Research in Information Security

# Project: Implementation of authentication scheme using PBC (PairingBased Cryptography) library C or JAVA (JPBC)

Group Number - 3
Manish Raushan (2021801005)
Sudipta Halder (2021202011)
Nitin Kumar (2021202020)

## Paper Summary

If e-health data gets leaked, it would cause serious threats since they are pretty sensitive and they can be tampered to capture one's personal information. This particular proposed scheme ensures security in case an adversary gets access to ephemeral secrets.

In the present times, Wireless Body Area Networks (WBANs) Technology is a very effective WAN technology for building devices that are capable of providing health services specially during emergencies.

**An authentication scheme makes sure of security in the following cases:**

- If somehow an adversary gets access of a session key or session state information, the sessions before and after this should not be affected.
- If somehow adversary gets access of long-term private keys of client or application provider, the sessions in the past should not be leaked, it is also known as perfect forward secrecy.

**The problem with traditional schemes are as follows:**

- Most of them are public-key infrastructure (PKI) based schemes.

- In this type of scheme, the user's identity is denoted by a certificate with which the corresponding public key is attached.
- It is very challenging to maintain a large number of public keys.

**In the proposed scheme, the above mentioned things have been addressed:**

- Lifetime-based pseudo-identity is used to decrease vulnerability of the particular system.
- In this scheme, the management of a large number of public keys is not necessary.

**Details of the proposed scheme:**

- Three roles -> Network Manager NM, WBAN Client C, Application Provider AP
- In the initialization phase, Network manager (NM) initializes its parameters.
- In the next phase, Client and Application Provider creates a secure connection between themselves after getting authentication credentials from Network Manager.

The steps of the proposed scheme are listed below (1. Initialization Phase, 2. Client Registration Phase, 3. AP registration Phase, 4. Connection establishment between Client and AP).

**Security characteristics of the proposed scheme:**

- If somehow an adversary gets access of a session key or session state information, the sessions before and after this should not be affected.
- If somehow adversary gets access of long-term private keys of client or application provider, the sessions in the past should not be leaked, it is also known as perfect forward secrecy.
- Credential privacy
- Anonymity
- Perfect Forward Secrecy
- Unlinkability and non-traceability
- Mutual authentication
- Resist impersonation attack

- Non-repudiation
- No key escrow

**Performance analysis of the proposed scheme:**

| Scheme | Client | Application provider |
|---|---|---|
| Liu et al. (2014) | 4TG1 + 2TGadd + 3Th 2 92.7529 ms | 1TGe + 1TG1 + 1TGadd + 1TGmul + 3Th 2 1.9777 ms |
| Zhao (2014) | 3TG1 + 4Th + 1Tg & 69.3365 ms | 6TG1 + 2TGadd + 5Th + 1Tg 2 7.6574 ms |
| Wang and Zhang (2015) | 1TGe + 3TG1 + 2TGH + 2Th + 1T 2 205.7073 ms | 1TG + 2TG1 + 2TGH + 2Th + 1T 2 8.536 ms |
| Wu et al. (2016) | 3TG1 + 2TG2 + 1TGadd + 3Th + 1T & 77.4037 ms | 1TGe + 3TG1 + 2TG2 + 2TGadd + 2Th + 1T 2 4.7286 ms |
| He et al. (2016) | 4TG1 + TGH + TGadd + 4Th + 1T & 147.1929 ms | 2TGe + 4TG1 + TGH + TGadd + 4Th + 1T 2 9.0395 ms |
| Our proposed scheme | 2TG1 + 2TG2 + 1TGadd + 2TGmul & 54.3838 ms | 1TGe + 3TG2 + 2TGmul + 4Th + 1T + 4Th + 1T 2 1.1319 ms |

**Comparison of Computational overhead**

| Scheme | Communication overhead (in bits) |
|---|---|
| Liu et al. (2014) | 3424 |
| Zhao (2014) | 3648 |
| Wang and Zhang (2015) | 3268 |
| Wu et al. (2016) | 3168 |
| He et al. (2016) | 3328 |
| Our proposed scheme | 2208 |

