

CSE4057 Spring 2022

Homework 1 REPORT

Ahmet BOZBAY 150119861

Anıl Batuhan ASLAN 150119656

Yasin ÇÖREKÇİ 150119858

Introduction

In this assignment we used lots of methods and libraries. Such as base64, hashlib, tinyec, random, string etc. Our main library is pycryptodome. We used that library for most of the methods that we implement.

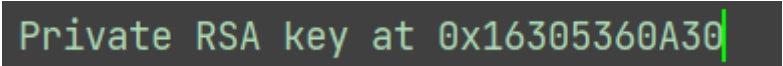
The purpose of the homework is the application of the methods covered in the course to the code. The code is written in python using version 3.9.

For generated keys, anytime we run the code, keys and outputs always will be different. So the results are for only one time run results.

1) Generation of public-private key pairs.

1.a)

Our generated key is shown in the figure[1].



figure[1].

1.b)

We assume that there are two users in this section. Then we create a public key for these two users.

After creating the public keys, we create the private keys. In order to create a public key with these keys, the private key of the first user and the public key of the second

user are multiplied. Likewise, this process is repeated for the other user. This is because two users store their private keys. If the shared keys created are equal, communication is established. This system is called ECDH.

```
Alice public key: 0x5c1ddb236891dfb3709b6a9f055ff3a8962e50d369051186578f2c470b4c1d020
Bob public key: 0x14117c13c5274d93e7288ee9239b808066e9625c49f151deb9c89cccca9785941
Now exchange the public keys (e.g. through Internet)
Alice shared key: 0x89e0bfb831c25de7c527687a1b40da82a9d1b075132b66cee73c7570a02bb5680
Bob shared key: 0x89e0bfb831c25de7c527687a1b40da82a9d1b075132b66cee73c7570a02bb5680
Equal shared keys: True
```

figure[2]: Example output.

2) Generation of Symmetric keys.

Symmetric-key algorithms are cryptographic algorithms that employ the same cryptographic keys for both plaintext encryption and ciphertext decoding. The keys could be the same, or there could be a simple change between them.

In this part, we entered the desired data. We have a text for encryption and we have a key for decryption figure[3].

```
IV_SIZE = 16 # 128 bit, fixed for the AES algorithm
KEY_SIZE = 32 # 256 bit meaning AES-256, can also be 128 or 192 bit
SALT_SIZE = 16 # This size is arbitrary

cleartext = b'This text is encrypted'
password = b'decrypt password is password'
```

figure[3].

The given message is first encrypted and then decrypted with the symmetric keys method figure[4].

```
The encrypted message is : b'\x1c\x85\xbej\xc6\x9d\x8b\x10\xde\xd2\xa1\xa8\x91\xe5\xe4\xe0\x05\xc8\xcc{\x02\x12\xe4\xc9\xe1$\x8d\xa9\xc4\xbe\x99\xee\xc9'
The decrypted message is : b'This text is encrypted'
```

figure[4].

3) Generation and Verification of Digital Signature.

Generation of Digital Signature: Hashing and public key cryptography are used to create a digital signature. You hash data and encrypt the results with your private key when you sign it. A digital signature is the encrypted hash value.

Verification of Digital Signature: The opposite of signing data is verifying a digital signature. Verifying a signature will reveal whether or not the data signed has changed. When a digital signature is confirmed, the public key is used to decrypt the signature, revealing the original hash value. The signed data has been hashed. The signature has been confirmed if the two hash values match. Make a program to accomplish this.

Our message contains a randomly generated text of 1000 characters.

