

## מטלה 5

מגיש: פלג זבורובסקי

2.1.1

א. ניצור טקסט חדש

```
[01/05/19]seed@VM:~/me$ touch bubble.txt
[01/05/19]seed@VM:~/me$
```

ב. נמלא את המסמך בתוכן רנדומלי של מילים

```
[01/05/19]seed@VM:~/me$ nano bubble.txt
```

ג. על מנת להמיר את האותיות הגדולות בטקסט לאותיות קטנות אנו נרשום:

```
Tr -cd '[a-z] [\n] [:space:]' <lowercase.txt > matala.txt
```

```
[01/05/19]seed@VM:~/.../matala$ touch matala.txt
[01/05/19]seed@VM:~/.../matala$ gedit matala.txt
[01/05/19]seed@VM:~/.../matala$ ls
matala.txt
[01/05/19]seed@VM:~/.../matala$ tr [:upper:] [:lower:] matala.txt
> lowerMatala.txt
tr: extra operand 'matala.txt'
Try 'tr --help' for more information.
[01/05/19]seed@VM:~/.../matala$ tr [:upper:] [:lower:] < matala.tx
t > lowerMatala.txt
[01/05/19]seed@VM:~/.../matala$ tr -cd '[a-z][\n][:space:]' < lowe
rMatala.txt.txt > plaintext.txt
bash: lowerMatala.txt.txt: No such file or directory
[01/05/19]seed@VM:~/.../matala$ tr -cd '[a-z][\n][:space:]' < lowe
rMatala.txt > plaintext.txt
```

ד. נרשום "python" על מנת שיפתח מסמך שדרכו נוכל להריץ איזה קוד שנרצה:  
(נזכור שבכל סוף שורה נרשום (shift + enter))

```
[01/05/19]seed@VM:~/me$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for
more information.
>>>
```

ה. נרשום `import random` : (ככה אנו בעצם מייבאים את הספרייה random)

```
>>> import random
```

ו. נרשום `s="abcdrfghijklmnopqrstuvwxyz"`

```
>>> s="abcdefghijklmnopqrstuvwxyz"
```

ז. נרשום: `list=random.sample(s,len(s))`

```
>>> list = random.sample(s,len(s))
```

ח. נרשום `join(list)` באמצעות כתיבת הקודים הללו אנחנו יצרנו את מפתח ההצפנה שלנו.

```
>>> ''.join(list)
'uzipyvakqhtonxwcjefglbsrmd'
```

ט. נצא מהפייטון באמצעות כתיבת ctrl+z (ככה בעצם אנחנו יוצאים מכל תוכנית)

י. כעת נרשום:

```
tr 'abcdefghijklmnopqrstuvwxyz' 'uzipyvakqhtonxwcjefglbsrmd' / < raffaello.txt >  
ciphertext.txt
```

```
[01/05/19]seed@VM:~/me$ tr 'abcdefghijklmnopqrstuvwxyz'  
ipyvakqhtonxwcjefglbsrmd' \  
> < raffaello.txt > ciphertext.txt
```

```
Ingredients list for Raffaello  
- Vegetable fats  
- Desiccated coconut  
- Sugar  
- Skimmed milk powder  
- Almond  
- Wheat flour  
- Sweet whey powder  
- Natural flavor  
- Emulsifiers, lecithin (soy)  
- Raising agent sodium  
- Salt  
- Vanillin
```

מה שמעל זה הלפני, ומה שמתחת זה האחרי

```
Ixaeyppqyxgf oqfg vwe Ruvvuyooow  
- Vyayguzoy vugf  
- Dyfqliugyp iwiwlg  
- Slaue  
- Stqnnyp nqot cwspye  
- Aonwxp  
- Wkyug vowe  
- Ssyg skym cwspye  
- Nugleuo voubwe  
- Enlofqvqyef, oyiqqgqx (fwm)  
- Ruqfqxa uayxg fwpqln  
- Suog  
- Vuxqooqx
```

י. דרך לפענוח מאפשרת לנו להחליף אותיות ספציפיות באותיות אחרות, על מנת לעשות זאת  
אנו נרשום את הפקודה הבאה:

```
tr 'aet' 'OPL' <bubble.txt> out.txt
```

