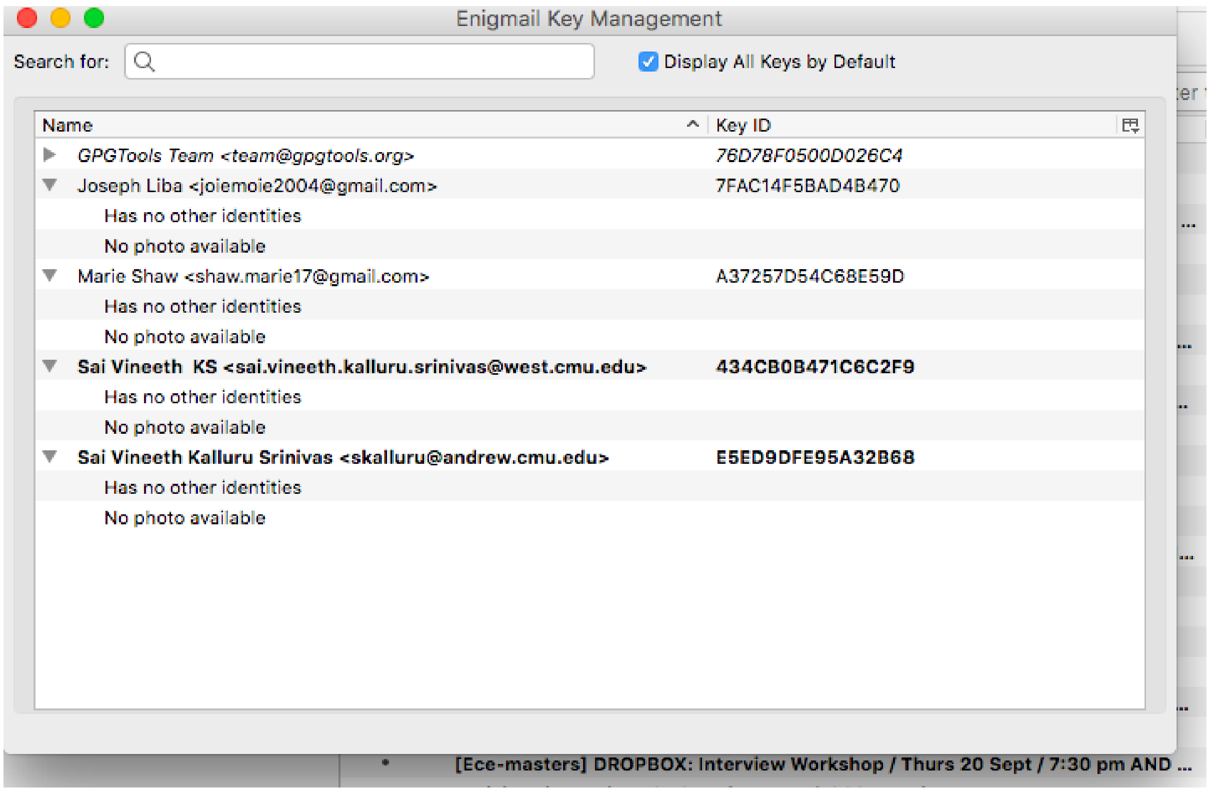
18631 – INTRODUCTION TO INFORMATION SECURITY

SAI VINEETH KALLURU SRINIVAS
CARNEGIE MELLON UNIVERSITY – SILICON VALLEY
ANDREW ID : SKALLURU

USING PGP (QUESTION 1) (CTF username – kssaivineeth15)

- PGP stands for Pretty good privacy.
- Usually email clients and email servers might not have protected end to end encryption services.
- PGP is an end to end encryption service.
- There is always a MITM possible where the email can be spoofed before it can leave the client PC to the nearest gateway possible
- Therefore it is recommended to use PGP for email communication
- There are multiple methods to setup PGP on different EMail clients.
- For example, We can use Mozilla thunderbird to setup PGP.
  - ENGMAIL is 'add-on' required to be installed on mozilla thunderbird.
  - Setup thunderbird to your official mail id accound (in my case: skalluru@andrew.cmu.edu)
  - We have to create a pubic key private key pair for the above setup email.
  - Usually the same keys can be used for Signing, Authenticating and Encrypting. However separate keys for S, E and A - 'subkeys' can be used for more protection.
  - Signing is usually used to prevent identity attacks from MITM, this maintains the sender identity .
  - Encryption maintains data integrity by preventing MITM to change or read or modify the data of the body of email.
- The following website was referred to complete PGP assignment – 'https://securityinabox.org/en/guide/thunderbird/mac/, https://support.mozilla.org/en-US/kb/digitally-signing-and-encrypting-messages'
- They keys were generated for two of my mail accounts, Andrew account and sv.cmu.edu account
- The other keys that are visible are the public keys of my friends which were used to test whether my PGP was working fine or not.


Enigmail Key Management

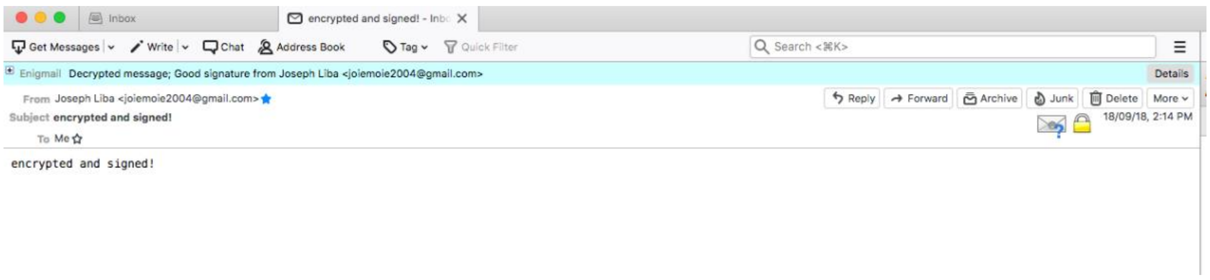
Search for: ✓ Display All Keys by Default

Name	Key ID
▶ *GPGTools Team <team@gpgtools.org>*	*76D78F0500D026C4*
▼ Joseph Liba <joiemoie2004@gmail.com>	7FAC14F5BAD4B470
Has no other identities	
No photo available	
▼ Marie Shaw <shaw.marie17@gmail.com>	A37257D54C68E59D
Has no other identities	
No photo available	
▼ **Sai Vineeth KS <sai.vineeth.kalluru.srinivas@west.cmu.edu>**	**434CB0B471C6C2F9**
Has no other identities	
No photo available	
▼ **Sai Vineeth Kalluru Srinivas <skalluru@andrew.cmu.edu>**	**E5ED9DFE95A32B68**
Has no other identities	
No photo available	

[Ece-masters] DROPBOX: Interview Workshop / Thurs 20 Sept / 7:30 pm AND ...

