ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΙΑ ΑΣΦΑΛΕΙΑ ΣΤΗΝ ΤΕΧΝΟΛΟΓΙΑ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ

Ονοματεπώνυμα :

Ελευθερία Τζαχρήστου

Αριθμός Μητρώου:

21390219

Ημερομηνία Παράδοσης:28/4/2024

ΠΕΡΙΕΧΟΜΕΝΑ

Δραστηριότητα 1: Δημιουργία ιδιωτικού κλειδιού

Έστω p και q δύο πρώτοι αριθμοί και e ένας κατάλληλα επιλεγμένος αριθμός ώστε να είναι σχετικά πρώτος με το Φ(N). Οι δεκαεξαδικές τιμές των p, q, και e αναφέρονται παρακάτω.

p = F7E75FDC469067FFDC4E847C51F452DF

q = E85CED54AF57E53E092113E62F436F4F

e = 0D88C3 Να γράψετε πρόγραμμα σε γλώσσα C το οποίο να υπολογίζει και να εκτυπώνει τα κάτωθι:

- Το modulo N
- Τον αριθμό Φ(N)
- Το ιδιωτικό κλειδί d

ΑΠΑΝΤΗΣΗ

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

void printBN(char *msg, BIGNUM * a)
{
   /* Use BN_bn2hex(a) for hex string
   * Use BN_bn2dec(a) for decimal string */
   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(); //ctx structure
   // Αρχικοποίηση μιας μεταβλητών BIGNUM
    BIGNUM *a = BN_new() ;
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *N = BN_new();
    BIGNUM *k = BN_new();
    BIGNUM *l = BN_new();
    BIGNUM *f = BN_new();
    BIGNUM *d = BN_new();

    // Δηλώσεις μεταβλητών
    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e, "0D88C3");
    BN_hex2bn(&a, "1");

    // Υπολογισμός res = p * q
```

```
BN_mul(N, p, q, ctx);
// Εκτύπωση αποτελέσματος
printBN("p * q  = ", N);

// Υπολογισμός k = p - 1
BN_sub(k, p, a);
// Εκτύπωση αποτελέσματος
printBN("p - a = ", k);

// Υπολογισμός l = q - 1
BN_sub(l, q, a);
// Εκτύπωση αποτελέσματος
printBN("q - a = ", l);

// Υπολογισμός res = k * l
BN_mul(f, k, l, ctx);
// Εκτύπωση αποτελέσματος
printBN("k * l  = ", f);

// Υπολογισμός αντίστροφου του e modulo f
BN_mod_inverse(d, e, f, ctx);
// Εκτύπωση αποτελέσματος
printBN("Modular Inverse of e modulo f: ", d);
```

```
    // Απελευθέρωση μνήμης BIGNUMs
    BN_CTX_free(ctx);
    BN_free(a);
    BN_free(p);
    BN_free(q);
    BN_free(e);
    BN_free(N);
    BN_free(k);
    BN_free(l);
    BN_free(f);
    BN_free(d);