**Comparison of Communication overhead**

| Scheme | Client (in bits) | Application provider (in bits) |
|---|---|---|
| Liu et al. (2014) | 2112 | (160 + 1536n) |
| Zhao (2014) | (5544n + 32) | 160 |
| Wang and Zhang (2015) | 1024 | 1024 |
| Wu et al. (2016) | 1084 | 1024 |
| He et al. (2016) | 1088 | 160 |
| Our proposed scheme | 1248 | 1024 |

**Comparison of Storage overhead**

## Requirements

- Create 2 servers, consider server 1 as Client C and Server 2 as Application Provider AP.
- Only access control between Client C Application Provider AP needs to be implemented.
- File transfer using session key.

## Reference

*Odelu, V., Saha, S., Prasath, R., Sadineni, L., Conti, M., & Jo, M. (2019). Efficient privacy preserving device authentication in WBANs for industrial e-health applications. Computers & Security (Elsevier), 83, 300-312.*

# Steps of the proposed scheme

## 1. Initialization Phase

### The following parameters are initialized by Network Manager:

a. Generator P ∈ G1

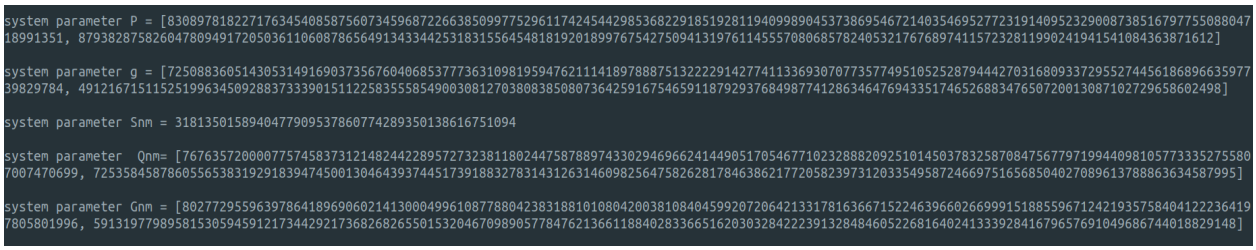
b. Generator g = e (P, P) ∈ G 2

c. Network Manager's private key is generated $S_{nm}$ <- $Z_q$*

d. Network Manager's public keys are generated i) $Q_{nm}$ <- $S_{nm}$P ∈ G1, ii) $g_{nm}$ <- $g^{Snm}$ ∈ G2

e. Refer to the below pictures for code snippet and corresponding output

```
/*********NM Parameter generation********************/
  element_init_G1(P, pairing);
  element_random(P);
  element_printf("system parameter P = %B\n", P);

  element_init_GT(g, pairing);
  element_pairing(g, P, P);
  element_printf("system parameter g = %B\n", g);

  element_init_Zr(Snm, pairing);
  element_random(Snm);
  element_printf("system parameter Snm = %B\n", Snm);

  element_init_G1(Qnm, pairing);
  element_mul_zn(Qnm,P,Snm);
  element_printf("system parameter  Qnm= %B\n", Qnm);

  element_init_GT(Gnm, pairing);
  element_pow_zn(Gnm, g, Snm);
  element_printf("system parameter Gnm = %B\n", Gnm);
```




## 2. Client Registration Phase

a. Client Choose $ID_c$ and send it to Network Manager.

b. Network Manager checks validity of $ID_c$

c. Network manager computes $r_c$ <- $Z_q$*

d. Network Manager also computes $g_c = g^{rc}$, $s_c = r_c + s_{nm}h_c$ ($h_c$ = hash($g_c$, $Right_c$, $Q_{nm}$))

e. Network Manager sends these parameters to Client and Client stores them ($Right_c$, $s_c$, $g_c$) for future reference.

