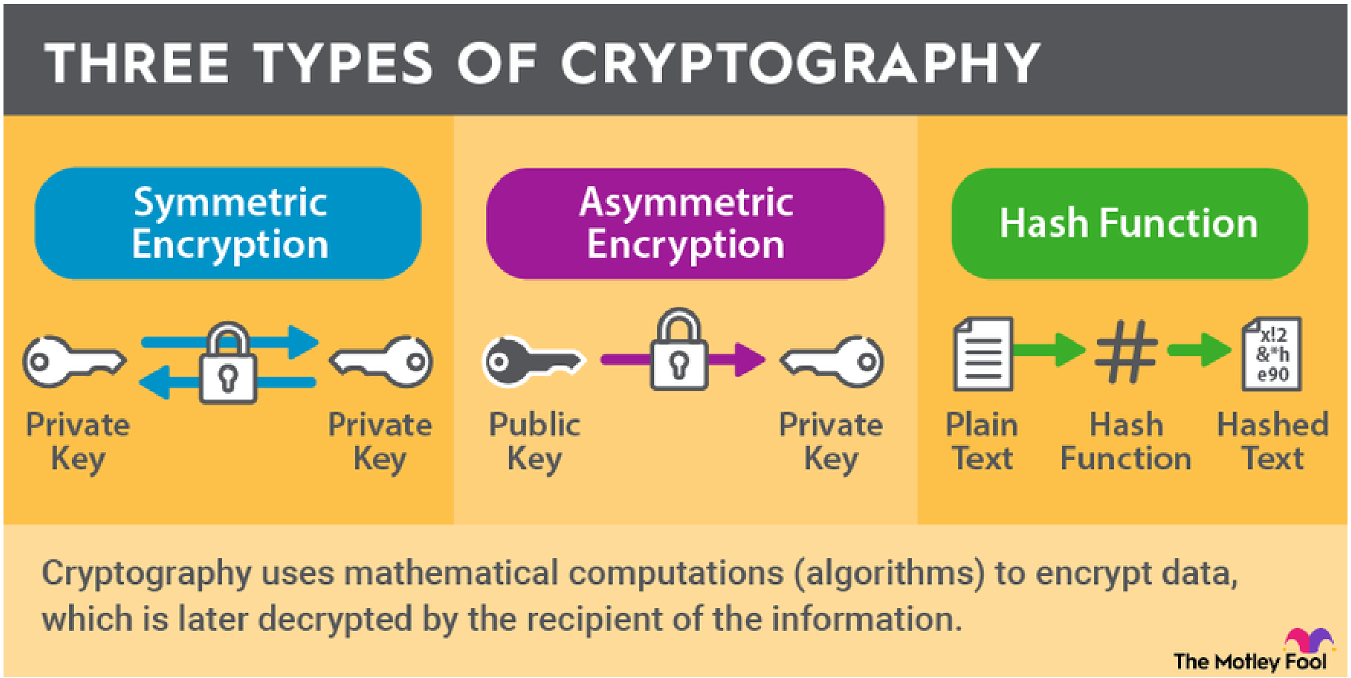
The scope of this project includes the implementation of a Python-based tool that performs cryptanalysis on classical ciphers, using various decryption techniques. The project will focus on automating the analysis and breaking of encrypted text through statistical and computational methods.

- Cipher Identification & Implementation
- Cryptanalysis Techniques
- Automation & Optimization
- User Interface

3.2 Project Modules

This project is divided into several key modules, each designed to perform a specific function in the cryptanalysis process. The **Cipher Implementation Module** focuses on encoding and decoding messages using the **Caesar cipher, substitution cipher, and Vigenère cipher**, ensuring accurate encryption and decryption for testing cryptanalysis techniques. The **Cryptanalysis Techniques Module** applies different methods to break these ciphers, including **brute force attacks**, which systematically try all possible keys, **frequency analysis**, which studies letter distribution to predict plaintext, and **dictionary attacks**, which compare encrypted text with common words. For the **Vigenère cipher**, the **Kasiski Examination** technique is used to determine the encryption key length. The **User Interaction Module** allows users to input encrypted text and receive decrypted outputs through a **command-line interface (CLI)**, with the possibility of adding a **graphical user interface (GUI)** for ease of use. The **Automation & Optimization Module** improves performance by implementing **multi-threading and heuristic-based optimizations**, making cryptanalysis more efficient for large texts. Finally, the **Reporting & Results Module** presents decrypted text along with a summary of the analysis, providing insights into vulnerabilities in the encryption. Together, these modules create a **powerful Python-based cryptanalysis tool** that automates decryption and helps users understand encryption weaknesses.

