Description of Protocol

Because the provided example protocol wban1 in IEEE package is an unauthenticated association, the public keys exchanged between two parties (A and B) cannot be guaranteed the public keys are correctly exchanged. Therefore, the protocol is vulnerable to the attack form an unauthorized third party, which is usually defined as the man-in-the-middle attack.

The proposed improvement protocol is IEEE 802.15.6 Password Authenticated Association. IEEE 802.15.6 is a globally recognized standard for Wireless Body Area Networks whose aim is to provide authentication, confidentiality and integrity. This standard intends to establish a secure communication channel between devices. Password Authenticated Association a protocol is based on Elliptic curve Diffie-Hellman key management that allows two parties having an elliptic curve public-private key pair to communicate with each other over an insecure channel with a shared secret key. The protocol is described as follows:

a.1) A side: Generate keypair PK_A and SK_A, where SK_A is chosen as random and PK_A is the public key. It is defined that $PK_A = SK_A \times G$, where G is the base point in the elliptic curve.
a.2) A side: Compute the password-scrambled public key $PK'_A = PK_A - PW$, where PW is the pre-shared password.
a.3) A side: Generate a nonce N_A.
a.4) A side: Transfer $M_1 = \{B, A, N_A, PK'_A\}$ to B side, where B is the identity of the responder, A is the identity of the initiator.

b.1) B side: Receive M_1 from A side. Obtain PK'_A and compute $PK_A = PK'_A + PW$.
b.2) B side: Compute the Diffie-Hellman shard secret key $K = SK_B \times PK_A$, where SK_B is chosen as random
b.3) B side: Generate a nonce N_B.
b.4) B side: Compute s message authentication code $mac_B = H(K_B, A, B, N_A, N_B)$.
b.5) B side: Transfer $M_2 = \{B, A, N_B, PK_B, mac_B\}$ to A side.

c.1) A side: Receive M_2 from B side. Obtain PK_B and compute $K_B = SK_A \times PK_B$
c.2) A side: Verify mac_B.
c.3) A side: Compute s message authentication code $mac_A = H(K_A, B, A, N_B, N_A)$.
c.4) A side: Transfer $M_3 = mac_A$ to B side.

d.1) B side: Receive M_3 from A side. Verify mac_B.

Experimental Platform

The platform for the coordinator and the sensor in this assignment were both operated on a laptop running 64-bit Windows10, with 16 GB RAM and a 2.6 GHz CPU. Visual Studio 2019 with Python 3 was used for demonstrating the proposed protocol.

Results

In order to explore the performance of the experimental result, multiple times of running the protocol are presented below.

```
begin connection
connection up
connected
M1 sent!
M2 received!
M3 received!
M4 sent!
macb is valid
the shared secrety is:
(76509206714174646280405871761189441664375982361506123823790970660574476575856, 110542852741378968052399745693401684504924074336967251915180394451325319198448)

Runtime: 0.134614 seconds
```

Figure 1: First test: sensor (A side)

```
Begin
Listen to the connection from client...
Connected. Got connection from ('172.31.82.78', 64403)
M1 received!
M2 sent!
M3 sent!
M4 received!
maca is valid
the shared secrety is:
(76509206714174646280405871761189441664375982361506123823790970660574476575856, 110542852741378968052399745693401684504924074336967251915180394451325319198448)

Runtime: 0.133647 seconds
```

Figure 2: First test: coordinator (B side)

```
begin connection
connection up
connected
M1 sent!
M2 received!
M3 received!
M4 sent!
macb is valid
the shared secrety is:
(54515258372314608467310087019377657514430404431380839641996156872074277879775, 70075305803421320543060693176059937140690328450537744247723342650828024898074)

Runtime: 0.136635 seconds
```

Figure 3: Second test: sensor (A side)

```
Begin
Listen to the connection from client...
Connected. Got connection from ('172.31.82.78', 64439)
M1 received!
M2 sent!
M3 sent!
M4 received!
maca is valid
the shared secrety is:
(54515258372314608467310087019377657514430404431380839641996156872074277879775, 70075305803421320543060693176059937140690328450537744247723342650828024898074)

Runtime: 0.134639 seconds
```

Figure 4: Second test: coordinator (B side)

```
begin connection
connection up
connected
M1 sent!
M2 received!
M3 received!
M4 sent!
macb is valid
the shared secrety is:
(73375281749088090117275130049338147852882569957547113238226829462616631820614, 58057793738094071780868392980065915743241964444476750640161182267510968855770)

Runtime: 0.136628 seconds
```

Figure 5: Third test: sensor (A side)

```
Begin
Listen to the connection from client...
Connected. Got connection from  ('172.31.82.78', 64467)
M1 received!
M2 sent!
M3 sent!
M4 received!
maca is valid
the shared secrety is:
(73375281749088090117275130049338147852882569957547113238226829462616631820614, 58057793738094071780868392980065915743241964444767506401611822675109688557770)

Runtime: 0.135637 seconds
```

Figure 6: Third test: coordinator (B side)

From Figure 1 to Figure 6, it can be observed that the sensor and the coordinator connected successfully and could establish a secure communication environment with sharing keys. Additionally, every party could verify successfully with each other. A list of multiple running time is summarized below.

Test	Sensor (A side)	Coordinator (B side)
1	0.134614	0.133647
2	0.136635	0.134639
2	0.135637	0.136628
Average time (s)	0.135629	0.134971

The following demonstration is the situation when the two parties failed verifying with each other. It can be seen that the protocol stopped running when the failure of verification occurred.

```
begin connection
connection up
connected
M1 sent!
M2 received!
M3 received!
M4 sent!
macb is invalid, protocol fails
```

Figure 7: Invalid verification when attacked by the unauthorized third party in sensor

```
Begin
Listen to the connection from client...
Connected. Got connection from  ('172.31.82.78', 64760)
M1 received!
M2 sent!
M3 sent!
M4 received!
maca is invalid, protocol fails
```

Figure 8: Invalid verification when attacked by the unauthorized third party in coordinator