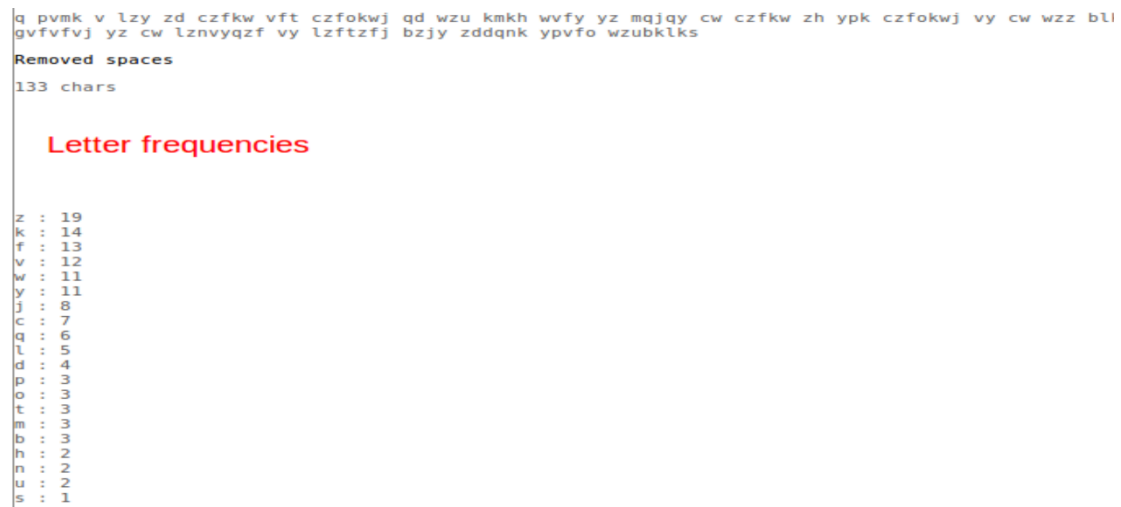
ה-out.txt זה המסמך החדש שבו ישמרו השינויים.

```
[01/05/19]seed@VM:~/me$ tr 'aet' 'OPL' < bubble.txt >  
out.txt
```

| article   | x | *out.txt | x |
|---|---|----------|---|
| hi my n0MP is lior0 Ond i likP Lo POL icP crP0m |   |          |   |

כ. כעת נעשה את הדבר הבא:

ננסה לפענח מהו המלל הטמון בטקסט הזה באמצעות כך שנבדוק כמה פעמים מופיעה כל אות, אנו נעזר באתר – <http://www.richkni.co.uk/php/crypta/freq.php> על מנת לעשות כן: כפי שאנו רואים בתמונה הבאה:



ל. נבדוק את זה גם לפי צירופים, על מנת לעשות זאת פשוט נגלול את האתר כלפי מטה ונראה מה קורה כשיש צירופים של שתי אותיות שחוזרים על עצמם ומה קורה כשיש עם שלוש אותיות שחוזרים על עצמם:

#### 2 letter sequences

```
zf => 5
wz => 4
vf => 4
kw => 4
cz => 4
```

#### 3 letter sequences

```
czf => 4
```

## 2.2.2

א. ניצור טקסט חדש ונמלא אותו בטקסט

```
[01/05/19]seed@VM:~/me$ touch lg.txt
```

ב. נרשום את הפקודה:

```
openssl enc -ciphertext -e -in lg.txt -out cipher.bin  
-k 00112233445566778899aabbccddeeff  
-iv 0102030405060708
```

```
> -k 00112233445566778899aabbccddeeff \  
> -iv 0102030405060708
```

ג. נראה שקיבלנו:

```
[01/05/19]seed@VM:~/me$ openssl enc -ciphertext -e -in lg.t  
xt -out cipher.bin \  
> -k 00112233445566778899aabbccddeeff \  
> -iv 0102030405060708  
unknown option '-ciphertext'  
options are  
-in <file>      input file  
-out <file>     output file  
-pass <arg>    pass phrase source
```

ד. נחליף את "ciphertext" ב- -aes-128-cfb, -bf-cbc, -aes-128-cbc, כלומר:

```
openssl enc -aes-128-cbc -in lg.txt -out cipher.bin  
-k 00112233445566778899aabbccddeeff  
-iv 0102030405060708
```

```
[01/05/19]seed@VM:~/me$ openssl enc -aes-128-cbc -e -in lg.  
txt -out chiper.bin -K 00112233445566778899aabbccddeeff -iv  
0102030405060708
```

