CAN304 Lab 3

The ECC lib and ECDH-based AKA protocols

In this lab, you will learn how to realize the basic ECDH key agreement protocol and how to achieve authentication for ECDH.

Dependencies

- Python 3
- VM

1. The ECDH key agreement protocol

Elliptic-curve Diffie-Hellman (ECDH) is a key agreement protocol that allows two parties, each having an elliptic-curve public-private key pair, to establish a shared secret over an insecure channel. This shared secret may be directly used as a key, or to derive another key. The key, or the derived key, can then be used to encrypt subsequent communications using a symmetric-key cipher. It is a variant of the Diffie–Hellman protocol using elliptic-curve cryptography.

Initially, A and B share the same ECC parameters (E, G) where E is an elliptic curve and G is the base point. The ECDH key agreement protocol involves the following steps:

- A generates a random ECC key pair: SK_A, PK_A where $PK_A = SK_A \cdot G$
- B generates a random ECC key pair: SK_B, PK_B where $PK_B = SK_B \cdot G$
- A and B exchange their public keys through the insecure channel
- A calculates $sharedKey = SK_A \cdot PK_B$
- B calculates $sharedKey = SK_B \cdot PK_A$
- Now both A and B have the same $sharedKey = SK_A \cdot SK_B \cdot G$

We provide the protocol prototypes in this lab. Now practice with the codes. Let your computer be A and the VM be B.

Practice:

(a) check the ip address of your VM

(b) change the ip address in both ECDH-A.py and ECDH-B.py into the ip address of your VM

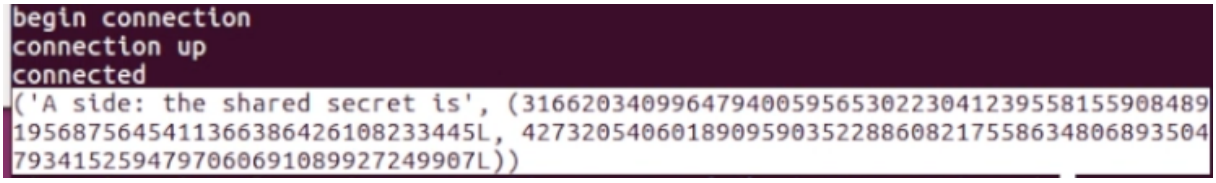
(c) run ECDH-B.py, you should see the following on the VM

```
Begin
Listen to the connection from client...
```

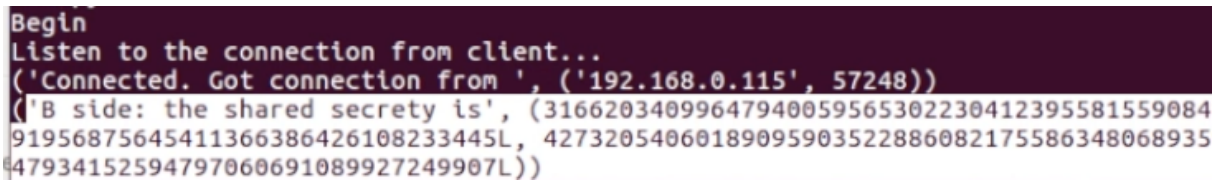
(d) run ECDH-A.py, you should see the following on your computer and VM respectively

(The value of shared secret is not the same as the following, but the values on your computer and VM should be equal.)

Computer:



VM:



Note: you can run the protocol between two VMs. Please remember to change the ip address to the ip address of the VM running ECDH-B.py

2. ECDH-based AKA protocols

The basic ECDH key agreement protocol does not involve any authentication. Thus, it is vulnerable to man-in-the-middle attack. To address this problem, authentication measure should be introduced.

