# ΑΣΦΑΛΕΙΑ ΣΤΗΝ ΤΕΧΝΟΛΟΓΙΑ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ

Εργασία 1

**ΠΑΝΤΕΛΕΗΜΩΝ ΠΡΩΙΟΣ**
ice18390023
6ο Εξάμηνο
ice18390023@uniwa.gr

Τμήμα ΑΣΦ09


ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

## Υπεύθυνοι καθηγητές

ΛΙΜΝΙΩΤΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
ΙΩΑΝΝΑ ΚΑΝΤΖΑΒΕΛΟΥ

Τμήμα Μηχανικών και Πληροφορικής Υπολογιστών
27 Μαρτίου 2021

# Περιεχόμενα



# Κώδικες

# 1 Δραστηριότητα 1: Δημιουργία ιδιωτικού κλειδιού

Με δεδομένα τα:

- $p = 953AAB9B3F23ED593FBDC690CA10E703$
- $q = C34EFC7C4C2369164E953553CDF94945$
- $e = 0D88C3$
- $n = p \cdot q$

Οπότε, για να υπολογίσουμε το ιδιωτικό κλειδί d, θα πρέπει πρώτα να υπολογίσουμε το $\phi(n)$ όπου γνωρίζουμε ότι κάνει $\phi(n) = (p-1) \cdot (q-1)$. Έπειτα, θα κάνουμε αντιστροφή αριθμητικού υπολοίπου και θα βρούμε το κλειδί d

```
63F67E805D8DEB0B4182C57C3DC24F3C1350CF182E8ABF85FD24062A3BC7F2EB
```

```c
// ex1.c
#include <stdio.h>
#include <openssl/bn.h>

void printBN(char *msg, BIGNUM * a)
{
    char * number_str = BN_bn2hex(a);
    printf("%s %s\n", msg, number_str);
    OPENSSL_free(number_str);
}

int main ()
{
    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *p1 = BN_new();
    BIGNUM *q1 = BN_new();
    BIGNUM *f = BN_new();
    BIGNUM *one = BN_new();

    // Initialize p, q, e, one
    BN_hex2bn(&p, "953AAB9B3F23ED593FBDC690CA10E703");
    BN_hex2bn(&q, "C34EFC7C4C2369164E953553CDF94945");
    BN_hex2bn(&e, "0D88C3");
    BN_dec2bn(&one, "1");
```

```
30
31    //calculate and print n = p*q
32    BN_mul(n, q, p, ctx);
33    printBN("n = ",n);
34
35    // find f
36    BN_sub(p1,p,one); // p-1
37    BN_sub(q1,q,one); // q-1
38    BN_mul(f,p1,q1,ctx); // f = (p-1)(q-1)
39    printBN("f = (p-1)(q-1) = ",f);
40
41    // calculated e * d mod f = 1 or e * d = 1 mod f
42    BN_mod_inverse(d,e,f,ctx);
43    printBN("d is ",d);
44    return 0;
45 }
```

Κώδικας 1.1: Δημιουργία ιδιωτικού κλειδιού

## 2 Δραστηριότητα 2: Κρυπτογράφηση μηνύματος

Χρησιμοποιώντας το private key που δημιουργήσαμε (ενότητα 1, σελ. 1) θα κρυπτογραφήσουμε το μήνυμα "padelis proios". Για να το κάνουμε αυτό θα πρέπει πρώτα να το μετατρέψουμε από ASCII σε δεκαεξαδική μορφή με την βοήθεια της εντολής python

```
python -c 'print("padelis proios".encode("hex"))'
```

και θα λάβουμε ως αποτέλεσμα

```
706164656c69732070726f696f73
```

Οπότε, εκχωρούμαι στην μεταβλητή P την δεκαεξαδική μορφή του μηνύματος, την κρυπτογραφούμε, έπειτα την αποκρυπτογραφούμαι και εκτυπώνουμε το αποτέλεσμα στην οθόνη με το αποτέλεσμα να είναι

```
706164656c69732070726f696f73
```