f. Refer to the below pictures for code snippet and corresponding output.

```
/*******************Additional NM Parameter******************/
    element_init_Zr(Rc, pairing);
    element_random(Rc);
    element_printf("system parameter Rc = %B\n\n", Rc);

    element_init_GT(Gc, pairing);
    element_pow_zn(Gc, g, Rc);
    element_printf("system parameter Gc = %B\n\n", Gc);

    element_init_Zr(Hc, pairing);
    element_from_hash(Hc, "gc, Rightc, Qnm", 16);
    element_printf("system parameter Hc = %B\n\n", Hc);

    element_init_Zr(SnmHc, pairing);
    element_mul_zn(SnmHc,Snm,Hc);
    element_printf("system parameter SnmHc= %B\n\n", SnmHc);

    element_init_Zr(Sc, pairing);
    element_add(Sc,Rc,SnmHc);
    element_printf("system parameter Sc= %B\n\n", Sc);
```

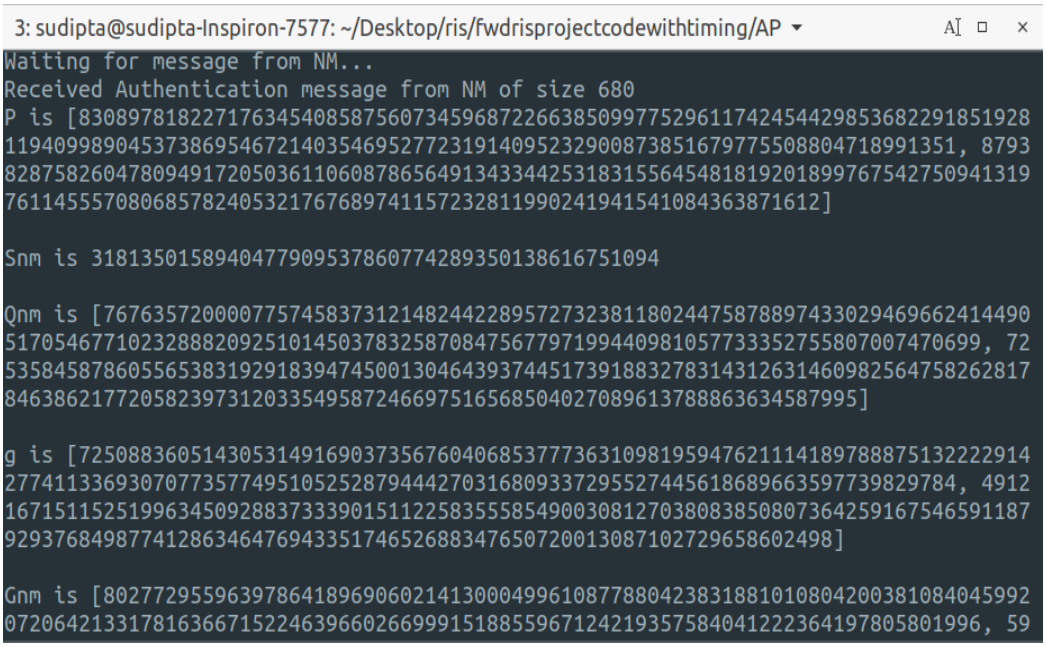
```
system parameter Rc = 730716962802279865662429872872194594894088373312

system parameter Gc = [41062073437341892248514453403258836242547467565385291091322788056733291468148725912238472412986205566390170614609490203132515487393587111323772474
39562513, 48797002382574132573959704724920730985337377970581041034201407602518252415636835213210314882246997468667668070226407747079973027314801281776859941542500867]

system parameter Hc = 590237667108670725796356627062109070680260043564

system parameter SnmHc= 193520636605864273285334413686017136957387823977

system parameter Sc= 193486780742692517586645040986706830445499637672
```