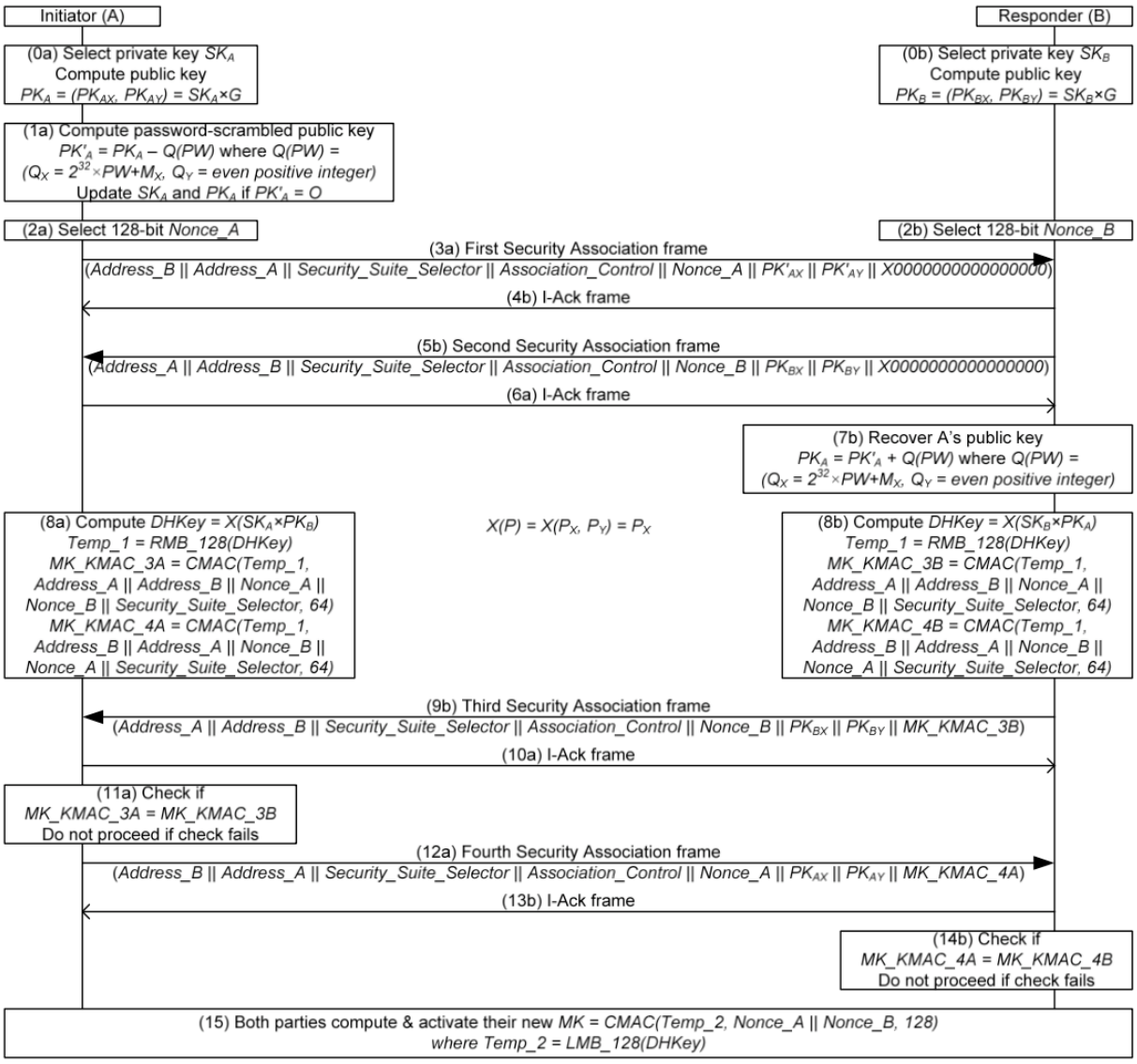
ECDH-based AKA protocols introduce authentication measures to basic ECDH key agreement to achieve authenticity. There is an increasing in popularity of ECDH-based AKA protocols as they can achieve higher security with shorter keys. Many internal standards related to communication or security are now including ECDH-based AKA protocols, for example, the IEEE 802.15.6, the Bluetooth 5.0, the TLS 1.3.