3.3 DIAGRAM



THREE TYPES OF CRYPTOGRAPHY

Cryptography uses mathematical computations (algorithms) to encrypt data, which is later decrypted by the recipient of the information.

4 Implementation of Methodology

The implementation of this project follows a structured methodology to ensure effective cryptanalysis of classical ciphers. The approach involves encryption, cryptanalysis techniques, automation, and result evaluation.

The first step is cipher implementation, where encryption and decryption functions for Caesar, substitution, and Vigenère ciphers are developed. The Caesar cipher shifts letters by a fixed amount, the substitution cipher replaces each letter with another character based on a key, and the Vigenère cipher applies a repeated keyword for shifting letters. These functions are essential for testing the cryptanalysis methods.

Next, the cryptanalysis techniques are implemented. The brute force attack is used to crack the Caesar cipher by systematically trying all possible shifts. Frequency analysis compares letter distributions in the ciphertext to common letter frequencies in the English language, aiding in breaking the substitution cipher. A dictionary attack is employed to guess plaintext by matching words from a predefined word list. For the Vigenère cipher, the Kasiski examination is applied to detect repeating patterns and estimate the key length, followed by frequency analysis to determine the individual shifts.

To enhance efficiency, an automation module is integrated, where multi-threading is used to accelerate brute force decryption. Additionally, heuristic-based optimizations refine frequency analysis results by dynamically adjusting predictions based on contextual word patterns.

The user interaction module provides a command-line interface (CLI) for inputting ciphertext and selecting the decryption technique. Optionally, a graphical user interface (GUI) can be added for a more user-friendly experience.

Finally, the results and reporting module displays the decrypted text along with an analysis report. This includes letter frequency distributions, success rates of different decryption techniques, and vulnerabilities identified in the encryption methods. By following this methodology, the program effectively demonstrates how classical ciphers can be broken using automated cryptanalysis techniques.

5 Technologies to be Used

Python Programming Language

- Python is the primary language for developing the cryptanalysis tool due to its simplicity, rich libraries, and flexibility. It allows rapid prototyping and is well-suited for implementing encryption, decryption, and cryptanalysis techniques.

Libraries and Frameworks

- NumPy: For handling arrays and performing mathematical operations like frequency analysis.
- Collections: For handling counters and frequency distributions in ciphertext.
- Threading / Multiprocessing: To implement multi-threading and speed up brute force decryption and other computationally intensive tasks.
- Tkinter (for GUI): If a graphical user interface is to be included, Tkinter will be used to build the front end, providing a simple interface for users to interact with the tool.

Cryptographic Algorithms

- Caesar Cipher Implementation: Standard shift-based encryption and decryption algorithms.
- Substitution Cipher: Letter mapping based on a key for encrypting and decrypting messages.
- Vigenère Cipher: Polyalphabetic cipher using a keyword, where the key is repeated for each letter of the plaintext.

6 Software Platform

Operating System (OS)

- The project is platform-independent and can run on Windows, Linux, or macOS.
- For testing and deployment, Ubuntu (Linux) is preferred due to its stability for Python-based applications.

Programming Language

- Python 3.x: The entire cryptanalysis tool will be implemented in Python due to its extensive libraries and ease of handling cryptographic operations.

Development Environment & Tools

- Jupyter Notebook: Useful for interactive development and testing.
- PyCharm / VS Code: Preferred IDEs for writing, debugging, and managing the Python project efficiently.
- Anaconda (Optional): A Python distribution with pre-installed libraries, useful for handling dependencies.

Libraries & Dependencies

- NumPy for mathematical computations.
- Collections for frequency analysis.
- Threading / Multiprocessing for performance optimization.
- Tkinter (Optional) for building a GUI-based user interface.

Version Control & Collaboration

- Git for version control and tracking changes.
- GitHub / GitLab for project collaboration and repository management.