דרך נוספת זה לעשות:

```
openssl enc -bf-cbc -e -in lg.txt -out cipher.bin  
-k 00112233445566778899aabbccddeeff  
-iv 0102030405060708
```



```
[01/05/19]seed@VM:~/me$ openssl enc -aes-128-cfb -e -in lg.tx  
t -out chiper.bin -K 00112233445566778899aabbccddeeff -iv 010  
2030405060708
```

כמו כן נוכל לעשות:

```
openssl enc -aes-128-cfb -e -in lg.txt -out cipher.bin  
-k 00112233445566778899aabbccddeeff  
-iv 0102030405060708
```

```
[01/05/19]seed@VM:~/me$ openssl enc -bf-cbc -e -in lg.txt -out chiper3.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708
```

נראה שקיבלנו כתוצאה מכל המהלכים הללו את הקבצים הבאים:

|   |             |          |        |       |
|---|-------------|----------|--------|-------|
|  | chiper.bin  | 30 bytes | Binary | 10:58 |
|  | chiper2.bin | 32 bytes | Binary | 10:58 |
|  | chiper3.bin | 32 bytes | Binary | 10:59 |

### 2.3.3

א. נוריד את התמונה מהאתר שניתן לנו:





ב. כעת נצפין את התמונה המקורית בשתי הצפנות כמבוקש הצפנת ECB והצפנת CBC:

```
[01/05/19]seed@VM:~/.../matala$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out pic1.bmp
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
[01/05/19]seed@VM:~/.../matala$ openssl enc -aes-128-cbc -e -in pic_original.bmp -out pic2.bmp
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
```

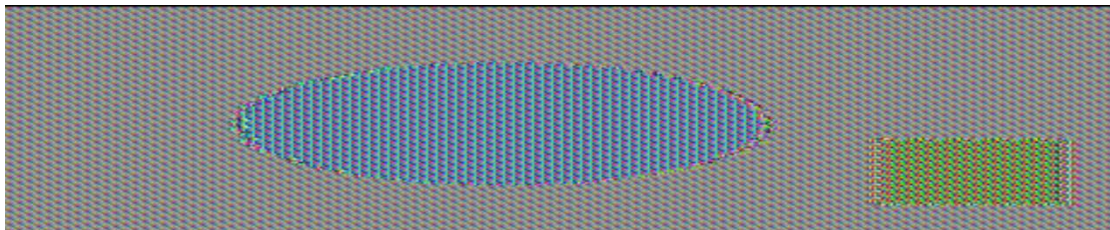
וניצור שתי תמונות מוצפנות אחת pic1 עם ECB והשנייה pic2 כ-CBC.

כעת ניקח את ה-54 בייטים של התמונה הראשונה בה יש את כל הגדרות התמונה בכדי שנוכל לפתוח את התמונה ונקרא לזה head, נקח את החלק הממוצפן של התמונה המקורית מהבייטים 55 ועד סוף הקובץ ונקרא לו body, נחבר את החלק של ההגדרות עם שאר החלק המוצפן ונקבל תמונות שניתן לפתוח ולראות את התמונה המוצפנת עם הפקודה eog.

```
[01/10/19]seed@VM:~/.../matala$ head -c 54 pic_original.bmp > head
[01/10/19]seed@VM:~/.../matala$ tail -c +55 pic1.bmp > body
[01/10/19]seed@VM:~/.../matala$ cat head body > New.bmp
[01/10/19]seed@VM:~/.../matala$ eog New.bmp
```

```
[01/10/19]seed@VM:~/.../matala$ head -c 54 pic_original.bmp > head
[01/10/19]seed@VM:~/.../matala$ tail -c +55 pic2.bmp > body2
[01/10/19]seed@VM:~/.../matala$ cat head body2 > New2.bmp
[01/10/19]seed@VM:~/.../matala$ eog New2.bmp
```

כעת נפתח את ההצפנה של ECB ונראה שהיא לא הצפנה הכי בטוחה .