    return 0;
}
```

```
[04/17/24]seed@VM:~/Downloads$ gcc -o prog1 prog1.c -lcrypto
[04/17/24]seed@VM:~/Downloads$ ./prog1
p * q  =  E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
p - a =  F7E75FDC469067FFDC4E847C51F452DE
q - a =  E85CED54AF57E53E092113E62F436F4E
k * l  =  E103ABD94892E3E74AFD724BF28E78348D52298BD687C44DEB3A81065A7981A4
Modular Inverse of e modulo f:  3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
[04/17/24]seed@VM:~/Downloads$
```

Δραστηριότητα 2: Κρυπτογράφηση μηνύματος

Θεωρήστε το μήνυμα m που αποτελείται από το ονοματεπώνυμό σας (με λατινικούς χαρακτήρες), με την παρακάτω μορφή:

Onoma Eponymo

Θα πρέπει να γράψετε πρόγραμμα σε γλώσσα C που να κρυπτογραφεί και να αποκρυπτογραφεί το μήνυμα αυτό. Το δημόσιο κλειδί (e, N) που θα χρησιμοποιήσετε θα είναι το ίδιο με αυτό που χρησιμοποιήσατε στη Δραστηριότητα 1. Περαιτέρω, με το ιδιωτικό κλειδί d που βρήκατε στη Δραστηριότητα 1, θα πρέπει να επαληθεύσετε το αποτέλεσμα της κρυπτογράφησης, δηλαδή να κάνετε αποκρυπτογράφηση. Και στις δυο περιπτώσεις θα πρέπει να εκτυπώσετε το αποτέλεσμα.

Υπόδειξη: Για να προχωρήσετε στην κρυπτογράφηση θα πρέπει πρώτα να μετατρέψετε το μήνυμά σας από συμβολοσειρά ASCII σε δεκαεξαδική (hex) συμβολοσειρά και στη συνέχεια, να μετατρέψετε την δεκαεξαδική συμβολοσειρά σε BIGNUM χρησιμοποιώντας τη συνάρτηση BN_hex2bn(). Η μετατροπή σε ASCII δεν είναι απαραίτητο να γίνει μέσα στο πρόγραμμα, αλλά μπορεί να γίνει στο terminal (και έπειτα να χρησιμοποιήσετε το αποτέλεσμα μέσα στο πρόγραμμά σας). Η ακόλουθη εντολή python μπορεί να χρησιμοποιηθεί για τη μετατροπή μιας απλής συμβολοσειράς ASCII σε μια δεκαεξαδική συμβολοσειρά.

$ python -c ’print("My Name”.encode("hex"))’

4d79204e616d65

Αντίστοιχα, για να μετατρέψτε μία δεκαεξαδική συμβολοσειρά σε μία ASCII συμβολοσειρά (αναγνώσιμη μορφή) μπορείτε να χρησιμοποιήσετε την παρακάτω εντολή python.

$ python -c ’print("4d79204e616d65".decode("hex"))’

My Name

ΑΠΑΝΤΗΣΗ

```
#include <stdio.h>

#include <openssl/bn.h>



void printBN(char *msg, BIGNUM * a)

{

   /* Use BN_bn2hex(a) for hex string

   * Use BN_bn2dec(a) for decimal string */

   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(); //ctx structure
   // Αρχικοποίηση  μεταβλητών BIGNUM
    BIGNUM *a = BN_new() ;
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *N = BN_new();
    BIGNUM *k = BN_new();
    BIGNUM *l = BN_new();
    BIGNUM *f = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *m = BN_new();
```

```
    BIGNUM *c = BN_new();

    // Δηλώσεις μεταβλητών

    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");

    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");

    BN_hex2bn(&e, "0D88C3");

    BN_hex2bn(&a, "1");

BN_hex2bn(&m,
"454c4546544845524120545a4134852495354 4f55");



    // Υπολογισμός res = p * q

    BN_mul(N, p, q, ctx);

    // Εκτύπωση αποτελέσματος

    printBN("p * q  = ", N);
```

```
// Υπολογισμός k = p - 1

BN_sub(k, p, a);

// Εκτύπωση αποτελέσματος

printBN("p - a = ", k);



// Υπολογισμός l = q - 1

BN_sub(l, q, a);

// Εκτύπωση αποτελέσματος

printBN("q - a = ", l);

BN_mul(f, k, l, ctx);

// Εκτύπωση αποτελέσματος

printBN("k * l  = ", f);



// Υπολογισμός αντίστροφου του e modulo f
```

```
    BN_mod_inverse(d, e, f, ctx);

    // Εκτύπωση αποτελέσματος

    printBN("Modular Inverse of e modulo f: ", d);

    BN_mod_exp(c, m, e, N, ctx);

    printBN("Result: ", c);

BN_mod_exp(m, c, d, N, ctx);

printBN("Result: ", m);

    // Απελευθέρωση μνήμης BIGNUMs

    BN_CTX_free(ctx);

    BN_free(a);

    BN_free(p);

    BN_free(q);

    BN_free(e);
```

```
    BN_free(N);

    BN_free(k);

    BN_free(l);

    BN_free(f);

    BN_free(d);

    BN_free(c);

BN_free(m);

    return 0;

}
```

```
[04/17/24]seed@VM:~$ python -c 'print("ELEFTHERIA TZACHRISTOU".encode("hex"))'
454c45465448455249412054 5a414348524953544f55
[04/17/24]seed@VM:~$
```