το οποίο είναι ίδιο με αυτό μου του δώσαμε, αλλά για καλύτερη ακρίβεια το μετατρέπουμε από δεκαεξαδικό σε ASCII ξανά μέσω της python με την εντολή

```
python -c 'print("706164656C69732070726F696F73".decode("hex"))'
```

όπου και λαμβάνουμε ως αποτέλεσμα το μήνυμα που κρυπτογραφήσαμε

```
padelis proios
```

```c
// ex2.c
#include <stdio.h>
#include <openssl/bn.h>

void printBN(char *msg, BIGNUM * a)
{
   char * number_str = BN_bn2hex(a);
   printf("%s %s\n", msg, number_str);
   OPENSSL_free(number_str);
}

int main ()
{
   BN_CTX *ctx = BN_CTX_new();
   BIGNUM *p = BN_new();
   BIGNUM *q = BN_new();
   BIGNUM *n = BN_new();
   BIGNUM *e = BN_new();
   BIGNUM *d = BN_new();
   BIGNUM *p1 = BN_new();
   BIGNUM *q1 = BN_new();
   BIGNUM *f = BN_new();
```

```
23    BIGNUM *one = BN_new();
24    BIGNUM *P = BN_new();
25    BIGNUM *C = BN_new();
26
27    // Initialize p, q, e, one
28    BN_hex2bn(&p, "953AAB9B3F23ED593FBDC690CA10E703");
29    BN_hex2bn(&q, "C34EFC7C4C2369164E953553CDF94945");
30    BN_hex2bn(&e, "0D88C3");
31    BN_dec2bn(&one, "1");
32
33    //calculate and print n = p*q
34    BN_mul(n, q, p, ctx);
35    printBN("n = ",n);
36
37    // find f
38    BN_sub(p1,p,one); // p-1
39    BN_sub(q1,q,one); // q-1
40    BN_mul(f,p1,q1,ctx); // f = (p-1)(q-1)
41    printBN("f = (p-1)(q-1) = ",f);
42
43    // calculate d
44    BN_mod_inverse(d,e,f,ctx);
45    printBN("d is ",d);
46
47    // padelis proios
48    BN_hex2bn(&P, "706164656c69732070726f696f73");
49
50    // C = P^e mod n
51    BN_mod_exp(C,P,e,n,ctx); // encode
52
53
54    // P = C^d mod n
55    BN_mod_exp(P,C,d,n,ctx); // decode
56    printBN("decode: ",P);
57
58    return 0;
59 }
```

Κώδικας 2.1: Κρυπτογράφηση μηνύματος

# 3 Δραστηριότητα 3: Αποκρυπτογράφηση μηνύματος

Έχουμε τα δεδομένα

- n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5
- e = 010001 (this hex value equals to decimal 65537)
- d = 74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D
- C= B3AF0A70793BB53492B5311AED5EA843D94661924C97A446E9DD75846DF860DF

και πρέπει να αποκρυπτογραφήσουμε το ciphertext C. Μπορούμε να αποκρυπτογραφήσουμε το ciphertext αν έχουμε το ιδιωτικό κλειδί (d,n) και να βρούμε το plaintext με $P = C^d \ mod \ n$. Το αποτέλεσμα του κώδικα 3.1, θα εμφανίσει στην οθόνη το μήνυμα σε δεκαεξαδική μορφή

```
49434520536563757269747920 6C616220323032302D3231
```