לעומת הצפנת CBC שהיא הצפנה שכבר לא ניתן לראות מה הייתה התמונה הקודמת לפני ההצפנה



### 3.6.2

א. נעשה את הדבר הבא:

ננסה להצפין קובץ טקסט עם שימוש של קודי IV שונים ולאחר מכן אותו קובץ טקסט עם הצפנה של קוד IV דומה

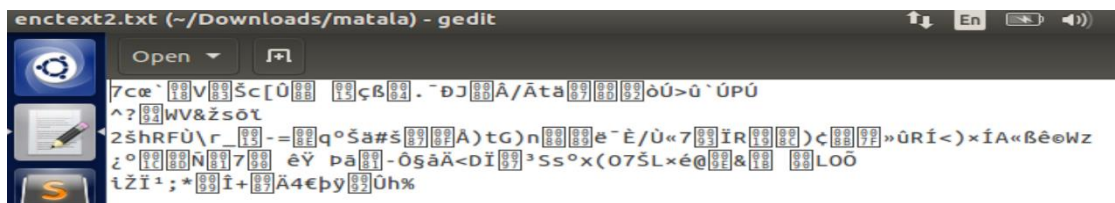
```
[01/06/19]seed@VM:~/.../matala$ openssl enc -bf-cbc -e -K 0123456789 -iv 12345 -in plaintext.txt -out enc2text.txt
[01/06/19]seed@VM:~/.../matala$ openssl enc -bf-cbc -e -K 0123456789 -iv 12435 -in plaintext.txt -out enc2text2.txt
[01/06/19]seed@VM:~/.../matala$
```

לתשומת לבכם- קוד ה- IV שונה טיפה מ: "12345" ל: "12435" בכדי ליצור שתי קצבים דומים עם הצפנה שונה.

```
[01/06/19]seed@VM:~/.../matala$ openssl enc -bf-cbc -e -K 0123456789 -iv 654321 -in plaintext.txt -out enc2text3.txt
[01/06/19]seed@VM:~/.../matala$ openssl enc -bf-cbc -e -K 0123456789 -iv 654321 -in plaintext.txt -out enc2text4.txt
[01/06/19]seed@VM:~/.../matala$
```

לעומת הצפנה של אותו קוד IV שיוצר שתי הצפנות דומות.  
\*חשוב לזכור שאם יש שינוי בטקסט ניתן להשתמש באותו IV, אך בכדי להצפין אותו טקסט בכמה דרכים כדאי להחליף קוד IV.

כאן נראה שכאשר שינינו את קוד ה- IV שונתה גם ההצפנה של הקובץ:



ב. נמשיך ונראה שכאשר שומש אותו קוד IV לאותו קובץ נוצר אותו הקובץ בדיוק :



#### 3.1.4

א. נתון לנו ש-  $p, q, e$  מספרים ראשוניים.

נאמר ש:  $n = p * q$

נעזר ב-  $e, n$  כמפתח ציבורי.

עלינו למצוא מהו ערכו של המפתח האישי  $d$ .

נתונים הערכים ההקסה דצימלים של  $e, p, q$ :

$p = F7E75FDC469067FFDC4E847C51F452DF$

$q = E85CED54AF57E53E092113E62F436F4F$

$e = 0D88C3$

על מנת למצוא מהו  $n$  נעזר בנתונים שלנו:

$n = p * q$

נבחר  $e$ , על מנת שה- $e$  שבחרנו יתאים לנו לעסק אנו נחשב את ה-  $\varphi(n)$  לפי פונקציית אוילר:

$$\varphi(n) = (p - 1)(q - 1)$$

נשים לב שה- $e$  שלנו שלם בין  $1 < e < \varphi(n)$ , כמו כן על  $e$  להיות זר ל- $\varphi(n)$ .

כדי לדעת מהו ערכו של  $d$  נאמר:

$$d * e = 1 \pmod{\varphi(n)}$$

נאמר ש:

$$\text{Cipher} = (\text{message})^e \pmod{n}$$

$$\text{message} = (\text{Cipher})^d \pmod{n}$$

כיוון שהפונקציה שרשמנו מוצאת הופכי מודלרי נוכל להעזר ב- $e$  הנתון לנו כדי למצוא את  $d$ , כלומר  
נאמר:

$$d = \frac{1}{e} \pmod{\varphi(n)}$$

$$d * e = 1 \pmod{\varphi(n)}$$

נסתכל על הקוד בשפת c

`BN_mod_inverse (d,e,  $\varphi(n)$ ,ctx)`

כאשר  $d$  זה המקום בו ישמר החישוב.

$ctx$  יהיה לנו לעזר.

באמצעות הקוד המופיע מטה אנו יכולים למצוא מהו  $d$ .