```
[04/17/24]seed@VM:~/Downloads$ gcc -o prog2 prog2.c -lcrypto
[04/17/24]seed@VM:~/Downloads$ ./prog2
p * q  =  E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
p - a =  F7E75FDC469067FFDC4E847C51F452DE
q - a =  E85CED54AF57E53E092113E62F436F4E
k * l  =  E103ABD94892E3E74AFD724BF28E78348D52298BD687C44DEB3A81065A7981A4
Modular Inverse of e modulo f:  3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
Result:  53C5450591010607D77E7D721C95AF28392CC8F7630174C3D2989D0D4F2D5290
Result:  454C45465448455249412054 5A414348524953544F55
[04/17/24]seed@VM:~/Downloads$
```

Δραστηριότητα 3: Αποκρυπτογράφηση μηνύματος

Το δημόσιο και το ιδιωτικό κλειδί που χρησιμοποιούνται σε αυτή τη δραστηριότητα είναι τα ίδια με αυτά που υπολογίσατε στη δραστηριότητα 1. Θα πρέπει αρχικά να γράψετε πρόγραμμα σε γλώσσα C που να αποκρυπτογραφεί (και να εκτυπώνει) το ακόλουθο κρυπτογράφημά c, και έπειτα να το μετατρέψτε το πίσω σε μία ASCII συμβολοσειρά σε αναγνώσιμή μορφή (είτε απευθείας μέσα στον κώδικα είτε χρησιμοποιώντας python όπως στην προηγούμενη δραστηριότητα).

c=9D368A5F8F562EB9553F6081B492C3D9527298DA5557B1895E6F7B7C8B01A308

ΑΠΑΝΤΗΣΗ

```
#include <stdio.h>

#include <openssl/bn.h>


void printBN(char *msg, BIGNUM * a)

{

   /* Use BN_bn2hex(a) for hex string

   * Use BN_bn2dec(a) for decimal string */

   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(); //ctx structure
   // Αρχικοποίηση μεταβλητών BIGNUM
    BIGNUM *a = BN_new() ;
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *N = BN_new();
    BIGNUM *k = BN_new();
    BIGNUM *l = BN_new();
    BIGNUM *f = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *m = BN_new();
    BIGNUM *c = BN_new();
```

```
    // Δηλώσεις μεταβλητών
    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e, "0D88C3");
    BN_hex2bn(&a, "1");
    BN_hex2bn(&c,
"9D368A5F8F562EB9553F6081B492C3D9527298DA5557B1895E6F7B
7C8B01A308");

    // Υπολογισμός res = p * q
    BN_mul(N, p, q, ctx);
    // Εκτύπωση αποτελέσματος
    printBN("p * q  = ", N);

    // Υπολογισμός k = p - 1
    BN_sub(k, p, a);
    // Εκτύπωση αποτελέσματος
    printBN("p - a = ", k);

    // Υπολογισμός l = q - 1
    BN_sub(l, q, a);
```

```
// Εκτύπωση αποτελέσματος
printBN("q - a = ", l);

// Υπολογισμός res = k * l
BN_mul(f, k, l, ctx);
// Εκτύπωση αποτελέσματος
printBN("k * l  = ", f);

// Υπολογισμός αντίστροφου του e modulo f
BN_mod_inverse(d, e, f, ctx);
// Εκτύπωση αποτελέσματος
printBN("Modular Inverse of e modulo f: ", d);
BN_mod_exp(m, c, d, N, ctx);
printBN("Result: ", m);
// Απελευθέρωση μνήμης BIGNUMs
BN_CTX_free(ctx);
BN_free(a);
BN_free(p);
BN_free(q);
BN_free(e);
```

```
    BN_free(N);

    BN_free(k);

    BN_free(l);

    BN_free(f);

    BN_free(d);

    BN_free(c);

    BN_free(m);

    return 0;

}
```

```
[04/17/24]seed@VM:~/Downloads$ gcc -o prog3 prog3.c -lcrypto
[04/18/24]seed@VM:~/Downloads$ ./prog3
p * q  =  E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
p - a =  F7E75FDC469067FFDC4E847C51F452DE
q - a =  E85CED54AF57E53E092113E62F436F4E
k * l  =  E103ABD94892E3E74AFD724BF28E78348D52298BD687C44DEB3A81065A7981A4
Modular Inverse of e modulo f:  3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
Result:  496E666F536563205370726 96E672053656D65737465722032303234
```