WORKING WITH PGP

- First the public keys have to be exchanged in a preferably secured channel.
- In our case we shared them via our emails only. To verify the public key given by anyone it has to be TRUSTED by any certificated provider. Or it can be confirmed over other channels like Voice call, Video call, In person etc.
- Once the public key of the person is ‘imported’ from the attachment sent by the person (to whom our email is intended to) we can start sending mails to them.
- The public key has to be attached to an email (usually has the extention ‘.asc’) and can be imported into the Engmail key management of thunderbird with a single option selection.
- Now Enigmail choses PGP/MIME in default
- Usually PGP inline is chosen for normal mails and PGP/MIME if multimedia messages are attached or sent (like images, video etc). PGP MIME is more secure and highly preferred.
- The engmail option in the menu bar of thunderbird allows to set up engmail preferences to either sign or encrypt or both.
- We can choose the corresponding option and send mails. When the other person has our public key they can send signed and encrypted mails like below



Enigmail Decrypted message; Good signature from Joseph Liba <joiemoie2004@gmail.com>

From Joseph Liba <joiemoie2004@gmail.com>
Subject encrypted and signed!
To Me
18/09/18, 2:14 PM

```
encrypted and signed!
```

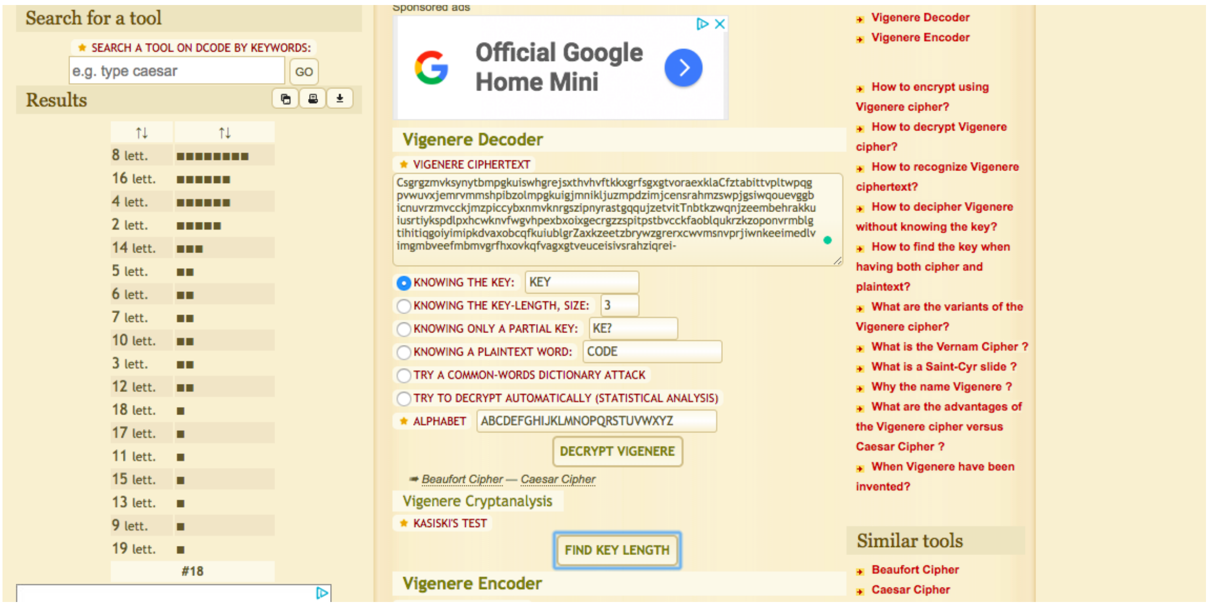
-
- We can see that it is mentioned as decrypted message and good signature from my friend Joseph. He would have got a similar notification when I sent my signed and encrypted messages.
- In the receivers side , the person receiving would use their private keys to decrypt the message, and use our public key to design the message.
- In reality engmail for thunderbird takes care of the decryption automatically and throws a similar notification as in the screenshot above.
- For the assignment we were asked to send mails to ta18631@gmx.com, the key was trusted by looking it at http://keyserver.ubuntu.com/pks/lookup?op=vindex&search=0x6FC8F0B4FE1316C8 public key repository.
- Further it can also be trusted by confirmed the key letter by letter on a secure phone or any preffered communication. This is encouraged because this verification is need only once.
- Based on the above working all three mails have been sent to the ta18631@gmx.com
- My user id : Sai Vineeth KS sai.vineeth.kalluru.srinivas@west.cmu.edu

  Public Key ID:

  A6BE B190 61B2 0E6E 24BA BF77 434C B0B4 71C6 C2F9

BREAKING VIGNERE (QUESTION 2) (CTF username – kssaivineeth15)

- For breaking the vignere the following the website was accessed ‘ https://www.dcode.fr/vigenere-cipher’
- Opening the Ip address mentioned in the questions redirects to html page that provides an encrypted message in vignere encryption method.
- The text has been copy pasted into the online solver as shown below



Search for a tool

SEARCH A TOOL ON DCODE BY KEYWORDS: e.g. type caesar GO

Results

↑↓	↑↓
8 lett.	■■■■■■■■
16 lett.	■■■■■■
4 lett.	■■■■■■
2 lett.	■■■■■
14 lett.	■■■
5 lett.	■■
6 lett.	■■
7 lett.	■■
10 lett.	■■
3 lett.	■■
12 lett.	■■
18 lett.	■
17 lett.	■
11 lett.	■
15 lett.	■
13 lett.	■
9 lett.	■
19 lett.	■
#18	

Sponsored ads

Official Google Home Mini

Vigenere Decoder

VIGENERE CIPHERTEXT

```
CsgrgzmvksynytbmpgkuiswhgrejsxthvhvftkkxgrfsgxgtvoraexklaCfztabittvpltwpqg
pvwuvxjemrvmmshpibzolmpgkuigjmnikljuzmpdzimjcensrahmzswpjgsiwqouevggb
icnuvrzmvcckjmzpiccybxnmvknrgszipnyrastgqqujzetvitTnbtkzwqnjzeembehrakku
iusrtiykspdlpxhcwknvfwgvhpexbxoixgecrgzzspitpstbvcckfaoblqukrzkzoponvrmblg
tihitiqgoiyimipkdvaxobcqfkuiublgrZaxkzeetzbrywzgrerxcwvmsnvprjiwnkeeimedlv
imgmbveefmbmvgrfhxovkqfvagxgtveuceisivsrahziqrei-
```

[✓ KNOWING THE KEY:] KEY

KNOWING THE KEY-LENGTH, SIZE: 3

KNOWING ONLY A PARTIAL KEY: KE?