We provide prototypes of some of those ECDH-based AKA protocols. Please practice with the codes to understand the protocols.

2.1 IEEE 802.15.6

Wireless Body Area Network (WBAN) supports a variety of real-time health monitoring and consumer electronics applications. The latest international standard for WBAN is the IEEE 802.15.6 standard which aims to provide an international standard for low power, short range, and extremely reliable wireless communication within the surrounding area of the human body, supporting a vast range of data rates for different applications.

The Password Authenticated Association in the IEEE 802.15.6 is essentially an ECDH-based AKA protocol using the password authentication methods. Its core steps are shown in the following figure.



We provide a pair of codes, wbanPW-A.py and wbanPW-B.py, for this protocol.

Practice:

(a) Change the ip address in wbanPW-A.py and wbanPW-B.py

(b) Run wbanPW-B.py

(c) Run wbanPW-A.py

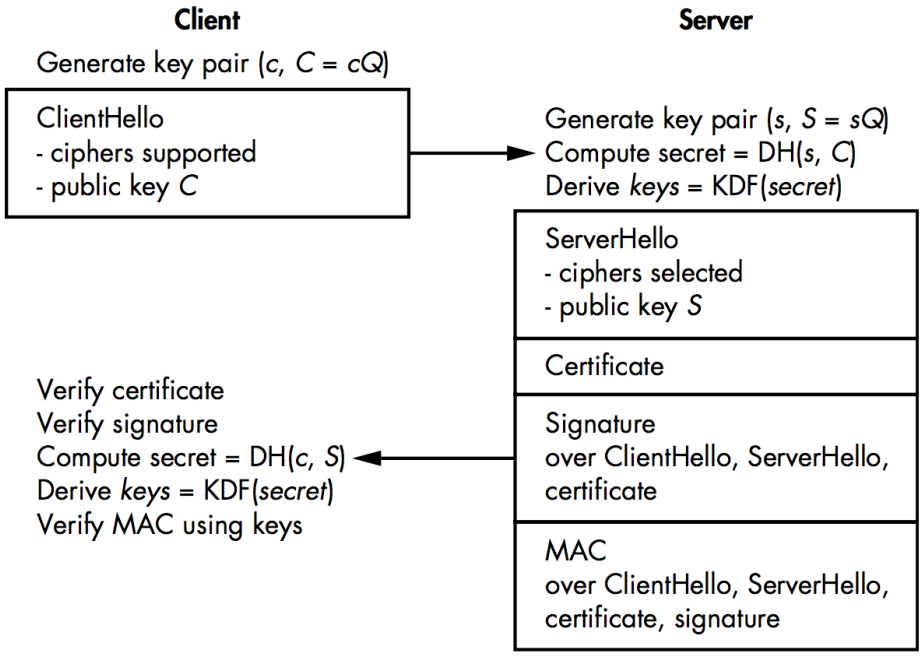
You should see the identical shared key is computed on both sides.

2.2 TLS 1.3

Transport Layer Security (TLS) standard defines cryptographic protocols to provide communications security over computer networks. It was first specified in RFC 2246 in 1999. TLS version 1.2 was released in 2008. It is currently the most widely implemented version of

TLS. In August 2018, version 1.3 of the TLS protocol was released. The new version includes a lot of privacy, security and performance improvements. With TLS 1.3, encrypted connections are much faster and more secure than before.

The elliptic curve Diffie-Hellman key ex- change (ECDHE) handshake in TLS 1.3 is essentially an ECDH-based AKA protocol using digital signature and message authentication code to realize authentication. The protocol is briefly shown in the following figure:



We provide a pair of codes, tlsserver.py and tlsclient.py, for this protocol.

Practice:

(a) Change the ip address in tlsserver.py and tlsclient.py

(b) Run tlsserver.py

(c) Run tlsclient.py

You should see the identical shared key is computed on both sides.

Homework:

The Password Authenticated Association protocol in the IEEE 802.15.6 has a vulnerability and has been broken. Programme to break and improve the Password Authenticated Association protocol.