

**LAB 12**

**Part 2: Symmetric Key Cryptography**

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

**Dr. Matthew Kisow**

VERSION 2

22-DEC-2024

**LAB**:NaCL Cryptography Programming – Part 2: Symmetric Key Cryptography

**OBJECTIVE**: Create a Python script that reads the symmetric key **part2.key.bin** and ciphertext **part2.ciphertext.bin** files, and performs decryption using the Salsa20 stream cipher to recover the original plaintext message.

**INSTRUCTIONS**: This script will decrypt the ciphertext in **part2.ciphertext.bin** using the symmetric key stored in **part2.key.bin**. The decryption is based on the Salsa20 stream cipher with Poly1305 message authentication code, ensuring both confidentiality and authenticity. Follow these steps to implement the **part2.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 2*” folder and enter the Python3 virtual environment.

1. Activate the virtual environment:

source venv/bin/activate

1. **Setting Up Your 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 Salsa20 Decryption:**

* Encryption and decryption in Salsa20 use the same **SecretBox** method.
* The **decrypt** method of **SecretBox** verifies the integrity of the ciphertext before decryption, ensuring it hasn’t been tampered with.

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

Use the **os** and **sys** , **pynacl** modules for file handling, and **SecretBox** from **nacl.secret** for decryption.

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\_salsa20(ciphertext, key):

"""Decrypt ciphertext using the Salsa20 symmetric key."""

# Create a SecretBox with the given key

box = SecretBox(key)

# Decrypt the ciphertext and return plaintext

plaintext\_bytes = box.decrypt(ciphertext)

return plaintext\_bytes.decode('ascii')

1. **Build the Main Script Logic**
   1. Read the **part2.key.bin** and **part2.ciphertext.bin** files.
   2. Pass them to the decryption function.
   3. Print the recovered plaintext.
2. **Testing**
   1. Save the script as **part2.py**.
   2. Run it from the command line, providing the **part2.ciphertext.bin** and **part2.key.bin** files as arguments:

python3 part2.py part2.ciphertext.bin part2.key.bin

**DELIVERABLE**

Write a Python3 script named **part2.py** that takes the *ciphertext* and *key* 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 2: Symmetric Decryption using a Salsa20 Stream Cipher

# with Poly1305 Message Authentication Code

# Name: part2.py

# Author: <students name>

IMPORT SecretBox FROM nacl.secret

IMPORT CryptoError FROM nacl.exceptions

IMPORT sys

# Define a function for decryption

DEFINE FUNCTION decrypt\_salsa20(ciphertext\_path, key\_path):

"""Decrypt a message encrypted with Salsa20 using a symmetric key."""

TRY:

# Open and read the ciphertext and key files

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

OPEN key\_path AS key\_file, READ AS key

# Check if the key length matches the required size

IF LENGTH OF key IS NOT EQUAL TO SecretBox.KEY\_SIZE:

PRINT "Error: The key size must be exactly 32 bytes for Salsa20 encryption."

EXIT PROGRAM WITH CODE 1

# Create a SecretBox object and decrypt the ciphertext

SET box TO SecretBox(key)

SET plaintext TO RESULT OF box.decrypt(ciphertext)

RETURN plaintext AS DECODED ASCII STRING

EXCEPT FileNotFoundError AS error:

RETURN "Error: " + error

EXCEPT CryptoError:

RETURN "Decryption failed. The ciphertext or key might be corrupted."

EXCEPT UnicodeDecodeError:

RETURN "Decryption succeeded, but the plaintext is not valid ASCII."

# Main program execution

IF SCRIPT IS RUN DIRECTLY:

# Check if the correct number of arguments is provided

IF NUMBER OF ARGUMENTS IS NOT 3:

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

EXIT PROGRAM WITH CODE 1

# Retrieve file paths from command-line arguments

SET ciphertext\_file TO ARGUMENT 1

SET key\_file TO ARGUMENT 2

# Call the decryption function and display the message

SET message TO RESULT OF decrypt\_salsa20(ciphertext\_file, key\_file)

PRINT message