```
[04/18/24]seed@VM:~/Downloads$ python -c 'print("496E666F536563205370726 96E672053656D65737465722032303234".decode("hex"))'
InfoSec Spring Semester 2024
[04/18/24]seed@VM:~/Downloads$
```

Δραστηριότητα 4: Υπογραφή μηνύματος

Το δημόσιο και το ιδιωτικό κλειδί που χρησιμοποιούνται σε αυτή τη δραστηριότητα είναι τα ίδια με αυτά στη δραστηριότητα 3. Θεωρήστε ένα σύντομο μήνυμα m της επιλογής σας (με λατινικούς χαρακτήρες). Θα πρέπει αρχικά να γράψετε πρόγραμμα σε γλώσσα C το οποίο να παράγει και να εκτυπώνει μία ψηφιακή υπογραφή του μηνύματός σας. Μετά κάντε μία μικρή αλλαγή στο μήνυμα m(όπως να αλλάξετε ένα γράμμα ή έναν αριθμό) και υπογράψτε

ξανά το τροποποιημένο μήνυμα. Τέλος, συγκρίνετε τις δυο υπογραφές και περιγράψτε τι παρατηρείτε.

Υπόδειξη (1): Για να προχωρήσετε στην υπογραφή θα πρέπει πρώτα να μετατρέψετε το μήνυμά σας από συμβολοσειρά ASCII σε δεκαεξαδική (hex) είτε απευθείας μέσα στον κώδικα είτε χρησιμοποιώντας python.

Υπόδειξη (2): Για τις ανάγκες της άσκησης, θεωρούμε ότι η υπογραφή εφαρμόζεται απευθείας στο μήνυμα και όχι στην τιμή κατακερματισμού (hash value):

ΑΠΑΝΤΗΣΗ

```
#include <stdio.h>

#include <openssl/bn.h>


void printBN(char *msg, BIGNUM * a)

{

   /* Use BN_bn2hex(a) for hex string

   * Use BN_bn2dec(a) for decimal string */

   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(); //ctx structure
   // Αρχικοποίηση μεταβλητών BIGNUM
    BIGNUM *a = BN_new() ;
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *N = BN_new();
    BIGNUM *k = BN_new();
    BIGNUM *l = BN_new();
    BIGNUM *f = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *m = BN_new();
    BIGNUM *s = BN_new();
    // Δηλώσεις μεταβλητών
    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e, "0D88C3");
```

```
BN_hex2bn(&a, "1");

BN_hex2bn(&m, "48454c4c4f20574f524c44");


// Υπολογισμός res = p * q

BN_mul(N, p, q, ctx);

// Εκτύπωση αποτελέσματος

printBN("p * q  = ", N);


// Υπολογισμός k = p - 1

BN_sub(k, p, a);

// Εκτύπωση αποτελέσματος

printBN("p - a = ", k);


// Υπολογισμός l = q - 1

BN_sub(l, q, a);

// Εκτύπωση αποτελέσματος

printBN("q - a = ", l);


// Υπολογισμός res = k * l

BN_mul(f, k, l, ctx);
```

```
// Εκτύπωση αποτελέσματος
printBN("k * l  = ", f);

// Υπολογισμός αντίστροφου του e modulo f
BN_mod_inverse(d, e, f, ctx);
// Εκτύπωση αποτελέσματος
printBN("Modular Inverse of e modulo f: ", d);
BN_mod_exp(s, m, d, N, ctx);
printBN("Result: ", s);
// Απελευθέρωση μνήμης BIGNUMs
BN_CTX_free(ctx);
BN_free(a);
BN_free(p);
BN_free(q);
BN_free(e);
BN_free(N);
BN_free(k);
BN_free(l);
BN_free(f);
```

```
    BN_free(d);

    BN_free(s);

    BN_free(m);

    return 0;

