Intro to Cybersecurity – Spring 2023
Homework/Lab #1
Due: Monday, April 10th 2023

Highlights
- Expected contribution towards the final score: 6%.
- You should work on this homework/Lab individually or in a team of up to 5 members (highly recommended). One submission per team.
- For experiments (e.g., running openssl commands), **give step-by-step screenshots**
- **Submit your work as a pdf** through the USTC Blackboard

1) **(5 points)** What are the risks of having the US government select a cryptosystem for widespread commercial use (both inside and outside the United States). How could users from outside the United States overcome some or all of these risks

   Cryptosystems are extremely complex, and it therefore can take years, or longer, to identify their vulnerabilities. Sophisticated governments may be able to promote the use of cryptosystems that those governments, and only those governments, know they can break. Given that many governments have historically shown great interest in intercepting foreign communications, using cryptosystems promoted by governments can be a risky proposition. This risk can be somewhat mitigated by adding a second layer of encryption using a different cryptosystem.

2) **(5 points)** Why do we need modes of operation for block ciphers? Also, Give a direct comparison between CBC and CTR.

   Without modes of operation, block ciphers are not IND-CPA robust.

   Regarding CBC and CTR, both are IND-CPA secure, but differ in how the initialization vector (nonce) is used during encryption. In CBC, the nonce is used to XOR the first plaintext and the XOR result is further encrypted to get the ciphertext, while for the left plaintext blocks, each of them XORs the ciphertext of the previous block. Differently, in CTR, for the ith plaintext block, $nonce + i - 1$ *is first encrypted, before XORing the ith plaintext block to get the ciphertext.*

   This encryption difference leads to two further distinctions. One is that CBC requires padding for the last block while CTR doesn't. Besides, the encryption of blocks can be concurrent in CTR while it must be sequential in CBC because the CBC encryption of one block requires the availability of ciphertext for the last block.

3) **(20 points)** Explain why hash collisions occur. That is, why must there always be two different plaintexts that have the same hash value? What property of a hash function means that collisions are not a security problem. That is, why can an attacker not capitalize on collisions and change the underlying plaintext to another form whose value collides with the hash value of the original plaintext?

   *For any given hash length (e.g., 128 bits), there will always be more possible strings in the universe *than there are possible hashes. Therefore, some pair of plaintexts must always hash to the same value.*

The property is strong collision resistance. For sufficiently strong hash functions, collisions are extremely rare and unpredictable, making the discovery of collisions computationally infeasible.

4) **(20 points)**
   a) (10 points) Identify the CAs (including the intermediate and root ones) that have issued the TLS certificate for https://ustc.edu.cn/. Then, identify the certificate expiration date. Besides, export the TLS certificate into a PEM file (e.g., through openssl), and calculate its md5 hash and SHA256 hash.
   b) (10 points) Similarly, redo the above experiments for www.12306.cn and www.bing.com

# Export the certificate

```
echo | openssl s_client -servername ustc.edu.cn -connect ustc.edu.cn:443 | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > certificate.pem

openssl x509 -noout -text  -in ./certificate.pem | head -n 10

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      af:91:49:10:fb:f1:82:95:4c:2a:4e:d4:cf:1b:19:d8
  Signature Algorithm: sha384WithRSAEncryption
    Issuer: C=AT, O=ZeroSSL, CN=ZeroSSL RSA Domain Secure Site CA
    Validity
      Not Before: Mar 21 00:00:00 2022 GMT
      Not After : Jun 19 23:59:59 2022 GMT
```

# Generate hash values

```
openssl dgst -md5 ./certificate.pem


openssl dgst -sha256 ./certificate.pem
```

5) **(30 points) Encrypt/Decrypt/Sign through openssl**
   a) Generate an AES-128 key with the cipher mode of CBC through openssl
   b) encrypt a message m = “introduction to cybersecurity 2023” and decrypt it back using the above AES-128-cbc secrets.
   c) Generate a public and private key pair
   d) generate a sha256 hash of the message m, and generate a signature by encrypting the hash with your private key
   e) Verify the digital signature, with your public key
   f) Take screenshots of step a-e, and embed them in the submission pdf.

a) Generate an AES-128 key with the cipher mode of CBC through openssl

```
openssl enc  -aes-128-cbc -pass pass:fdadfladkl -P

salt=8F9D1C45FD0519FB
```

```
key=52CB7655E7B231E56718E20DF9F5E3F9
iv =FE76A7AE19DEB9997F3D8BCF10B7A2CF
```

b) encrypt a message m and decrypt it back using the above AES-128-cbc secrets.

```
echo "helloAES" >plaintext.txt

cat plaintext.txt
helloAES

openssl enc -aes-128-cbc -K 52CB7655E7B231E56718E20DF9F5E3F9 -iv FE76A7AE19DEB9997F3D8BCF10B7A2CF -S 8F9D1C45FD0519FB -e -in ./plaintext.txt -out ciphertext.txt

 hexdump ./ciphertext.txt
0000000 84 00 ad c7 98 13 dd 68 31 ec 94 ff 2a f9 e5 60
0000010


openssl enc -aes-128-cbc -K 52CB7655E7B231E56718E20DF9F5E3F9 -iv FE76A7AE19DEB9997F3D8BCF10B7A2CF -S 8F9D1C45FD0519FB -d -in ./ciphertext.txt

helloAES
```

c) Generate a public and private key pair

Both openssl genpkey and genrsa can achieve this goal. Here, we utlize genpkey since it is designed to replace genrsa

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 -out rsa_private.pem

openssl pkey -in ./rsa_private.pem  -out rsa_public.pem
```

d) generate a sha256 hash of the message m, and generate a signature by encrypting the hash with your private key

```
cat message.txt
HelloAES

openssl dgst -sha256 ./message.txt

SHA256(./message.txt)=
63a8696b0ac22d952df4465a5e0a3591bc2eca8bed76cb057ef61a9cec1ae5ab

openssl dgst  -sign ./rsa_private.pem -out message_signature.binary -sha256 ./message.txt
```

e) Verify the digital signature, with your public key

```
openssl dgst -verify ./rsa_public.pem -signature ./message_signature.binary ./message.txt

Verified OK

echo helloworld >message_1.txt

openssl dgst -verify ./rsa_public.pem -signature ./message_signature.binary ./message_1.txt

Verification Failure
```

6) **(20 points)** Read ONE of the following papers, summarize its ideas, and give your critical reviews (e.g., pros and cons of this paper):
   a) Diffie, Whitfield, and Martin E. Hellman. "**New directions in cryptography**." In Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman
   b) ElGamal, Taher. "**A public key cryptosystem and a signature scheme based on discrete logarithms**." IEEE transactions on information theory 31, no. 4 (1985)

   Review: open question

7) **(20 points) Read the following paper, summarize its ideas, and give your critical reviews: “New Directions in Cryptography” by Whitfield Diffie and Martin Hellman”.**