KNOWING A PLAINTEXT WORD: CODE

TRY A COMMON-WORDS DICTIONARY ATTACK

TRY TO DECRYPT AUTOMATICALLY (STATISTICAL ANALYSIS)

ALPHABET ABCDEFGHIJKLMNOPQRSTUVWXYZ

DECRYPT VIGENERE

➡ Beaufort Cipher — Caesar Cipher

Vigenere Cryptanalysis

KASISKI'S TEST

FIND KEY LENGTH

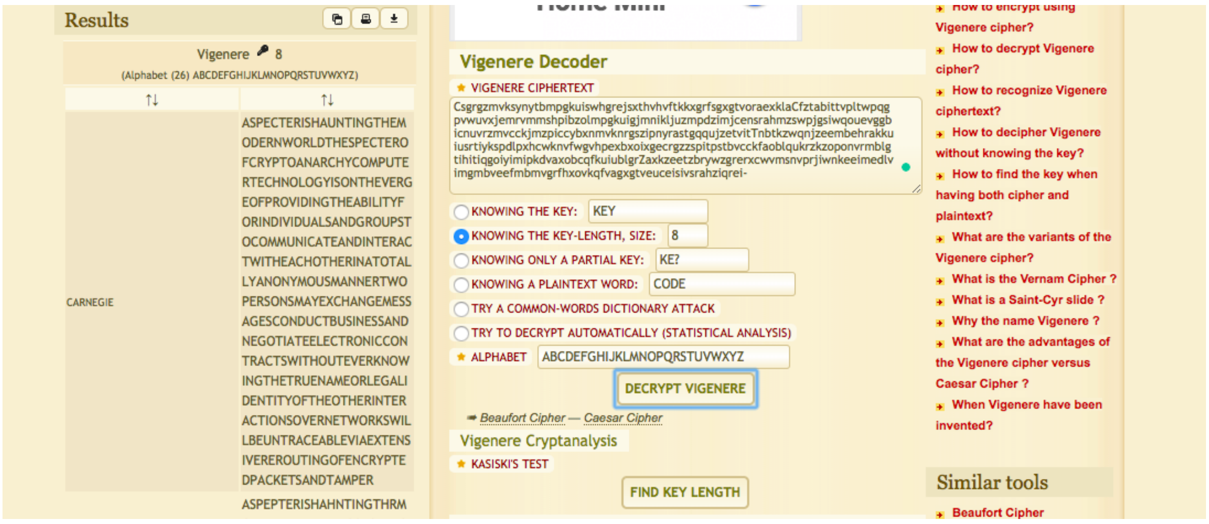
Vigenere Encoder

- Vigenere Decoder
- Vigenere Encoder
- How to encrypt using Vigenere cipher?
- How to decrypt Vigenere cipher?
- How to recognize Vigenere ciphertext?
- How to decipher Vigenere without knowing the key?
- How to find the key when having both cipher and plaintext?
- What are the variants of the Vigenere cipher?
- What is the Vernam Cipher ?
- What is a Saint-Cyr slide ?
- Why the name Vigenere ?
- What are the advantages of the Vigenere cipher versus Caesar Cipher ?
- When Vigenere have been invented?

Similar tools

- Beaufort Cipher
- Caesar Cipher

- Then FIND KEY LENGTH is checked for, the website uses frequency analysis attack and finds out the all key lengths possible. In this case the key length with maximum value is found to be ‘8’.
- Now the value can be added to ‘Knowing the key length:’ as 8. Now the results show as below


Results

Vigenere 8
(Alphabet (26) ABCDEFGHIJKLMNOPQRSTUVWXYZ)

↑↓	↑↓
CARNEGIE	ASPECTERISHAUNTINGTHEM ODERNWORLDTHESPECTERO FCRYPTOANARCHYCOMPUTE RTECHNOLOGYISONTHEVERG EOFPROVIDINGTHEABILITYF ORINDIVIDUALSANDGROUPST OCOMMUNICATEANDINTERAC TWITHEACHOTHERINATOTAL LYANONYMOUSMANNERTWO PERSONSMAYEXCHANGEMESS AGESCONDUCTBUSINESSAND NEGOTIATEELECTRONICCON TRACTSWITHOUTEVERKNOW INGTHETRUENAMEORLEGALI DENTITYOFTHEOTHERINTER ACTIONSOVERNETWORKSWIL LBEUNTRACEABLEVIAEXTENS IVEREROUTINGOFENCRYPTE DPACKETSANDTAMPER
	ASPEPTERISHAHNTINGTHRM