}
```

```
[04/18/24]seed@VM:~/Downloads$ python -c 'print("HELLO WORLD".encode("hex"))'
48454c4c4f20574f524c44
[04/18/24]seed@VM:~/Downloads$ gcc -o prog4 prog4.c -lcrypto
[04/18/24]seed@VM:~/Downloads$ ./prog4
p * q  =  E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
p - a =  F7E75FDC469067FFDC4E847C51F452DE
q - a =  E85CED54AF57E53E092113E62F436F4E
k * l  =  E103ABD94892E3E74AFD724BF28E78348D52298BD687C44DEB3A81065A7981A4
Modular Inverse of e modulo f:  3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
Result:  5993D8A821EBCD5C8C7838C5732F202DCCA744E77E00BBFC9D2A910508E5A8E5
[04/18/24]seed@VM:~/Downloads$
```

Δραστηριότητα 5: Επαλήθευση Υπογραφής

Θα πρέπει αρχικά να γράψετε πρόγραμμα σε γλώσσα C το οποίο να κάνει επαλήθευση υπογραφής στις παρακάτω δύο περιπτώσεις: Περίπτωση Α: Η Alice λαμβάνει ένα μήνυμα m = “Launch a missile.” από τον Bob, μαζί με την υπογραφή του s. Γνωρίζουμε ότι το δημόσιο κλειδί του Bob είναι το (e, N). Επιβεβαιώστε ότι η υπογραφή είναι του Bob ή όχι. Το δημόσιο κλειδί του Bob και η υπογραφή (σε δεκαεξαδική μορφή) δίνονται παρακάτω:

m = Launch a missile.

s =
643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C054
2CBDB6802F

e = 010001 (this hex value equals to decimal 65537)

N =
AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B45
0F18116115

Στη συνέχεια, θεωρείστε ότι η υπογραφή έχει αλλοιωθεί (καταστραφεί), έτσι ώστε το τελευταίο byte της υπογραφής να αλλάζει από 2F σε 3F, δηλαδή, υπάρχει μόνο ένα bit που έχει αλλάξει. Επαναλάβατε τη δραστηριότητα αυτή και περιγράψτε τι θα συμβεί κατά τη διαδικασία επαλήθευσης της υπογραφής.
Περίπτωση Β: Ο Bob λαμβάνει το μήνυμα m =“Please transfer me $2000.Alice.” από την Alice, μαζί με την υπογραφή της s. Γνωρίζουμε ότι το δημόσιο κλειδί της Alice είναι το (e, N). Επιβεβαιώστε ότι η υπογραφή είναι της Alice ή όχι. Το δημόσιο κλειδί και η υπογραφή (σε δεκαεξαδική μορφή) δίνονται παρακάτω:

m = Please transfer me $2000.Alice.

s =
DB3F7CDB93483FC1E70E4EACA650E3C6505A3E5F49EA6EDF3E95E9
A7C6C7A320

e = 010001 (this hex value equals to decimal 65537)

N =
DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB816
29242FB1A5

Υπόδειξη: μπορείτε, εάν θέλετε, να χρησιμοποιήσετε και τη συνάρτηση BN_cmp για τη σύγκριση δύο αριθμών – βλ. σχετικά και το σύνδεσμο: https://www.openssl.org/docs/man1.0.2/man3/BN_cmp.html

ΑΠΑΝΤΗΣΗ

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

void printBN(char *msg, BIGNUM * a)
{
    /* Use BN_bn2hex(a) for hex string
    * Use BN_bn2dec(a) for decimal string */
    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(); //ctx structure
   // Αρχικοποίηση μεταβλητών BIGNUM
    BIGNUM *e = BN_new();
    BIGNUM *N = BN_new();
    BIGNUM *d = BN_new();
```

```
    BIGNUM *m = BN_new();

    BIGNUM *s = BN_new();

    // Δηλώσεις μεταβλητών

    BN_hex2bn(&e, "010001");

    BN_hex2bn(&N,
"AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B4
50F18116115");

    BN_hex2bn(&m, "4c61756e63682061206d697373696c652e");

    BN_hex2bn(&s,
"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C05
42CBDB6802F");

    BN_mod_exp(m, s, e, N, ctx);

printBN("Result: ", m);

    // Απελευθέρωση μνήμης BIGNUMs

    BN_CTX_free(ctx);

    BN_free(e);

    BN_free(N);

    BN_free(d);

    BN_free(s);

BN_free(m);

    return 0;
```

}

```
[04/18/24]seed@VM:~/Downloads$  python -c 'print("Launch a missile.".encode("hex"))'
4c61756e63682061206d697373696c652e
```

```
[04/19/24]seed@VM:~/Downloads$ gcc -o prog5 prog5.c -lcrypto
[04/19/24]seed@VM:~/Downloads$ ./prog5
Result:  4C61756E63682061206D697373696C652E
[04/19/24]seed@VM:~/Downloads$
```

ΑΛΛΑΓΗ του s

BN_hex2bn(&s,
"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C05
42CBDB6803F");

```
[04/19/24]seed@VM:~/Downloads$ gcc -o prog5 prog5.c -lcrypto
[04/19/24]seed@VM:~/Downloads$ ./prog5
Result:  91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294
```