οπότε με την βοήθεια της python θα το αποκωδικοποιήσουμε σε ASCII και θα βρούμε το αποτέλεσμα.

```
python -c 'print("49434520536563757269747920 6C616220323032302D3231".decode("hex"))'
ICE Security lab 2020-21
```

```c
// ex3.c
#include <stdio.h>
#include <openssl/bn.h>

void printBN(char *msg, BIGNUM * a)
{
   char * number_str = BN_bn2hex(a);
   printf("%s %s\n", msg, number_str);
   OPENSSL_free(number_str);
}

int main ()
{
   BN_CTX *ctx = BN_CTX_new();
   BIGNUM *n = BN_new();
   BIGNUM *e = BN_new();
   BIGNUM *d = BN_new();
   BIGNUM *P = BN_new();
   BIGNUM *C = BN_new();

   // Initialize n, d, e, C
   BN_hex2bn(&n, "
     DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
```

```
23    BN_hex2bn(&d, "74
        D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
24    BN_hex2bn(&e, "010001");
25    BN_hex2bn(&C, "
        B3AF0A70793BB53492B5311AED5EA843D94661924C97A446E9DD75846DF860DF");
26
27
28    // P = C^d mod n
29    BN_mod_exp(P,C,d,n,ctx);
30    printBN("decode: ",P);
31
32    return 0;
33 }
```

Κώδικας 3.1: Αποκρυπτογράφηση ciphertext

# 4 Δραστηριότητα 4: Υπογραφή μηνύματος

Θα χρησιμοποιήσουμε ως δεδομένα για ιδιωτικό και δημόσιο κλειδί αυτά της ενότητας 3 (σελ. 5). Θεωρούμαι ως μήνυμα το μήνυμα "ice". Για να υπολογίσουμε την ψηφιακή του υπογραφή θα πρέπει να κάνουμε $S = P^d \ mod \ n$. Η δεκαεξαδική μορφή του "ice" είναι **696365** και η ψηφιακή του υπογραφή

```
491D279A03AF06F18586B956FF64FC311C3EC60CB1D4CAC9E4AE8693711EFF92
```

Αν όμως αλλάξουμε το μήνυμα σε "ici" όπου έχει δεκαεξαδική μορφή **696369** παρατηρούμαι πως η υπογραφή είναι τελείως διαφορετική αλλά το πλήθος παραμένει ίδιο.

```
4E0330B8D4A83FB5EF76EDAA7601071EFBF935A04CF51430FAB56EFC55D034C7
```

```c
// ex4.c
#include <stdio.h>
#include <openssl/bn.h>

void printBN(char *msg, BIGNUM * a)
{
   char * number_str = BN_bn2hex(a);
   printf("%s %s\n", msg, number_str);
   OPENSSL_free(number_str);
}

int main ()
{
   BN_CTX *ctx = BN_CTX_new();
   BIGNUM *n = BN_new();
   BIGNUM *e = BN_new();
   BIGNUM *d = BN_new();
   BIGNUM *P = BN_new();
   BIGNUM *C = BN_new();
   BIGNUM *S = BN_new();

   // Initialize n, d, e, P
   BN_hex2bn(&n, "
      DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
   BN_hex2bn(&d, "74
      D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
   BN_hex2bn(&e, "010001");
   BN_hex2bn(&P, "696365"); //ice


   // S = P^d mod n
   BN_mod_exp(S,P,d,n,ctx);
```

```
31    printf("ice\n");
32    printBN("sing: ",S);
33
34    // S = P^d mod n
35    BN_hex2bn(&P, "696369"); //ici
36    BN_mod_exp(S,P,d,n,ctx);
37    printf("\nici\n");
38    printBN("sing: ",S);
39
40    return 0;
41 }
```

Κώδικας 4.1: Υπογραφή μηνύματος

# 5 Δραστηριότητα 5: Επαλήθευση Υπογραφής

## 5.1 Περίπτωση Α

Έχουμε ως δεδομένα

- M = Launch a missile.
- S = 643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F
- e = 010001 (this hex value equals to decimal 65537)
- n = AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115

όπου το M στο δεκαεξαδικό είναι

```
4c61756e63682061206d697373696c652e
```

για να βεβαιωθούμε πως η υπογραφή είναι του Bob θα πρέπει να υπολογίσουμε το $DIGSET = S^e \ mod \ n$. Πράγματι, είναι ίδιο με το μήνυμα στην δεκαεξαδική μορφή. Έπειτα, αλλάζουμε το S στο τέλος από 2F σε 3F και ξανά υπολογίζουμε την υπογραφή. Παρατηρούμε πως είναι σχεδόν το διπλάσιο από το έγκυρο.

