

**LAB 12**

**Part 1: One-Time Pad Cryptography**

**Copyright © (2022-2025) All Rights Reserved**

**Dr. Matthew Kisow**

VERSION 2

22-DEC-2024

**LAB**:NaCL Cryptography Programming – Part 1: One-Time Pad Cryptography

**OBJECTIVE**: Create a Python script that reads the one-time pad **part1.otp.bin** and ciphertext **part1.ciphertext.bin** files, and performs decryption using XOR operations to recover the original plaintext message.

**INSTRUCTIONS**: In this lab, you will write a Python script to decrypt the ciphertext in the **part1.ciphertext.bin** file using the one-time pad stored in **part1.otp.bin**. The decryption will reconstruct the original plaintext message. Ensure you follow these guidelines when building your **part1.otp\_decrypter.py** script. Please follow the instructions for submitting this assignment on Blackboard.

1. **Prerequisites**
2. **Using the Lab Python Environment**

Navigate to the “*LAB12/Part 1*” folder and enter the Python3 virtual environment.

1. Activate the virtual environment:

source venv/bin/activate

1. **Setting Up a Home Python Environment**

Before you start programming your client, set up your Python environment:

1. Create a virtual environment:

python3 -m venv .venv

1. Activate the virtual environment:

* **Linux/macOS:**

source .venv/bin/activate

* **Windows:**

.venv\Scripts\activate

1. Install the necessary dependencies:

pip install wheel pynacl requests

1. **Understand XOR Decryption:**

Encryption and decryption with a one-time pad work as follows:

* + Encryption: **ciphertext\_byte = plaintext\_byte ^ otp\_byte**
  + Decryption: **plaintext\_byte = ciphertext\_byte ^ otp\_byte**

1. **Writing the otp\_decrypter.py Script**
2. **Import Necessary Modules**

Use the **os** and **sys** , **pynacl** modules for file handling.

1. **Define Helper Functions**
   * **Read Binary Data from File:** Create a function to read binary data from the given file:

def read\_from\_file(filename):

"""Read binary data from a file."""

with open(filename, "rb") as file:

return file.read()

* + **Decrypt Using OTP:** Implement the XOR decryption logic:

def decrypt\_with\_otp(ciphertext, otp):

"""Decrypt ciphertext using the one-time pad."""

# XOR each byte of the ciphertext with the corresponding byte of the OTP

plaintext\_bytes = bytes([c ^ o for c, o in zip(ciphertext, otp)])

return plaintext\_bytes.decode('ascii')

1. **Build the Main Script Logic**
   1. Read the **part1.ciphertext.bin** and **part1.otp.bin** files.
   2. Ensure the files have matching lengths.
   3. Decrypt the ciphertext using the OTP.
   4. Print the recovered plaintext.
2. **Testing**
   1. Save the script as **part1.py**.
   2. Run it from the command line, providing the **part1.ciphertext.bin** and **part1.otp.bin** files as arguments:

python3 part1.py part1.ciphertext.bin part1.otp.bin

**DELIVERABLE**

Write a Python3 script named **part1.py** that takes the *ciphertext* and *one-time pad* files as input, then decrypts and prints the plaintext message to the screen.

*This is the working pseudocode solution to the program.*

# Source Code File: Part 1: One-Time Pad Cryptography

# Name: part1.py

# Author: <students name>

IMPORT sys

IMPORT random FROM nacl.utils

# Define a function to reverse the one-time pad

DEFINE FUNCTION reverse\_one\_time\_pad(ciphertext\_path, otp\_path):

"""Decrypt the ciphertext using the one-time pad."""

TRY:

# Open and read the ciphertext and OTP files

OPEN ciphertext\_path AS ciphertext\_file, READ AS ciphertext

OPEN otp\_path AS otp\_file, READ AS otp

EXCEPT FileNotFoundError AS error:

PRINT "Error:", error

EXIT PROGRAM WITH CODE 1

# Check if ciphertext and OTP are of equal length

IF LENGTH OF ciphertext IS NOT EQUAL TO LENGTH OF otp:

PRINT "Error: The ciphertext and OTP files must be of the same length."

EXIT PROGRAM WITH CODE 1

# Decrypt the ciphertext using XOR

SET plaintext TO RESULT OF XORing EACH BYTE OF ciphertext WITH otp

# Try to decode and print the plaintext

TRY:

PRINT plaintext AS ASCII

EXCEPT UnicodeDecodeError:

PRINT "Error: The decoded plaintext contains non-ASCII characters."

# Main program execution

IF SCRIPT IS RUN DIRECTLY:

# Ensure two arguments (ciphertext and OTP paths) are provided

IF NUMBER OF ARGUMENTS IS NOT 3:

PRINT "Usage: python3 otp\_decrypter.py <ciphertext\_file> <otp\_file>"

EXIT PROGRAM WITH CODE 1

# Retrieve file paths from command-line arguments

SET ciphertext\_file TO ARGUMENT 1

SET otp\_file TO ARGUMENT 2

# Call the reverse\_one\_time\_pad function

CALL reverse\_one\_time\_pad(ciphertext\_file, otp\_file)