Vigenere Decoder

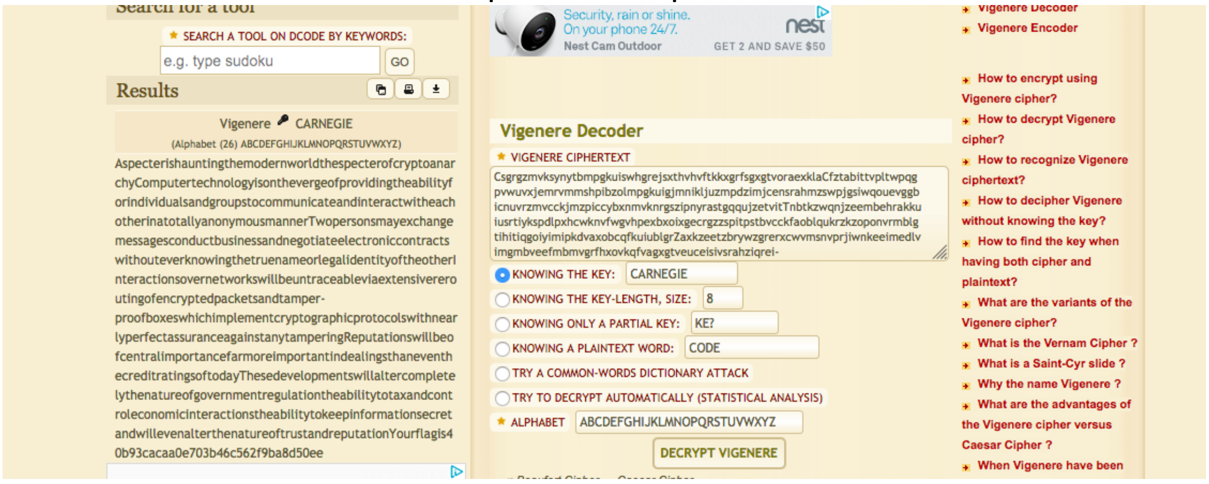
VIGENERE CIPHERTEXT

```
CsgrgzmvksynytbmpgkuiswhgrejsxthvhvftkkxgrfsgxgtvoraexklaCfztabittvpltwpqg
pvwuvxjemrvmmshpibzolmpgkuigjmnikljuzmpdzimjcensrahmzswpjgsiwqouevggb
icnuvrzmvcckjmzpiccybxnmvknrgszipnyrastgqqujzetvitTnbtkzwqnjzeembehrakku
iusrtiykspdlpxhcwknvfwgvhpexbxoixgecrgzzspitpstbvcckfaoblqukrzkzoponvrmblg
tihitiqgoiyimipkdvaxobcqfkuiublgrZaxkzeetzbrywzgrerxcwvmsnvprjiwnkeeimedlv
imgmbveefmbmvgrfhxovkqfvagxgtveuceisivsrahziqrei-
```

KNOWING THE KEY: KEY
[✓ KNOWING THE KEY-LENGTH, SIZE: 8]
KNOWING ONLY A PARTIAL KEY: KE?
KNOWING A PLAINTEXT WORD: CODE
TRY A COMMON-WORDS DICTIONARY ATTACK
TRY TO DECRYPT AUTOMATICALLY (STATISTICAL ANALYSIS)
ALPHABET ABCDEFGHIJKLMNOPQRSTUVWXYZ

DECRYPT VIGENERE
Beaufort Cipher — Caesar Cipher
Vigenere Cryptanalysis
KASISKI'S TEST
FIND KEY LENGTH
How to encrypt using Vigenere cipher?
How to decrypt Vigenere cipher?
How to recognize Vigenere ciphertext?
How to decipher Vigenere without knowing the key?
How to find the key when having both cipher and plaintext?
What are the variants of the Vigenere cipher?
What is the Vernam Cipher ?
What is a Saint-Cyr slide ?
Why the name Vigenere ?
What are the advantages of the Vigenere cipher versus Caesar Cipher ?
When Vigenere have been invented?
Similar tools
Beaufort Cipher

- We can see that the Key was found about to be *CARNEGIE* and the same key is now used to crack the entire encryption.
- Now the value 'carnegie' can be added to 'KNOWING THE KEY:' row which is later used to crack and provide the plaintext



SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type sudoku
GO
Security, rain or shine. On your phone 24/7.
nest
Nest Cam Outdoor
GET 2 AND SAVE $50

Results

Vigenere CARNEGIE
(Alphabet (26) ABCDEFGHIJKLMNOPQRSTUVWXYZ)

```
Aspecterishauntingthemodernworldthespecterofcryptoanar
chyComputertechnologyisonthevergeofprovidingtheabilityf
orindividualsandgroupstocommunicateandinteractwitheach
otherinatotallyanonymousmannerTwopersonsmayexchange
messagesconductbusinessandnegotiateelectroniccontracts
withouteverknowingthetruenameorlegalidentityoftheotherI
nteractionsovernetworkswillbeuntraceableviaextensivererο
utingofencryptedpacketsandtamper-
proofboxeswhichimplementcryptographicprotocolswithnear
lyperfectassuranceagainstanytamperingReputationswillbeo
fcentralimportancefarmoreimportantindealingsthaneventh
ecreditratingsoftodayThesedevelopmentswillaltercomplete
lythenatureofgovernmentregulationtheabilitytotaxandcont
roleconomicinteractionstheabilitytokeepinformationsecret
andwillevenalterthenatureoftrustandreputationYourflagis4
0b93cacaa0e703b46c562f9ba8d50ee
```

Vigenere Decoder

VIGENERE CIPHERTEXT

```
CsgrgzmvksynytbmpgkuiswhgrejsxthvhvftkkxgrfsgxgtvoraexklaCfztabittvpltwpqg
pvwuvxjemrvmmshpibzolmpgkuigjmnikljuzmpdzimjcensrahmzswpjgsiwqouevggb
icnuvrzmvcckjmzpiccybxnmvknrgszipnyrastgqqujzetvitTnbtkzwqnjzeembehrakku
iusrtiykspdlpxhcwknvfwgvhpexbxoixgecrgzzspitpstbvcckfaoblqukrzkzoponvrmblg
tihitiqgoiyimipkdvaxobcqfkuiublgrZaxkzeetzbrywzgrerxcwvmsnvprjiwnkeeimedlv
imgmbveefmbmvgrfhxovkqfvagxgtveuceisivsrahziqrei-
```