## 5.2 Περίπτωση Β

Έχουμε ως δεδομένα

- M = Please transfer me $2000.Alice.
- S = DB3F7CDB93483FC1E70E4EACA650E3C6505A3E5F49EA6EDF3E95E9A7C6C7A320
- e = 010001 (this hex value equals to decimal 65537)
- n = DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5

Με τον ίδιο τρόπο όπως και στην περίπτωση Α, αποδεικνύουμε πως η υπογραφή είναι valid.

```c
// ex5.c
#include <stdio.h>
#include <openssl/bn.h>

void printBN(char *msg, BIGNUM * a)
{
   char * number_str = BN_bn2hex(a);
   printf("%s %s\n", msg, number_str);
   OPENSSL_free(number_str);
}

int main ()
{
```

```
14  BN_CTX *ctx = BN_CTX_new();
15  BIGNUM *M = BN_new();
16  BIGNUM *S = BN_new();
17  BIGNUM *e = BN_new();
18  BIGNUM *n = BN_new();
19  BIGNUM *DIGSET = BN_new();
20
21  // Initialize n, S, e, M
22  BN_hex2bn(&n, "
      AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
23  BN_hex2bn(&S, "643
      D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");
24  BN_hex2bn(&e, "010001");
25  BN_hex2bn(&M, "4c61756e63682061206d697373696c652e"); //Launch a missile.
26
27
28  // DIGSET = S^e mod n
29  BN_mod_exp(DIGSET,S,e,n,ctx);
30
31  printBN("DIGSET: ",DIGSET);
32  printBN("M.....: ",M);
33
34  // Broken sign
35  BN_hex2bn(&S, "643
      D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F");
36
37  // DIGSET = S^e mod n
38  BN_mod_exp(DIGSET,S,e,n,ctx);
39
40  printf("Broken sign\n");
41  printBN("DIGSET: ",DIGSET);
42  printBN("M.....: ",M);
43
44  // Please transfer me $2000.Alice.
45  BN_hex2bn(&M, "506c65617365207472616e73666572206d652024323030302e416c6963652e");
46  BN_hex2bn(&n, "
      DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
47  BN_hex2bn(&S, "
      DB3F7CDB93483FC1E70E4EACA650E3C6505A3E5F49EA6EDF3E95E9A7C6C7A320");
48  BN_hex2bn(&e, "010001");
49
50  // DIGSET = S^e mod n
51  BN_mod_exp(DIGSET,S,e,n,ctx);
52
```

```
53    printf("bob\n");
54    printBN("DIGSET: ",DIGSET);
55    printBN("M.....: ",M);
56    printf("BN_cmp returned: %d\n", BN_cmp(DIGSET,M));
57    return 0;
58 }
```

Κώδικας 5.1: Επαλήθευση υπογραφής

# 6 Δραστηριότητα 6: Μη αυτόματη επαλήθευση πιστοποιητικού X.509

## 6.1 Βήμα 1

Με την εντολή

```
openssl s_client -connect www.uniwa.gr:443 -showcerts
```

λαμβάνουμε τα δύο πιστοποιητικά και τα τοποθετούμαι στα αρχεία c0.pem και c1.pem.

## 6.2 Βήμα 2

Εξάγουμε το modulo n με την εντολή

```
openssl x509 -in c1.pem -noout -modulus
```

και με την εντολή

```
openssl x509 -in c1.pem -text -noout
```

βρίσκουμε την γραμμή που λέει

```
Exponent: 65537 (0x10001)
```

όπου είναι το public key. Άρα, έχουμε στην κατοχή μας το (e,n).