2Η ΠΕΡΙΠΤΩΣΗ

```
#include <stdio.h>

#include <openssl/bn.h>


void printBN(char *msg, BIGNUM * a)

{
```

```
    /* Use BN_bn2hex(a) for hex string
    * Use BN_bn2dec(a) for decimal string */
    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(); //ctx structure
   // Αρχικοποίηση μιας μεταβλητών BIGNUM
    BIGNUM *e = BN_new();
    BIGNUM *N = BN_new();
    BIGNUM *m = BN_new();
    BIGNUM *mu = BN_new();
    BIGNUM *s = BN_new();
    // Δηλώσεις μεταβλητών
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&N,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81
```

```
629242FB1A5");

    BN_hex2bn(&m,
"506c65617365207472616e73666572206d652024323030302e416c6
963652e");

    BN_hex2bn(&s,
"DB3F7CDB93483FC1E70E4EACA650E3C6505A3E5F49EA6EDF3E95E9
A7C6C7A320");

    BN_mod_exp(mu, s, e, N, ctx);

    printBN("Result: ", mu);

    if(BN_cmp(mu,m)==0)

        printf("YES\n");
    else
        printf("No\n");


    // Απελευθέρωση μνήμης BIGNUMs

    BN_CTX_free(ctx);

    BN_free(e);

    BN_free(N);

    BN_free(s);

    BN_free(m);

    BN_free(mu);

    return 0;

}
```

```
[04/19/24]seed@VM:~/Downloads$  python -c 'print("Please transfer me $2000.Alice.".encode("hex"))'
506c65617365207472616e73666572206d652024323030302e416c6963652e
```

```
[04/19/24]seed@VM:~/Downloads$ gcc -o prog6 prog6.c -lcrypto
[04/19/24]seed@VM:~/Downloads$ ./prog6
Result:  506C65617365207472616E73666572206D652024323030302E416C6963652E
YES
[04/19/24]seed@VM:~/Downloads$
```

Δραστηριότητα 6: Μη αυτόματη επαλήθευση πιστοποιητικού X.509

```
[04/20/24]seed@VM:~/Desktop$ openssl x509 -in c2.pem -noout -modulus
Modulus=BB021528CCF6A094D30F12EC8D5592C3F882F199A67A4288A75D26AAB52BB9C54CB1AF8E6BF975C8A3D70F4794145535578C9EA8A23919F5823C42A94E6EF53BC32ED
B8DC0B05CF35938E7EDCF69F05A0B1BBEC094242587FA3771B313E71CACE19BEFDBE43B45524596A9C153CE34C852EEB5AEED8FDE6070E2A554ABB66D0E97A540346B2BD3BC66
EB66347CFA6B8B8F572999F830175DBA726FFB81C5ADD286583D17C7E709BBF12BF786DCC1DA715DD446E3CCAD25C188BC606775666B3F118F7A25CE653FF3A88B647A5FF1318E
A9809773F9D53F9CF01E5F5A6701714AF63A4FF99B3939DDC53A706FE48851DA169AE2575BB13CC5203F5ED51A18BDB15
```