Testing & Debugging Tools

- unittest (Python's built-in testing framework) for validating cryptanalysis techniques.
- PyLint for code quality checks and debugging.

Testing & Debugging Tools

- unittest (Python's built-in testing framework) for validating cryptanalysis techniques.
- PyLint for code quality checks and debugging.

Execution Environment

- The program can be executed via command-line interface (CLI) for direct interaction.
- Optionally, a GUI-based application can be developed for user-friendly interactions.

7 Hardware Platform

1. General-Purpose Hardware
For basic cryptanalysis tasks like frequency analysis, Caesar cipher decryption, or simple substitution cipher attacks, a standard personal computer or laptop is sufficient.

- Processor: Intel Core i5/i7 or AMD Ryzen 5/7
- RAM: 8GB or higher (16GB recommended for large-scale operations)
- Storage: SSD (for faster data processing)
- GPU: Not required for basic tasks

2. High-Performance Computing (HPC)
For cracking more complex ciphers (e.g., brute force attacks on Vigenère cipher or RSA), a high-performance machine or cloud-based solution is recommended.

- Processor: Intel Xeon or AMD Threadripper
- RAM: 32GB+
- GPU: NVIDIA RTX 3090/4090 or AMD equivalent (for parallel processing)
- Storage: NVMe SSD
- Cloud Services: AWS EC2 (GPU instances), Google Cloud, or Microsoft Azure