g. This is how Network manager transfers the parameters to Client and Application Provider over network..

```
element_to_bytes(P_bytes,P);
element_to_bytes(Snm_bytes,Snm);
element_to_bytes(Qnm_bytes,Qnm);
element_to_bytes(g_bytes,g);
element_to_bytes(Gnm_bytes,Gnm);
element_to_bytes(Sc_bytes,Sc);
element_to_bytes(Gc_bytes,Gc);


printf("\nSize of P is %d",element_length_in_bytes(P));
printf("\nSize of Snm is %d",element_length_in_bytes(Snm));
printf("\nSize of Qnm is %d",element_length_in_bytes(Qnm));
printf("\nSize of g is %d",element_length_in_bytes(g));
printf("\nSize of Gnm is %d",element_length_in_bytes(Gnm));
printf("\nSize of Sc is %d",element_length_in_bytes(Sc));
printf("\nSize of Gc is %d\n",element_length_in_bytes(Gc));

memcpy(buf,P_bytes,128);
memcpy(buf+128,Snm_bytes,20);
memcpy(buf+148,Qnm_bytes,128);
memcpy(buf+276,g_bytes,128);
memcpy(buf+404,Gnm_bytes,128);
memcpy(buf+532,Sc_bytes,20);
memcpy(buf+552,Gc_bytes,128);



        if (sendto(sfd, buf, 680 , 0, (struct sockaddr*) &client_address, sizeof(client_address)) == -1)
 {
   printf("Sendto failed\n");
 }
        if (sendto(sfd, buf, 680 , 0, (struct sockaddr*) &ap_address, sizeof(ap_address)) == -1)
 {
   printf("Sendto failed\n");
```

```
Size of P is 128
Size of Snm is 20
Size of Qnm is 128
Size of g is 128
Size of Gnm is 128
Size of Sc is 20
Size of Gc is 128
```

h. This is how Client and AP gets the parameters over network from NM.


## 3. Application Provider(AP) Registration Phase

a. AP chooses one $ID_{ap}$

b. AP sends $ID_{ap}$ to NM over network..

c. NM checks the validity of $ID_{ap}$ sent by AP.

d. NM calculates $K_{ap} = P/(hash(ID_{ap}) + S_{nm})$

e. NM sends this $K_{ap}$ to AP over network..

f. AP calculates $S_{ap} = hash(K_{ap} || ID_{ap})$

g. AP then makes $ID_{ap}$ public and stores the pair ($K_{ap}$, $S_{ap}$).

h. Refer to the below pictures for code snippet and corresponding output.

```
element_init_Zr(h1, pairing);
element_from_hash(h1, "ID(AP)", 7);
element_printf("system parameter h1 = %B\n\n", h1);

element_init_Zr(h1plusSnm, pairing);
element_add(h1plusSnm,h1,Snm);
element_printf("system parameter h1plusSnm = %B\n\n", h1plusSnm);

element_init_Zr(Invh1plusSnm, pairing);
element_invert(Invh1plusSnm,h1plusSnm);
element_printf("system parameter Invh1plusSnm = %B\n\n", Invh1plusSnm);

element_init_G1(Kap, pairing);
element_mul_zn(Kap,P,Invh1plusSnm);
element_printf("system parameter Kap = %B\n\n", Kap);
```

```
element_init_Zr(Sap, pairing);
element_from_hash(Sap, "Kap||ID(AP)", 12);
element_printf("system parameter Sap = %B\n", Sap);
```

```
system parameter h1 = 418276283659982611770177922534362380329835309121

system parameter h1plusSnm = 421457633818923089679715783308651730468452060215

system parameter Invh1plusSnm = 716796731548057768253613340415198032419459840160

system parameter Kap = [364953502959550735120783444936912398569123346842610664065060360068197371958760859879191353060917791880090444891038721914777327421645562944117489817285996259436041568036537700824111765965214817646688973752217116806567111814901142789110962482233136765761702437950768350776314717832913779136684728622639750728 30]
```

```
system parameter Sap = 430347279032287981647404239704611786255509506372
```

# 4. Authentication Phase between Client and AP

## a. Client C Authentication message($m_1$) generation:

i. Client computes $x_c$ <- $Z_q$*

ii. It also calculates $T_1 = x_c(hash(Id_{ap})P + Q_{nm})$.