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

```
24]seed@VM:~/Desktop$  cat signature | tr -d '[:space:]:'
SignatureAlgorithmsha256WithRSAEncryption4dad100559d48633f99beba5109aa245de5d00ed3594f44c9e6155135456c229182758e36ece783297f2dde672aa55c51c7
ff8c9a8eed354ed16e3246557d98d253d3d4a9a520afcccd61592dcdeda4a2b5c326c09803d5bfeda378d0e57ee770aa9c9dd4ff661bb97be08ac143c947ccd95f5bcc81edc1
c26cb83e0d1ff825bbbcd45fe14909d1d6239e2897bcb8a8c28a7c43905aae2df78b08736219685e03b06c1147c46fc68c78d01aa356a8f87a09e18953efd5f0522f8e0f38ac
6e8ad5f89e4e2e34d6fb52cda9d16a58fe1e855b71e5b550d38b0d6e89cfeab162faabf1973ce409fb7fa337e24c0395116ff778ad026de3ddf7ee76397bbbc4f140d[04/20/
```

```
20/24]seed@VM:~/Desktop$ openssl asn1parse -i -in c1.pem
    0:d=0  hl=4 l=1269 cons: SEQUENCE
    4:d=1  hl=4 l= 989 cons:  SEQUENCE
    8:d=2  hl=2 l=   3 cons:   cont [ 0 ]
   10:d=3  hl=2 l=   1 prim:    INTEGER           :02
   13:d=2  hl=2 l=  18 prim:   INTEGER           :03C92EFE401EDC4ABE66231CDC0A5AD9B062
   33:d=2  hl=2 l=  13 cons:   SEQUENCE
   35:d=3  hl=2 l=   9 prim:    OBJECT            :sha256WithRSAEncryption
   46:d=3  hl=2 l=   0 prim:    NULL
   48:d=2  hl=2 l=  50 cons:   SEQUENCE
   50:d=3  hl=2 l=  11 cons:    SET
   52:d=4  hl=2 l=   9 cons:     SEQUENCE
   54:d=5  hl=2 l=   3 prim:      OBJECT            :countryName
   59:d=5  hl=2 l=   2 prim:      PRINTABLESTRING   :US
   63:d=3  hl=2 l=  22 cons:    SET
   65:d=4  hl=2 l=  20 cons:     SEQUENCE
   67:d=5  hl=2 l=   3 prim:      OBJECT            :organizationName
   72:d=5  hl=2 l=  13 prim:      PRINTABLESTRING   :Let's Encrypt
   87:d=3  hl=2 l=  11 cons:    SET
   89:d=4  hl=2 l=   9 cons:     SEQUENCE
   91:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
   96:d=5  hl=2 l=   2 prim:      PRINTABLESTRING   :R3
  100:d=2  hl=2 l=  30 cons:   SEQUENCE
  102:d=3  hl=2 l=  13 prim:    UTCTIME           :240403100019Z
  117:d=3  hl=2 l=  13 prim:    UTCTIME           :240702100018Z
  132:d=2  hl=2 l=  22 cons:   SEQUENCE
  134:d=3  hl=2 l=  20 cons:    SET
  136:d=4  hl=2 l=  18 cons:     SEQUENCE
  138:d=5  hl=2 l=   3 prim:      OBJECT            :commonName
  143:d=5  hl=2 l=  11 prim:      PRINTABLESTRING   :panteion.gr
  156:d=2  hl=4 l= 290 cons:   SEQUENCE
  160:d=3  hl=2 l=  13 cons:    SEQUENCE
  162:d=4  hl=2 l=   9 prim:     OBJECT            :rsaEncryption
  173:d=4  hl=2 l=   0 prim:     NULL
  175:d=3  hl=4 l= 271 prim:    BIT STRING
  450:d=2  hl=4 l= 543 cons:   cont [ 3 ]
```