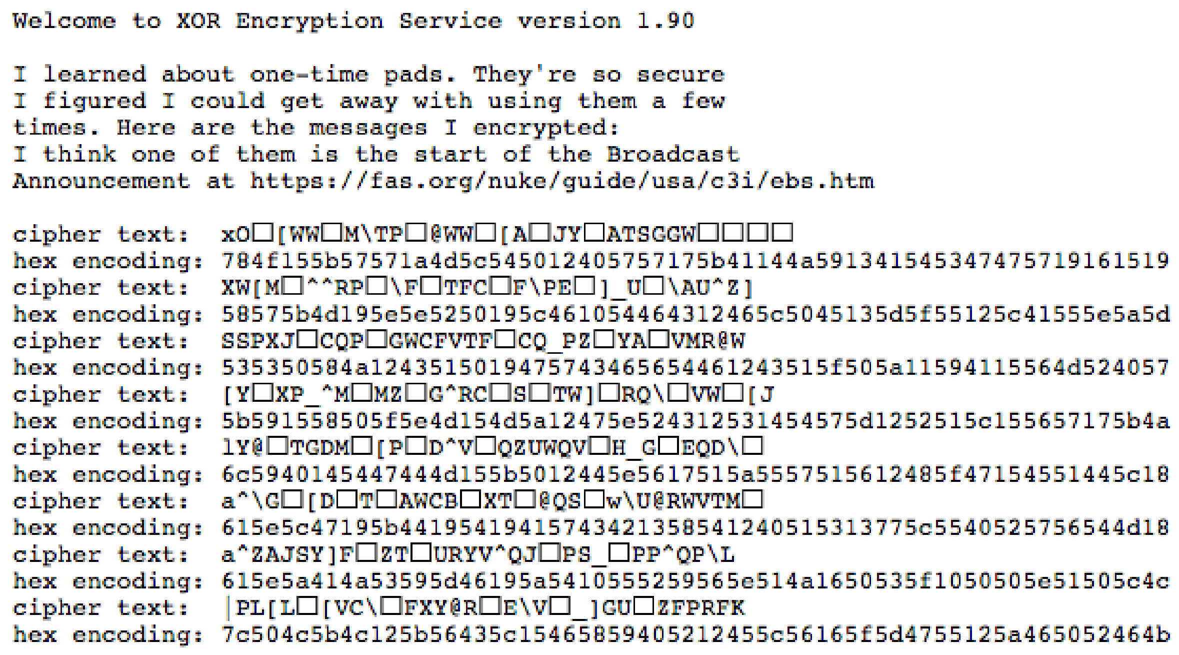
[✓ KNOWING THE KEY: CARNEGIE]
KNOWING THE KEY-LENGTH, SIZE: 8
KNOWING ONLY A PARTIAL KEY: KE?
KNOWING A PLAINTEXT WORD: CODE
TRY A COMMON-WORDS DICTIONARY ATTACK
TRY TO DECRYPT AUTOMATICALLY (STATISTICAL ANALYSIS)
ALPHABET ABCDEFGHIJKLMNOPQRSTUVWXYZ

DECRYPT VIGENERE
Vigenere Decoder
Vigenere Encoder
How to encrypt using Vigenere cipher?
How to decrypt Vigenere cipher?
How to recognize Vigenere ciphertext?
How to decipher Vigenere without knowing the key?
How to find the key when having both cipher and plaintext?
What are the variants of the Vigenere cipher?
What is the Vernam Cipher ?
What is a Saint-Cyr slide ?
Why the name Vigenere ?
What are the advantages of the Vigenere cipher versus Caesar Cipher ?
When Vigenere have been

- We can see the complete decrypted text in the left hand side of the website. The last sentence consists of the flag for the problem and the flag has been submitted.

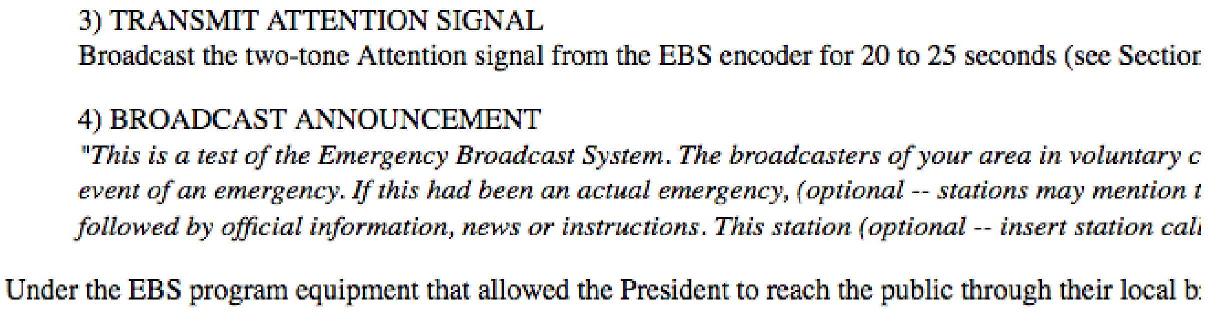
TWO TIME PADS (QUESTION 3) (CTF username – kssaivineeth15)

- The question was given as the pic below



```
Welcome to XOR Encryption Service version 1.90

I learned about one-time pads. They're so secure
I figured I could get away with using them a few
times. Here are the messages I encrypted:
I think one of them is the start of the Broadcast
Announcement at https://fas.org/nuke/guide/usa/c3i/ebs.htm

cipher text:   xO□[WW□M\TP□@WW□[A□JY□ATSGGW□□□□
hex encoding:  784f155b57571a4d5c545012405757175b41144a5913415453474757191615l9
cipher text:   XW[M□^^RP□\F□TFC□F\PE□]_U□\AU^Z]
hex encoding:  58575b4d195e5e5250195c461054464312465c5045135d5f55125c41555e5a5d
cipher text:   SSPXJ□CQP□GWCFVTF□CQ_PZ□YA□VMR@W
hex encoding:  535350584a12435150194757434656544612435l5f505a11594115564d524057
cipher text:   [Y□XP_^M□MZ□G^RC□S□TW]□RQ\□VW□[J
hex encoding:  5b591558505f5e4d154d5a12475e524312531454575d1252515c155657175b4a
cipher text:   lY@□TGDM□[P□D^V□QZUWQV□H_G□EQD\□
hex encoding:  6c59401454474 44d155b5012445e5617515a5557515612485f47154551445c18
cipher text:   a^\G□[D□T□AWCB□XT□@QS□w\U@RWVTM□
hex encoding:  615e5c47195b44195419415743421358541240515313775c55405257 56544d18
cipher text:   a^ZAJSY]F□ZT□URYV^QJ□PS_□PP^QP\L
hex encoding:  615e5a414a53595d46195a5410555259565e514a1650535f1050505e51505c4c
cipher text:   |PL[L□[VC\□FXY@R□E\V□_]GU□ZFPRFK
hex encoding:  7c504c5b4c125b56435c15465859405212455c56165f5d4755125a465052464b
```

- We can see that the first plain text we get is the first part of the start of the broadcast in the website, going to that website we can see this



3) TRANSMIT ATTENTION SIGNAL
Broadcast the two-tone Attention signal from the EBS encoder for 20 to 25 seconds (see Sectior