iii. It also calculates $k_1 = g^{xc}$, $C_1 = Enc(g_c, Rights_c\ t_1)$

iv. Then it calculates $Auth_1 = hash(T_1, g_c, Right_c, t_1, k_1)$

v. Then it sends message $m_1 = (T_1, C_1, Auth_1)$ to AP over network..

vi. We also have calculated time required for performing hash operation and encryption operation and also communication delay over network.

vii. For hashing we have used **SHA-256**, and for **encryption** we have used **AES.**

```
element_to_bytes(T1_bytes,T1);
memcpy(buf,T1_bytes,element_length_in_bytes(T1));

element_to_bytes(K1_bytes,K1);
sha256(K1_bytes,128,hash);
memcpy(enc_key,hash,32);
//printf("Sizeof element_t Gc is %d\n",element_length_in_bytes(Gc));
element_printf("\nGc is %B",Gc);
element_to_bytes(Gc_bytes,Gc);
gettimeofday(&now,NULL);
prev_time=now.tv_sec*1000000+now.tv_usec;
encrypt_aes(Gc_bytes,128,enc_buf,enc_key);
gettimeofday(&now,NULL);
pres_time=now.tv_sec*1000000+now.tv_usec;
printf("\nTime for encryption (AES-256) computation time is %d microseconds\n",(pres_time-prev_time));


memcpy(buf+128,enc_buf,element_length_in_bytes(Gc));


memcpy(T1Gc,T1_bytes,128);
memcpy(T1Gc+128,Gc_bytes,128);
memcpy(T1Gc+256,K1_bytes,128);
gettimeofday(&now,NULL);
prev_time=now.tv_sec*1000000+now.tv_usec;
sha256(T1Gc,384,hash);
memcpy(buf+256,hash,32);
gettimeofday(&now,NULL);
pres_time=now.tv_sec*1000000+now.tv_usec;
printf("Time for SHA-256 computation time is %d microseconds\n",(pres_time-prev_time));

if (sendto(sfd, buf, 288, 0, (struct sockaddr*) &ap_address, sizeof(ap_address)) == -1)
        {
                printf("Sendto failed\n");
```

```
system parameter T1 = [623510651305075509532297369663451124505773356448489684998217180913873752689547647133930718316336188789857829259983318363734854218087646738468274100621846 5, 753857073828150889299663455223259353064099208238599488180909456369425071079441699474010356263341905836920513815964097034023096632343102412255712917569503 3]

system parameter K1 = [450221024906281041637959726528131165230626975776929620470535670587493929841945633201383622605606707825842208246157801702877833093656108798063884375234248 4, 864466197986427429354366397560971322736910496819904638425716917919719713902336552434117999721389242633178914687410741210773290282388841311270405846224195 2]


Gc is [410620734373418922485144534032588362425474675653852910913227880567332914681487259122384724129862055663901706146094902031325154873935871113237724743956251 3, 4879700238257413257395970472492073908533737797058104103420140760251825241563683521321031488224699746866766807022640774707997302731480128177685994154250086]

Time for encryption (AES-256) computation time is 21 microseconds
Time for SHA-256 computation time is 6 microseconds
Waiting for message from AP...

Communication delay over network is 33 microseconds
```

## b. AP Phase-2 Authentication i.e. Message m2 Generation

i. AP calculates $k_2 = Enc(T_1, K_{ap})$

ii. AP decrypts $C_1$ received from Client ($Dec_{k2}(C_1)$) and retrieves [$g_c$`, $Right_c$`, $t_1$`].

iii. Then it confirms the validity of $t_1$` and $Right_c$`. If they match, then accept or Reject.

iv. Now, AP calculates hash($T_1$, $g_c$`, $Right_c$`, $t_1$`, $k_2$). If it matches with $Auth_1$ which has been sent by Client, then accept, otherwise reject.

v. If the above step is successful, then AP generates $y_{ap}$ <- $Z_q$*.

vi. AP calculates $h_c$ = hash($g_c$, $Right_c$, $Q_{nm}$).