...

```
06082B0601050507300286166874747 03A2F2F72332E692E6C656E63722E6F72672F
  670:d=4  hl=2 l=  39 cons:     SEQUENCE
  672:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Subject Alternative Name
  677:d=5  hl=2 l=  32 prim:      OCTET STRING      [HEX DUMP]:301E820B70616E7465696F6E2E6772820F7777772E70616E7465696F6E2E6772
  711:d=4  hl=2 l=  19 cons:     SEQUENCE
  713:d=5  hl=2 l=   3 prim:      OBJECT            :X509v3 Certificate Policies
  718:d=5  hl=2 l=  12 prim:      OCTET STRING      [HEX DUMP]:300A3008060667810C010201
  732:d=4  hl=4 l= 261 cons:     SEQUENCE
  736:d=5  hl=2 l=  10 prim:      OBJECT            :CT Precertificate SCTs
  748:d=5  hl=3 l= 246 prim:      OCTET STRING      [HEX DUMP]:0481F300F1007700 1998107109F0D6522E3080D29E3F64BB836E28CCF90F528EEEDFCE4A3F16B4
CA0000018EA39DEC2400000403004830460221 0097577AA1DB6B7F29B455E16A95DBEC4C401972D7E61C72A9F487078912BCA410022100A598A7E1234C43E8AE4F9A7036D13EE
9136F13542980C731FA36C95A4C434B5F00760048B0E36BDAA647340FE56A02FA9D30EB1C5201CB56DD2C81D9BBBFAB39D884730000018EA39DEDF7000004030047304502210 0
889DA92819E58E4FA6B9E3AAB453867701334EB5B6DA050CC1ED2D1539A49F2902204D78136DA35BF5267C4564D3177D889BA09E97339B5E22FBE5CD2D8F6B2D384C
  997:d=1  hl=2 l=  13 cons:  SEQUENCE
  999:d=2  hl=2 l=   9 prim:   OBJECT            :sha256WithRSAEncryption
 1010:d=2  hl=2 l=   0 prim:   NULL
 1012:d=1  hl=4 l= 257 prim:  BIT STRING
```

```
[04/20/24]seed@VM:~/Desktop$ openssl asn1parse -i -in c1.pem -strparse 4 -out c1_body.bin -noout
[04/20/24]seed@VM:~/Desktop$ sha256sum c1_body.bin
866a6ba5cc5c3bdeb013b36bc4e4d8675b85618b145154f1fb477c91295f04e5  c1_body.bin
```

```
#include <openssl/bn.h>

#include <stdio.h>
```

```
void printBN(char *msg, BIGNUM * a)
{
   /* Χρησιμοποιήστε τη BN_bn2hex(a) για μια συμβολοσειρά εξαδικού
   * Χρησιμοποιήστε τη BN_bn2dec(a) για μια δεκαδική συμβολοσειρά */
   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(); // Δημιουργία μιας δομής περιβάλλοντος (context) BN_CTX

   // Δημιουργία των BIGNUMs
   BIGNUM *e = BN_new();
   BIGNUM *N = BN_new();
   BIGNUM *m = BN_new();
   BIGNUM *s = BN_new();

   // Ορισμός των μεταβλητών με τιμές
   BN_hex2bn(&e, "010001");
   BN_hex2bn(&N,
"BB021528CCF6A094D30F12EC8D5592C3F882F199A67A4288A75D2
```

```
6AAB52BB9C54CB1AF8E6BF975C8A3D70F4794145535578C9EA8A23
919F5823C42A94E6EF53BC32EDB8DC0B05CF35938E7EDCF69F05A0
B1BBEC094242587FA3771B313E71CACE19BEFDBE43B45524596A9C
153CE34C852EEB5AEED8FDE6070E2A554ABB66D0E97A540346B2BD
3BC66EB66347CFA6B8B8F572999F830175DBA726FFB81C5ADD2865
83D17C7E709BBF12BF786DCC1DA715DD446E3CCAD25C188BC6067
7566B3F118F7A25CE653FF3A88B647A5FF1318EA9809773F9D53F9C
F01E5F5A6701714AF63A4FF99B3939DDC53A706FE48851DA169AE2
575BB13CC5203F5ED51A18BDB15");

    BN_hex2bn(&s,
"4dad100559d48633f99beba5109aa245de5d00ed3594f44c9e61551
35456c229182758e36ece783297f2dde672aa55c51c7ff8c9a8eed354
ed16e3246557d98d253d3d4a9a520afcccd61592dcdeda4a2b5c326c0
9803d5bfeda378d0e57ee770aa9c9dd4ff661bb97be08ac143c947ccd
95f5bcc81edc1c26cb83e0d1ff825bbbcd45fe14909d1d6239e2897bcb
8a8c28a7c43905aae2df78b08736219685e03b06c1147c46fc68c78d0
1aa356a8f87a09e18953efd5f0522f8e0f38ac6e8ad5f89e4e2e34d6fb
52cda9d16a58fe1e855b71e5b550d38b0d6e89cfeab162faabf1973ce
409fb7fa337e24c0395116ff778ad026de3ddf7ee76397bbbc4f140d");

    BN_hex2bn(&m,
"866a6ba5cc5c3bdeb013b36bc4e4d8675b85618b145154f1fb477c91
295f04e5");


    // Νέο BIGNUM για την αποκρυπτογράφηση της υπογραφής

    BIGNUM *decrypted_S = BN_new();

    BN_mod_exp(decrypted_S, s, e, N, ctx);


    // Εκτύπωση της αποκρυπτογραφημένης υπογραφής και του
κατακερματισμού του μηνύματος

    printBN("Decrypted signature =", decrypted_S);
```

```
    printBN("Hash of the message =", m);

    // Έλεγχος εάν η αποκρυπτογραφημένη υπογραφή ταιριάζει με τον κατακερματισμό του μηνύματος
    if(BN_cmp(decrypted_S, m) == 0)
        printf("Signature Valid\n");
    else
        printf("Signature Invalid\n");

    // Απελευθέρωση μνήμης BIGNUMs
    BN_CTX_free(ctx);
    BN_free(e);
    BN_free(N);
    BN_free(s);
    BN_free(decrypted_S);
    BN_free(m);

    return 0;
}
```

```
[04/20/24]seed@VM:~/Downloads$ gcc -o prog6 prog6.c -lcrypto
[04/20/24]seed@VM:~/Downloads$ ./prog6
Decrypted signature = 01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFF003031300D060960864801650304020105000420866A6BA5CC5C3BDEB013B36BC4E4D8675B85618B145154F1FB477C91295F04E5
Hash of the message = 866A6BA5CC5C3BDEB013B36BC4E4D8675B85618B145154F1FB477C91295F04E5
Signature Invalid
```