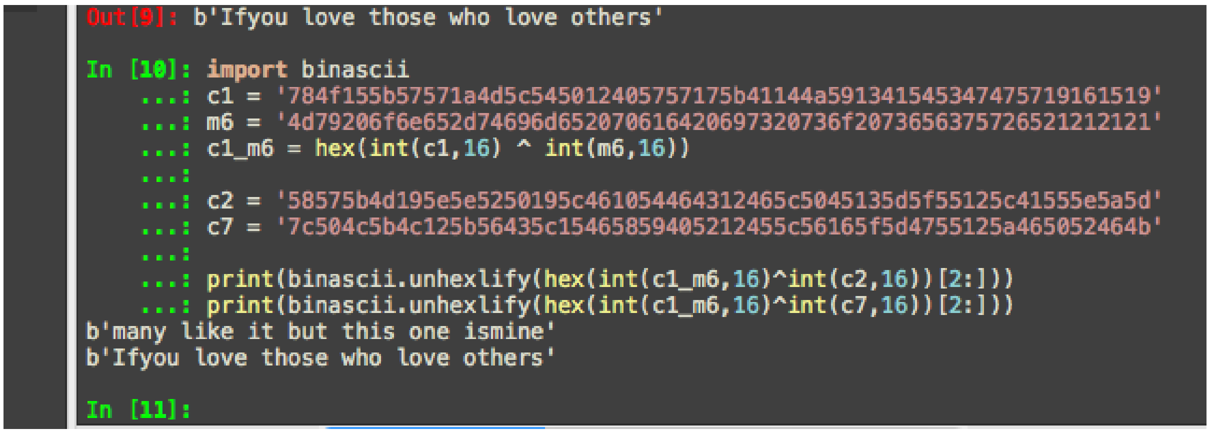
4) BROADCAST ANNOUNCEMENT
"This is a test of the Emergency Broadcast System. The broadcasters of your area in voluntary c
event of an emergency. If this had been an actual emergency, (optional -- stations may mention t
followed by official information, news or instructions. This station (optional -- insert station cali

Under the EBS program equipment that allowed the President to reach the public through their local b

- Counting the number of characters that matches with the 32 character cipher text we can see that the start of the broadcast message to be considered would be "This is a test of the emergency " (lets call this P1).
- We know that ciphertext (xor) key = plaintext . Now since Xor is reversible, we can say that plaintext (xor) ciphertext = key. Therefore I tried all cipher texts with the plaintext given to us(P1).
- Initially P1 has to be converted to hex code and this along with hex code,
  P1_hex =
  '546869732069732061207465737420 6f662074686520456d657267656e637920'
  And cipher text (C1) is xored. I have found that C6 XOR P1
  C1_hex =
  '784f155b57571a4d5c545012405757175b41144a591341545347475719161519' gave me the required key (lets call it K1) .

K1_hex =
2c277c28773e696d3d742477332377783d6160223c3304393635203277756c39

- It was informed that this key was reused, therefore K1 was tried with rest of the ciphers (C2……C8) it was found that C6 with hex 615e5c47195b4419541941574342135854124051531377 5c5540525756544d18 xored with K1 to give a proper plain text – 'My one-time pad is so secure!!!!' (lets call this P6)
- If the key has been used a couple of times, then Key 1 should have been used again, however K1 is a composite of C1 and P1, therefore C1 is composite of the Key
- On XORing C1 with other (C2…..C8) we could find an interesting pattern however XORing with P6 gave us the plaintexts of all patterns.



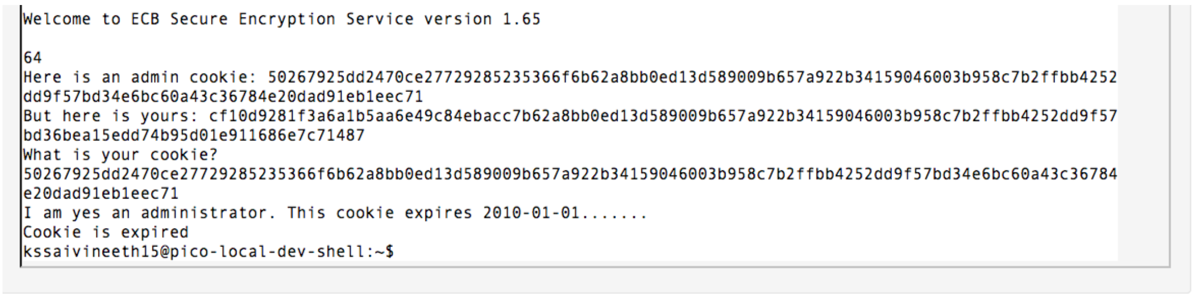
```
Out[9]: b'Ifyou love those who love others'

In [10]: import binascii
    ...: c1 = '784f155b57571a4d5c545012405757175b41144a591341545347475719161519'
    ...: m6 = '4d79206f6e652d74696d6520706164206973207365637572652121212121'
    ...: c1_m6 = hex(int(c1,16) ^ int(m6,16))
    ...:
    ...: c2 = '58575b4d195e5e5250195c461054464312465c5045135d5f55125c41555e5a5d'
    ...: c7 = '7c504c5b4c125b56435c15465859405212455c56165f5d4755125a465052464b'
    ...:
    ...: print(binascii.unhexlify(hex(int(c1_m6,16)^int(c2,16))[2:]))
    ...: print(binascii.unhexlify(hex(int(c1_m6,16)^int(c7,16))[2:]))
b'many like it but this one ismine'
b'Ifyou love those who love others'

In [11]:
```

Above picture shows the plain texts cracked for C2 and C7, here we can see C1 XOR P6 (m6) is common for both the plain text decryption. Therefore we can safely conclude that C1 XOR P6 is the required key.

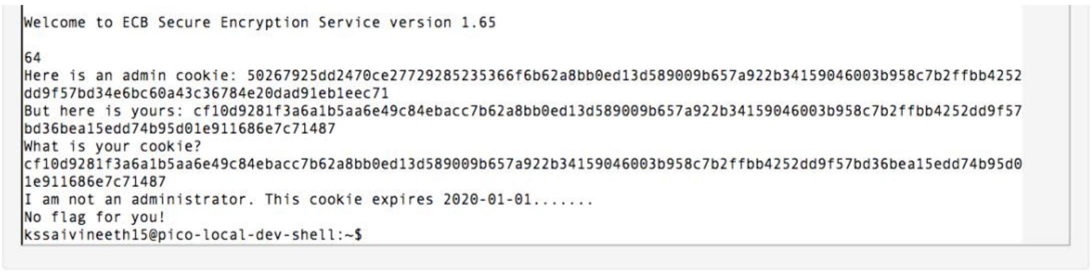
This key was submitted and it was correct. This example proves that if the PADS or KEYS are used twice they can be used to break other texts as well, as the Key can be reversed by XORing Cipher text and Plain text.