Necessary signature was made on the produced text. If the signature according to the specified message is consistent with each other, the signature is valid. Otherwise, the signature is invalid.

```
msg = b'vVWReWMCvgILxtq0VrLvcwb8JcIqv6JrcDHZIXX6Y2QbZesodTLLoXuvhLUzAYLzajcNFsZKEmedIUsi6VisXeHzNCNyDwIkcmoNdFVEfse' \
b'VXcxWuP0qfSPnoGeDcm6fKPunzaLmLDbEWpspXEPa0aVYXUYstJTTMTkkCzdkVfoqEwdqKQZzhMkHhbWFFHHAnpkHPfnDslrrvr0rNInmT' \
b'kNRfLpspyfCnGojWoIDQibLRrkjDaWAjMdKvkYXsLxpTSSpaFBjXegcpJLmZUYpmyNGFADUQHRLHwxqFIpDmE0rBrF6zU0qBpQDBBrt0XZit' \
b'VbxFyVDDsAJMksyUuaAEXTeyFicpAUbotoVKNDihpkDMMKzcqis6JsKYEmYjXAAezDhasdCamPqraMxDghScNZmVDZVDXaApNAXpbfeCiBs' \
b'dRvvtFJDhSXLsNEGGrxJKTSrDimGuvFiKAMCiXrcpprPtrhYUOHLFjYpWMHCsNSxffxrnxpIXZBJwnujsdTmewicFLiCve0LbfAdrCuNPgE' \
b'QTHMxttXtynagfDaXtFiYthTxcTspcwSvutAtyxojdWkXHdiYVBjpeXvVzukkLheIBEFGwGxzuFQaYtDahoeImGPWCWmLMbNZAAIsULGNMgk' \
b'IWOURORcnXoyIAQqcXPMEzNKrmoEMPTibKwSaWMDL6dDVJcRwMpLtroGcoAQtkQrWNWTPnnCkZeWuKKmunmRJLNyaQzTYtxrHBAIudxAZZ' \
b'RcrPTPgTLsYh0QxxPxLVh0ejoMGQBLRfNhvFjvgwJohKFqCvbwaSIBPJuhuTWRqa0HHKqXSpSCUFhpktuehrBsscuH0QMZMSQRsXuyRiPS' \
b'iRYwryudbFzvCTdJELsoSilzkaduvD6GESxtEgoApTCgerknvzAzwUIeDjwuDVNjFYhNIysVUfEekTCLVQxhFtiSmdBB0zpadIzzA6zyxfu' \
b'ATTJLXraLMq6kmBVNYIOwyPknCILT6'
```

Figure[5].

```
Signature: b'as2ee7leee8127d78a4d339245ebc8d798299da887d7d796d17f0b08c745859dc171f89ae18c27af2ae333d7fd8f339a60467af778ef20bae21ab8c06af05ec20c8c8b0821825eba87fd192cd0e4c00215cedd3c346a2ade74a6daac1d2db53945a5e8f06'
Signature is invalid.
Signature is invalid.
```

```
8f06214b2e967063cc143bbdec4b757e7961486738710941ab84c3'
```

Figure[6].

4) AES Encryption.

The generated characters are encrypted with 128 and 256 bit AES CBC mode. They are encrypted by hashing with SHA256 bit encoder.

Since the output file will be very large, AES Encryption is used as 100 bits in the example. In the original code, 8000000 bits were used.

```
AES 128 bit key in CBC mode. Encrypted message is:b"V843uXWCFmqdZOT87LZ90bBHI2edYLn18n1PTHuqSU4LAYEEEE+HiH4nVCqMx5B+eLPbVLSn/dsW9Lm+jihNC5zp8I28K8nXk1JNSddEyA68rE4dTLfM3m6xxxCnBaXUwIVLVvuqu6Ln18HqsuA=="
AES 256 bit key in CBC mode. Encrypted message is:b"dPRZT2Te6JNecvVPTqW8Bvp8NSuQdLpbC39MeQ6138k+qKAwQe15gfn+WWv37/D6S8nZVqzvEsEDsy4QDCo1tdx+LNjNbHxUa+10UR1k9bcN3Zj+Y9w184daEV96ZqnTL1hT6x80Tfih1PM4P50KsyLP0jnBVT+3ir68dQ
```

Figure[7].

5) Message Authentication Codes

We don't have any code for this part.