3. Embedded & IoT Devices

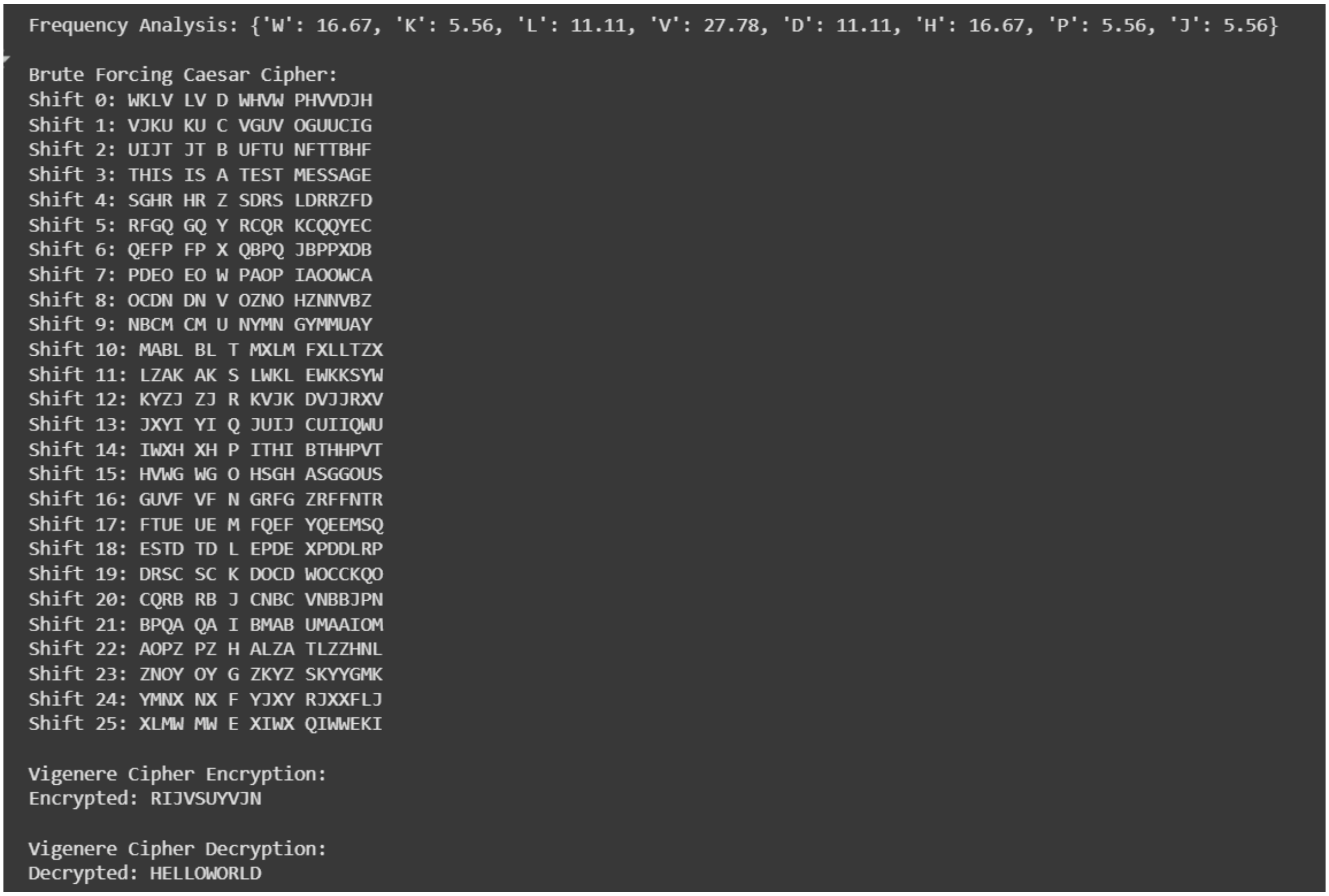
9 Screenshots, Source Code and Output

9.1 Source Code

```
import string
import collections
from itertools import permutations

# Helper function to clean text
def clean_text(text):
    return ''.join(filter(str.isalpha, text.upper()))

# Frequency analysis function
def frequency_analysis(text):
    text = clean_text(text)
    freq = collections.Counter(text)
    total_chars = sum(freq.values())
    return {char: round((count / total_chars) * 100, 2) for char, count in freq.items()}

# Brute-force attack for Caesar Cipher
def caesar_brute_force(ciphertext):
    for shift in range(26):
        decrypted = ''.join(
            chr(((ord(char) - 65 - shift) % 26) + 65) if char.isalpha() else char for char in ciphertext.upper()
        )
        print(f'Shift {shift}: {decrypted}')

# Dictionary attack for Caesar Cipher
def caesar_dictionary_attack(ciphertext, wordlist):
    for shift in range(26):
        decrypted = ''.join(
            chr(((ord(char) - 65 - shift) % 26) + 65) if char.isalpha() else char for char in ciphertext.upper()
        )
        if any(word in decrypted for word in wordlist):
            print(f'Possible match with shift {shift}: {decrypted}')

# Vigenere Cipher encryption
```

```
def vigenere_encrypt(plaintext, key):
    key = key.upper()
    encrypted = ''
    key_index = 0
    for char in plaintext.upper():
        if char.isalpha():
            shift = ord(key[key_index]) - 65
            encrypted += chr(((ord(char) - 65 + shift) % 26) + 65)
            key_index = (key_index + 1) % len(key)
        else:
            encrypted += char
    return encrypted

# Vigenere Cipher decryption
def vigenere_decrypt(ciphertext, key):
    key = key.upper()
    decrypted = ''
    key_index = 0
    for char in ciphertext.upper():
        if char.isalpha():
            shift = ord(key[key_index]) - 65
            decrypted += chr(((ord(char) - 65 - shift) % 26) + 65)
            key_index = (key_index + 1) % len(key)
        else:
            decrypted += char
    return decrypted

# Simple substitution cipher decryption using frequency analysis
def substitution_decrypt(ciphertext, mapping):
    return ''.join(mapping.get(char, char) for char in ciphertext.upper())

# Detect the cipher type based on character distribution
def detect_cipher(ciphertext):
    char_set = set(ciphertext)
    if all(char in string.ascii_uppercase for char in char_set):
        print("Likely a monoalphabetic substitution cipher")
    elif any(char.isdigit() for char in char_set):
        print("Might be a transposition cipher")
    else:
        print("Unknown cipher type")

# Example usage
if __name__ == "__main__":
    sample_text = "Wklv lv d whvw phvvdjh"  # Encrypted with Caesar Cipher shift 3
    print("Frequency Analysis:", frequency_analysis(sample_text))
    print("\nBrute Forcing Caesar Cipher:")
    caesar_brute_force(sample_text)

    print("\nVigenere Cipher Encryption:")
    encrypted_text = vigenere_encrypt("HELLOWORLD", "KEY")
    print(f'Encrypted: {encrypted_text}')

    print("\nVigenere Cipher Decryption:")
    print(f'Decrypted: {vigenere_decrypt(encrypted_text, "KEY")}')

    print("\nCipher Type Detection:")
    detect_cipher(sample_text)
```