ECB (QUESTION 4) (CTF username – kssaivineeth15)

- This attack is relatively easier by exploiting the print(cookie2decoded) statement in the program. Lets consider the two screenshots given below

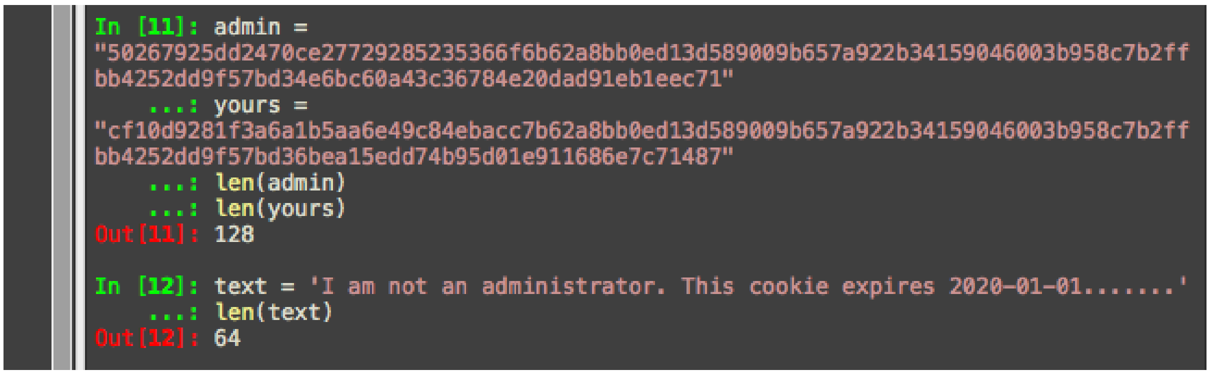


```
Welcome to ECB Secure Encryption Service version 1.65

64
Here is an admin cookie: 50267925dd2470ce27729285235366f6b62a8bb0ed13d589009b657a922b34159046003b958c7b2ffbb4252
dd9f57bd34e6bc60a43c36784e20dad91eb1eec71
But here is yours: cf10d9281f3a6a1b5aa6e49c84ebacc7b62a8bb0ed13d589009b657a922b34159046003b958c7b2ffbb4252dd9f57
bd36bea15edd74b95d01e911686e7c71487
What is your cookie?
50267925dd2470ce27729285235366f6b62a8bb0ed13d589009b657a922b34159046003b958c7b2ffbb4252dd9f57bd34e6bc60a43c36784
e20dad91eb1eec71
I am yes an administrator. This cookie expires 2010-01-01.......
Cookie is expired
kssaivineeth15@pico-local-dev-shell:~$
```



```
Welcome to ECB Secure Encryption Service version 1.65

64
Here is an admin cookie: 50267925dd2470ce27729285235366f6b62a8bb0ed13d589009b657a922b34159046003b958c7b2ffbb4252
dd9f57bd34e6bc60a43c36784e20dad91eb1eec71
But here is yours: cf10d9281f3a6a1b5aa6e49c84ebacc7b62a8bb0ed13d589009b657a922b34159046003b958c7b2ffbb4252dd9f57
bd36bea15edd74b95d01e911686e7c71487
What is your cookie?
cf10d9281f3a6a1b5aa6e49c84ebacc7b62a8bb0ed13d589009b657a922b34159046003b958c7b2ffbb4252dd9f57bd36bea15edd74b95d0
1e911686e7c71487
I am not an administrator. This cookie expires 2020-01-01.......
No flag for you!
kssaivineeth15@pico-local-dev-shell:~$
```

- By copy pasting both cookies we can see that the first one has an expired cookie date, yet has admin access. However the second cookie has not expired but doesn't give administrator permissions.
- Therefore the required plaintext to crack would be 'I am yes an administrator. This cookie expires 2020-01-01…….'(lets call it P_req)
- However since ECB uses block encryption method, we can find out the encrypted version of P_req by just swapping the required blocks.



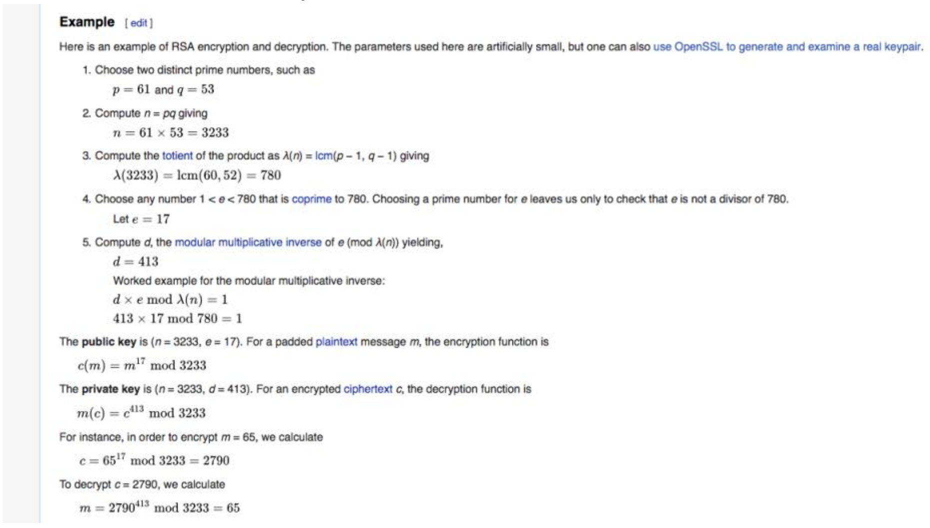
```
In [11]: admin =
"50267925dd2470ce27729285235366f6b62a8bb0ed13d589009b657a922b34159046003b958c7b2ff
bb4252dd9f57bd34e6bc60a43c36784e20dad91eb1eec71"
    ...: yours =
"cf10d9281f3a6a1b5aa6e49c84ebacc7b62a8bb0ed13d589009b657a922b34159046003b958c7b2ff
bb4252dd9f57bd36bea15edd74b95d01e911686e7c71487"
    ...: len(admin)
    ...: len(yours)
Out[11]: 128

In [12]: text = 'I am not an administrator. This cookie expires 2020-01-01.......'
    ...: len(text)
Out[12]: 64
```

- We can see that the text length in both cases in 64 bytes, but Cipher text(ECB encrypted) 128 bytes long. Therefore each plaintext translates to two in the cipher text. This correlation is possible because that’s the loophole of ECB.