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



```

void printBN(char * msg, BIGNUM * a){// changes were made here
for printing style
    char * number_str = BN_bn2hex(a);
    printf("%s%s", msg , number_str);
    OPENSSL_free(number_str);
}

int main(){
    BN_CTX * ctx = BN_CTX_new();

    BIGNUM *i = BN_new();
    BIGNUM *p = BN_new();
    BIGNUM *pminus = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *qminus = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *piN = BN_new();

    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
    BN_hex2bn(&e, "0D88C3");
    BN_dec2bn(&i, "01");

    BN_mul(n, p, q, ctx);
    printf("Public key is : ( ");printBN("", e);printBN(", ",
n);
    printf(")\n");

    BN_sub(pminus,p,i);
    BN_sub(qminus,q,i);
    BN_mul(piN, pminus, qminus, ctx);
    BN_mod_inverse(d, e, piN, ctx);

    printf("Private key is : ( ");printBN("",d);printf(")\n");

    return 0;
}

```

3.2:

א. נתונים לנו  $d, n, e$  ואנחנו צריכים להצפין את ההודעה: "A top secret!"

כעת נשתמש בפקודה בשפת C:

$BN\_mod\_exp(c, m, e, n, ctx);$

בשביל ליצור את המשוואה הבאה:  $Cipher = (message)^e \pmod n$

```

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

```

```

void printBN(char * msg, BIGNUM * a){// changes were made here
for printing style
    char * number_str = BN_bn2hex(a);
    printf("%s%s", msg , number_str);
    OPENSSL_free(number_str);
}

int main(){
    BN_CTX * ctx = BN_CTX_new();

    BIGNUM *n = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *m = BN_new();//message
    BIGNUM *c = BN_new();//cipher text

    BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB
1A5");
    BN_hex2bn(&e, "010001");
    BN_hex2bn(&m, "4120746f702073656372657421");

    BN_mod_exp(c, m, e, n, ctx);
    printBN("Cipher text: ",c);

    return 0;
}

```

ונקבל את ההודעה המוצפנת :

6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC

```

[01/10/19]seed@VM:~/.../matala$ gcc -o encryptMessage bn_sample.c
-l crypto
[01/10/19]seed@VM:~/.../matala$ encryptMessage
Cipher text: 6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5
DE5CFC5FADC[01/10/19]seed@VM:~/.../matala$ █

```

3.3:

בכדי לפענח את ההודעה מהמשימה הקודמת נשתמש במפתח שלנו d ובעזרת המשוואה :

$message = (Cipher)^d \pmod n$  נקבל את ההודעה הרגילה שהיא בבסיס אקסדצימלי

נעזר בקוד בכדי לחשב את הכל :

```

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

void printBN(char * msg, BIGNUM * a){// changes were made here
for printing style
    char * number_str = BN_bn2hex(a);
    printf("%s%s", msg , number_str);
    OPENSSL_free(number_str);
}

int main(){
    BN_CTX * ctx = BN_CTX_new();

    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *m = BN_new();//message

```

```

    BIGNUM *c = BN_new();//cipher text

    BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB
1A5");
    BN_hex2bn(&d, "
74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D3
0D");
    BN_hex2bn(&c,
"6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5F
AD");

    BN_mod_exp(m, c, d, n, ctx);
    printBN("message in hex: ", m);
    printf("\n");

    return 0;
}

```

ובסוף נשתמש בפקודה `python -c`

```
'print("4120746f702073656372657421".decode("hex"))'
```

ונקבל את ההודעה : **A top secret!**

```

[01/10/19]seed@VM:~/.../matala$ gcc -o decryption bn_sample.c -l
crypto
[01/10/19]seed@VM:~/.../matala$ decryption
Message: 4120746F702073656372657421
[01/10/19]seed@VM:~/.../matala$ python -c 'print("4120746F70207365
6372657421".decode("hex"))'
A top secret!
[01/10/19]seed@VM:~/.../matala$

```

### 3.4:

בכדי לחתום על הצפנה כדי לשייך אותה לאדם מסויים נעזר במפתח הפרטי כדי להצפין את ההודעה ובכדי לפענח אחרים הצתרכו להשתמש במפתח הציבורי של אותו אדם

המשוואה הזאת תדגים את אופן הביצוע:  $\text{sign} = (\text{message})^d \pmod{n}$

נעזר בקוד בכדי לחתום על הודעה:

```

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

void printBN(char * msg, BIGNUM * a){// changes were made here
for printing style
    char * number_str = BN_bn2hex(a);
    printf("%s%s", msg , number_str);
    OPENSSL_free(number_str);
}

int main(){
    BN_CTX * ctx = BN_CTX_new();

    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *m = BN_new();//message
    BIGNUM *s = BN_new();//signed text