vii. Then, AP calculates $y_2 = y_1^{(yap + sap)}$

viii. Then, AP calculates $sk_{ap} = (y_1 * k_2)^{(yap + sap)}$

ix. Then AP calculates $Auth_2$ = hash($y_2$, $sk_{ap}$, $g_c$, $ID_c$, $T_1$, $k_2$, $t_2$).

x. Now, AP constructs a message $m_2$ = ($y_2$, $Auth_2$, $t_2$) and sends it to Client over network.

xi. Refer to the below pictures for code snippet and corresponding output.

```
/**************Phase-2 Authentication i.e. Message m2 Generation*******************/
 element_init_Zr(Yap, pairing);
 element_random(Yap);
 element_printf("system parameter Yap = %B\n", Yap);

 element_init_Zr(Sap, pairing);
 element_from_hash(Sap, "Kap||ID(AP)", 12);
 element_printf("system parameter Sap = %B\n", Sap);

 element_init_GT(Y1, pairing);
 element_pow_zn(Y1, g, Sc);
 element_printf("system parameter Y1 = %B\n", Y1);

 element_init_Zr(YapPlusSap, pairing);
 element_add(YapPlusSap,Yap,Sap);
 element_printf("system parameter YapPlusSap = %B\n", YapPlusSap);

 element_init_GT(Y2, pairing);
 element_pow_zn(Y2, Y1,YapPlusSap);
 element_printf("system parameter Y2 = %B\n", Y2);

 element_init_GT(Y1K2, pairing);
 element_mul(Y1K2, Y1,K2);
 element_printf("system parameter Y1K2 = %B\n", Y1K2);

 element_init_GT(SKap, pairing);
 element_pow_zn(SKap, Y1K2,YapPlusSap);
 element_printf("system parameter SKap = %B\n", SKap);

 element_to_bytes(Y2_bytes,Y2);
 element_to_bytes(SKap_bytes,SKap);

 memcpy(buf,Y2_bytes,128);
```

```
memcpy(Y2SKap,Y2_bytes,128);
memcpy(Y2SKap+128,SKap_bytes,128);
memcpy(Y2SKap+256,Gc_bytes,128);
memcpy(Y2SKap+384,T1_bytes,128);
memcpy(Y2SKap+512,K2_bytes,128);




/*printf("\nY2SKap is");
for(int i=0;i<640;i++)
{
  if(i%128==0)
  printf("\n");
  printf("%hhu ",Y2SKap[i]);
}*/


sha256(Y2SKap,640,hash);

memcpy(buf+128,hash,32);
gettimeofday(&now,NULL);
time2=now.tv_sec*1000000+now.tv_usec;
//printf("\ntime2 is %lu",time2);
memcpy(buf+160,(unsigned char*)&time2,8);



if (sendto(sfd, buf, 168 , 0, (struct sockaddr*) &client_address, sizeof(client_address)) == -1)
{
  printf("Sendto failed\n");
}
```

```
3: sudipta@sudipta-Inspiron-7577: ~/Desktop/ris/fwdrisprojectcodewithtiming/AP
system parameter K2 = [450221024906281041637959726528131165230626975776929620470535
670587493929841945633201383622605606707825842208246157801702877833093656108798063883
43752342484, 864466197986427429354366397560971322736910496819904638425716917919719
7139023365524341179997213892426331789146874107412107732902823888413112704058462241952]


system parameter Gc = [410620734373418922485144534032588362425474675653852910913227
880567332914681487259122384724129862055663901706146094902031325154873935871113237724
7439562513, 487970023825741325739597047249207390853373779705810410342014076025182524
156368352132103148822469974686676680702264077470799730273148012817768599415425008
6]

Authentication of message m1 verified!!!!!!
system parameter Yap = 720644737266238686591989891996708074700915811316
system parameter Sap = 430347279032287981647404239704611786255509506372
system parameter Y1 = [307329006295721546032389177143882818674688140048755003296190
559253239133124961297561690120115101817746781537572097405158769133671721857962222395
0939971344, 209379417092096797995673097660720582660114147163946744505894852778892822
444721572339369746049733741478697355468486502158819634484665162904440656498863133
2]
```

```
system parameter YapPlusSap = 42024119763307504687827488612981495955044875807 1
system parameter Y2 = [42572722134193661155651350985642492117977066612931124220807816758460806549728150629674488615925811168861935372730498068434174722611216923754046
87887593705, 70582801326050897834108640364081404554980228667225006691394536350630896384184609851356326284991212845656518528230969981817563985147370755453076949501775
5]
system parameter Y1K2 = [48813291166477597603958127307782287514503486177086796025242342583979292429767726521188395506238916140820381259844853154396251414716497396212957349778664 33, 52164989503599259815649579504218989371376163007703990063689348456297707896212521835634664658902078347839577225727299349735161025154719813541247957736143 20]
system parameter SKap = [24233170809161584875968903383639433334509955177745010331305129942737084514577811978796852730478817423823074575573620113042556987989707642139176251299427370845145778119787968527304788174238230745755736201130425569879897076421391762992450496 0, 31112575340459904197974788261585720733374029845792661274045639965788256319733461138690935258259342839166180163767666102392632518820161946707816066079746]
```