## 6.3 Βήμα 3

Για να εξάγουμε την υπογραφή, εκτελούμε την εντολή

```
openssl x509 -in c0.pem -text -noout
```

και αντιγράφουμε το μέρος που λέει <<Signature Algorithm>>. Έπειτα, αφού το τοποθετήσουμε σε ένα αρχείο (π.χ. με το όνομα signature) για να απαλλαγούμε από τα κενά και τις άνω και κάτω τελείες, εκτελούμε την εντολή

```
cat signature | tr -d '[:space:]:'
```

## 6.4 Βήμα 4

Με την εντολή

```
openssl asn1parse -i -in c0.pem -strparse 4 -out c0_body.bin -noout
```

εξάγουμε το σώμα του πιστοποιητικού χωρίς το block υπογραφής και υπολογίζουμε το hash του με την εντολή

```
sha256sum c0_body.bin
48344c0264e91ad1a410675fcea3c598e7a8490e814cb2df9cdbe79b1d2b3e55  c0_body.bin
```

Με τον κώδικα 6.1 παρατηρούμε πως το τέλος του αποτελέσματος είναι ίδιο

```
01FF...FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF003031300D060960864801650304020105000420
48344C0264E91AD1A410675FCEA3C598E7A8490E814CB2DF9CDBE79B1D2B3E55
```

```c
// ex6.c
#include <stdio.h>
#include <openssl/bn.h>

void printBN(char *msg, BIGNUM * a)
{
   char * number_str = BN_bn2hex(a);
   printf("%s %s\n", msg, number_str);
   OPENSSL_free(number_str);
}

int main ()
{
   BN_CTX *ctx = BN_CTX_new();
   BIGNUM *S = BN_new();
   BIGNUM *e = BN_new();
   BIGNUM *n = BN_new();
   BIGNUM *DIGSET = BN_new();

   // Initialize p, q, e, one
   BN_hex2bn(&n, "C5760F0FD943293B6C6DD147ADDE10BF23C278A84A773
      5F1235BE04C1E41E7C23100BD88374575DDB90210801E8FED64230445A7A
      0393B814DCF633FC249FF229E88B0D296B95C8A741F922A2AF212C8B7685
      4B55841814068061A4F8529FBB54D3C0F4F3F40961BCEA8CC5E35FF6498F
      575DD745405A036110412245563EF94772E77F11576EED3A45945219FA8B
      ED127ED0AE8AB38CA3F87D1DAF18FB90B1F44E7E0ADF395C2164DEC84A33
      A92D4CFC67DE6BDCB1A404FB354B1F38F6F0D1EE3BE49A356E407BC8DA7C
      E1DB05B5756D1C41CFC9865D1CD462F9194BF458549F86D52871C0256012
      716AB722EF471E461B520A0FA26696A0AF1AB9F6DB7CF25");
   BN_hex2bn(&S, "7e8ecc2e0d4cfee069e686d37aff5b9669fbf348ec6bd
      255b0f111d2d09b61e39d473c877eeb539de84e966a29f5674f4031458a2
      6a43d7ae4c9c9edadd619b4389be4acada8c3541468ed6687c0825b73e37
      a721a97443e90cec7e2f02989b0f43bf5ccc5f38c5866f9e342911f25991
      45856ed7ae7407ec1b341809a2e0ba57963fcc9235ad42112f569ee9efe9
      4a14aa99b72ded690666adc37fccee679db87b473ebaaebe2253c31cca06
      c4748fcc47520055be26d50364a7d82c84ca3e1fc5f0fdf63507b5269085
      fa362e2127e411bc86feea9dafc738118aa1109e58e6099176a63e7b3434
      42755f2f9bbc584af6bf0f3c651859c982ddf45ec81b9c6");
   BN_hex2bn(&e, "10001");


```

```
42    // DIGSET = S^e mod n
43    BN_mod_exp(DIGSET,S,e,n,ctx);
44
45    printBN("DIGSET: ",DIGSET);
46    return 0;
47 }
```

Κώδικας 6.1: Επαλήθευση εγκυρότητας υπογραφής