```

```

    BN_hex2bn(&n,
"DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB
1A5");
    BN_hex2bn(&d, "
74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D3
0D");
    BN_hex2bn(&m, "49206f776520796f75202432303030");

    BN_mod_exp(s, m, d, n, ctx);
    printBN("signed Message: ", s);
    printf("\n");

    return 0;
}

```

ובקוד נשנה בכל הרצה את ההודעה שאנחנו מצפינים בכדי לחתום על הודעה שונה ונראה שיוצאת הצפנה שונה

```

[01/07/19]seed@VM:~/.../matala$ python -c 'print("I owe you $2000"
.encode("hex"))'
49206f776520796f75202432303030
[01/07/19]seed@VM:~/.../matala$ gcc -o Sign bn_sample2.c -l crypto
[01/07/19]seed@VM:~/.../matala$ Sign

signed Message:80A55421D72345AC199836F60D51DC9594E2BDB4AE20C804823
FB71660DE7B82
[01/07/19]seed@VM:~/.../matala$ python -c 'print("I owe you $3000"
.encode("hex"))'
49206f776520796f75202433303030
[01/07/19]seed@VM:~/.../matala$ gcc -o Sign bn_sample2.c -l crypto
[01/07/19]seed@VM:~/.../matala$ Sign

signed Message:04FC9C53ED7BBE4ED4BE2C24B0BDF7184B96290B4ED4E3959F5
8E94B1ECEAE2EB
[01/07/19]seed@VM:~/.../matala$ █

```

3.5:

בכדי לאמת את ההודעה נשתמש במשוואה הזאת:  $\text{Verify} = (\text{sign})^e \pmod n$

נשתמש בקוד בכדי לחשב את ההודעה ולאמת:

```

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

void printBN(char * msg, BIGNUM * a){// changes were made here
for printing style
    char * number_str= BN_bn2hex(a);
    printf("%s%s", msg , number_str);
    OPENSSL_free(number_str);
}

int main(){
    BN_CTX * ctx = BN_CTX_new();

    BIGNUM *n = BN_new();
    BIGNUM *d = BN_new();
    BIGNUM *m = BN_new();//message
    BIGNUM *s = BN_new();//signed text

    BN_hex2bn(&n,
"AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116
115");

```

```

    BN_hex2bn(&e, "010001");
    BN_hex2bn(&s,
"643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB68
02F");

    BN_mod_exp(m, s, e, n, ctx);
    printBN("Verified message: ", s);
    printf("\n");

    return 0;
}

```

וכעת נשתמש בקוד בפייטון בכדי להעביר את ההודעה ל ascii ונקבל את ההודעה הרצויה

```

[01/07/19]seed@VM:~/.../matala$ gcc -o Verify bn_sample3.c -l cryp
to
[01/07/19]seed@VM:~/.../matala$ Verify
Verified message: 4C61756E63682061206D697373696C652E
[01/07/19]seed@VM:~/.../matala$ python -c 'print("4C61756E63682061
206D697373696C652E".decode("hex"))'
Launch a missile.
[01/07/19]seed@VM:~/.../matala$ █

```

ואם נקבל מידע שהוא לא נכון.. נשנה בקוד את ההודעה החתומה לסימט של 3F ונסה לראות את הפענוח של ההודעה:

```

[01/07/19]seed@VM:~/.../matala$ gcc -o Verify bn_sample3.c -l cryp
to
[01/07/19]seed@VM:~/.../matala$ Verify
Verified message: 4C61756E63682061206D697373696C652E
[01/07/19]seed@VM:~/.../matala$ python -c 'print("4C61756E63682061
206D697373696C652E".decode("hex"))'
Launch a missile.
[01/07/19]seed@VM:~/.../matala$ gcc -o Verify bn_sample3.c -l cryp
to
[01/07/19]seed@VM:~/.../matala$ Verify
Verified message: 91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6
B7BE0203B41AC294
[01/07/19]seed@VM:~/.../matala$ python -c 'print("91471927C80DF1E4
2C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294".decode("hex"))'
00, 00c000rm=f0:N000000000
[01/07/19]seed@VM:~/.../matala$ █

```

נראה שנקבל הודעה ממש לא ברורה.