## c. Client Phase-3 Authentication i.e. Message m3 Generation

i. Client checks the validity of $t_2$ sent by AP.

ii. Client calculates $sk_c = y_2^{(xc+sc)/sc}$

iii. Then, Client calculates hash($y_2$, $s_k$, $g_c$, $ID_c$, $T_1$, $k_1$, $t_2$), if it matches with $Auth_2$ then Client accepts message $m_2$ otherwise rejects it.

iv. If the above step is true, then Client calculates Auth3 = hash($sk_c$, $k_1$, $y_2$, $t_1$, $t_2$) and sends it as $m_3 = Auth_3$ to AP.

v. Refer to the below pictures for code snippet and corresponding output.

OBJ

```
Received Authentication message from AP of size 168
system parameter Y2 = [42572722134193661155651350985642492117977066612931124220807816758460806549728150629674488615925811168861935372730498068434174722611216923754046
87887593705, 70582801326050897834108640364081404554980228667225006691394536350630896384184609851356326284991212845656518528230969981817563985147370755453076949501775
5]
system parameter XcPlusSc = 11842150392663284882042159871259099416877568736
system parameter XcPlusScDivSc = 665380712430492126796025934094538826944226332332
system parameter SKc = [24233170809161584875968903383639433334509955177745010331305129942737084514577811978796852730478817423823074575573620113042556987989707642139176
9924504960, 31112575340459904197974788261585720733374029845792661274045639965788256319733461138690935258259342839166180163767666102392632518820161946707816066079746]

Authentication of message m2 verified!!!!!
```

## d. AP Phase-4 Authentication

i. AP checks $Auth_3$ = hash($sk_{ap}$, $k_2$, $y_2$, $t_1$, $t_2$) or not. If equal, accept, otherwise reject.

ii. Refer to the below pictures for code snippet and corresponding output.

```
printf("Waiting for message from client...\n");
len=recvfrom(sfd,buf,1024,0,0,0);
printf("Received Authentication message from Client of size %d\n",len);

memcpy(SKapK2,SKap_bytes,128);
memcpy(SKapK2+128,K2_bytes,128);
memcpy(SKapK2+256,Y2_bytes,128);
sha256(SKapK2,384,hash);

if(memcmp(hash,buf,32)==0)
{
    printf("\nAuthentication of message m3 verified!!!!!!\n");
    printf("***************Authentication Successful******************\n");
}
else
{
    printf("\nAuthentication of message m3 failed????????\n");
    printf("\nXXXXXXXXXXXXXXXXAuthentication FailedXXXXXXXXXXXXXXXXXXXXX\n");
}
```

```
Waiting for message from client...
Received Authentication message from Client of size 32

Authentication of message m3 verified!!!!!!
***************Authentication Successful******************
```

## 5. Future Scope:

a. Encode and decode the messages using CBOR (Concise Binary Object Representation). Also, it helps to compress the message size which will reduce the communication overhead.