9.2 Screenshot



```
Frequency Analysis: {'W': 16.67, 'K': 5.56, 'L': 11.11, 'V': 27.78, 'D': 11.11, 'H': 16.67, 'P': 5.56, 'J': 5.56}

Brute Forcing Caesar Cipher:
Shift 0: WKLV LV D WHVW PHVVDJH
Shift 1: VJKU KU C VGUV OGUUCIG
Shift 2: UIJT JT B UFTU NFTTBHF
Shift 3: THIS IS A TEST MESSAGE
Shift 4: SGHR HR Z SDRS LDRRZFD
Shift 5: RFGQ GQ Y RCQR KCQQYEC
Shift 6: QEFP FP X QBPQ JBPPXDB
Shift 7: PDEO EO W PAOP IAOOWCA
Shift 8: OCDN DN V OZNO HZNNVBZ
Shift 9: NBCM CM U NYMN GYMMUAY
Shift 10: MABL BL T MXLM FXLLTZX
Shift 11: LZAK AK S LWKL EWKKSYW
Shift 12: KYZJ ZJ R KVJK DVJJRXV
Shift 13: JXYI YI Q JUIJ CUIIQWU
Shift 14: IWXH XH P ITHI BTHHPVT
Shift 15: HVWG WG O HSGH ASGGOUS
Shift 16: GUVF VF N GRFG ZRFFNTR
Shift 17: FTUE UE M FQEF YQEEMSQ
Shift 18: ESTD TD L EPDE XPDDLRP
Shift 19: DRSC SC K DOCD WOCCKQO
Shift 20: CQRB RB J CNBC VNBBJPN
Shift 21: BPQA QA I BMAB UMAAIOM
Shift 22: AOPZ PZ H ALZA TLZZHNL
Shift 23: ZNOY OY G ZKYZ SKYYGMK
Shift 24: YMNX NX F YJXY RJXXFLJ
Shift 25: XLMW MW E XIWX QIWWEKI

Vigenere Cipher Encryption:
Encrypted: RIJVSUYVJN

Vigenere Cipher Decryption:
Decrypted: HELLOWORLD
```

8 Advantages of this Project

Raspberry Pi 4: Suitable for lightweight cryptanalysis tasks

NVIDIA Jetson Nano: Good for GPU-accelerated decryption

FPGA-based Hardware: Optimized for cryptographic computations

4. Specialized Hardware for Cryptanalysis

For breaking modern encryption schemes (AES, RSA, etc.), specialized hardware is used, such as:

- ASIC Miners: Used for brute-force key search
- TPM (Trusted Platform Module): Secure cryptographic operations
- Quantum Computers (e.g., IBM Q System): Experimental use in breaking encryption faster

10 Conclusion

The choice of hardware platform for performing cryptanalysis in Python depends on the complexity of the ciphers being analyzed. For basic cryptographic attacks like frequency analysis and simple substitution ciphers, a standard personal computer with a decent CPU and RAM is sufficient. However, for more advanced cryptanalysis, including brute-force attacks and modern encryption-breaking attempts, high-performance computing (HPC) setups, GPUs, and even specialized hardware like FPGAs or quantum computers may be necessary. Embedded systems such as Raspberry Pi and Jetson Nano can also be used for lightweight cryptanalysis tasks. Cloud computing platforms like AWS, Google Cloud, and Azure offer scalable solutions for handling computationally intensive operations.

11 References

- **William Stallings** - *Cryptography and Network Security: Principles and Practice* (Latest Edition)
- **Bruce Schneier** - *Applied Cryptography: Protocols, Algorithms, and Source Code in C*
- National Institute of Standards and Technology (NIST) - Cryptographic Standards
- PyCrypto and Cryptography Python Libraries - https://pypi.org/project/pycryptodome/
- GPU Computing for Cryptanalysis - NVIDIA CUDA Toolkit: https://developer.nvidia.com/cuda-toolkit
- AWS EC2 GPU Instances for Cryptographic Computation https://aws.amazon.com/ec2/instance-types/gpu/

THANKYOU VGU CSA

(A6 PROJECT CRYPTOGFRAPHY &NET ... SEC…)