- Comparing the cookies we can see that the first 32 differs and the last 32 differs. However the last 32 of the ‘your_cookie’ is something we require. However the first 32 of your_cookie should be replaced with first 32 (50267925dd2470ce27729285235366f6) of the admin cookie, because these 32 characters contains the ‘Iam yes and admi’ part of the admin cookie text.
- No combining the first 32 of admin cookie text and last 96 of the ‘your_cookie’ text can be combined to give 128 character ECB encrypted text of P_req without the need for the key to encrypt. Passing this to the server, the flag was found.
- Therefore there is a need to scramble characters before encrypting.

RSA (QUESTION 5) (CTF username – kssaivineeth15)

- To begin breaking RSA algorithm lets start with an example that introduces us to required formulae



Example [edit]

Here is an example of RSA encryption and decryption. The parameters used here are artificially small, but one can also use OpenSSL to generate and examine a real keypair.

1. Choose two distinct prime numbers, such as
   $p = 61$ and $q = 53$
2. Compute $n = pq$ giving
   $n = 61 \times 53 = 3233$
3. Compute the totient of the product as $\lambda(n) = \text{lcm}(p - 1, q - 1)$ giving
   $\lambda(3233) = \text{lcm}(60, 52) = 780$
4. Choose any number $1 < e < 780$ that is coprime to 780. Choosing a prime number for e leaves us only to check that e is not a divisor of 780.
   Let $e = 17$
5. Compute d, the modular multiplicative inverse of e (mod $\lambda(n)$) yielding,
   $d = 413$
   Worked example for the modular multiplicative inverse:
   $d \times e \bmod \lambda(n) = 1$
   $413 \times 17 \bmod 780 = 1$

The **public key** is ($n = 3233$, $e = 17$). For a padded plaintext message m, the encryption function is

$c(m) = m^{17} \bmod 3233$

The **private key** is ($n = 3233$, $d = 413$). For an encrypted ciphertext c, the decryption function is

$m(c) = c^{413} \bmod 3233$

For instance, in order to encrypt $m = 65$, we calculate

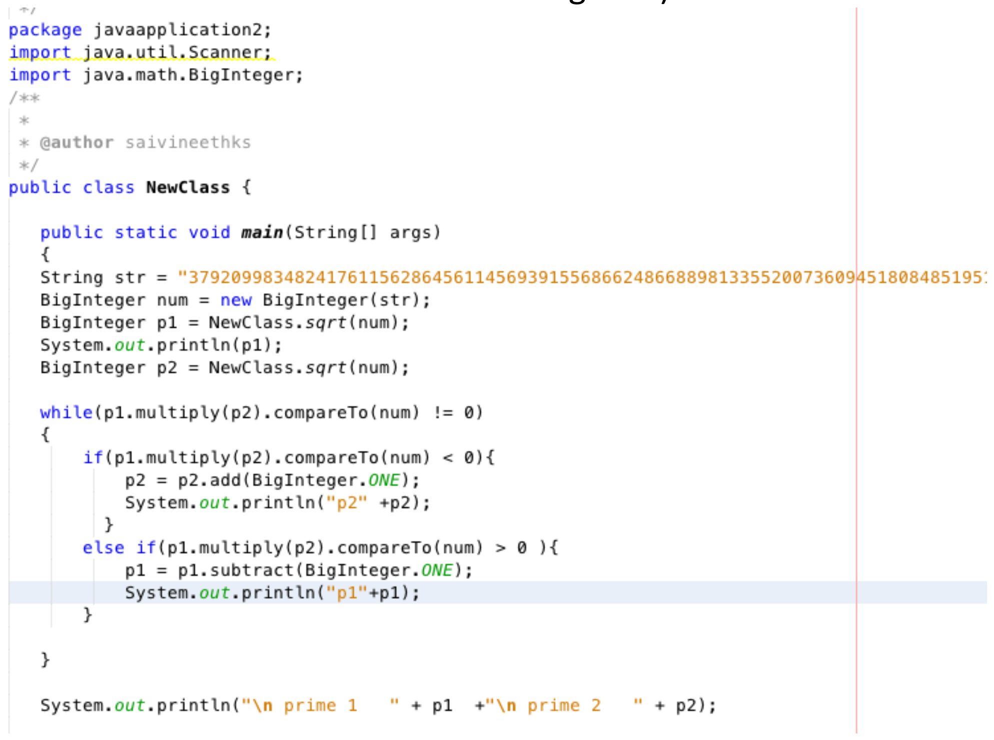
$c = 65^{17} \bmod 3233 = 2790$

To decrypt $c = 2790$, we calculate

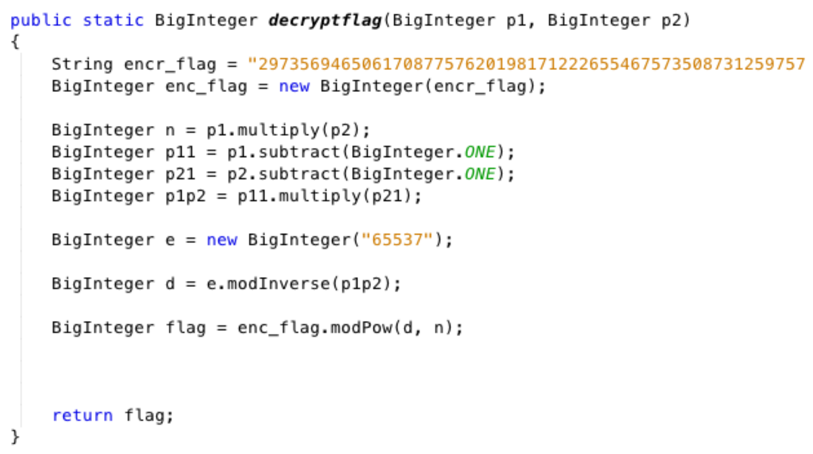
$m = 2790^{413} \bmod 3233 = 65$

- Here we can see that two prime numbers p and q are chosen to calculate n. e is chosen in random. The public key is given by (e,n).
- However, if the primes are adjacent then it would be easier to calculate them because both primes have to be close to the SQUARE ROOT OF THE value of ‘n’.
- Exploiting this concept a Java program has been written using BigInteger function that can find out the squareroot of n. The concept is referenced from https://stackoverflow.com/a/16804098 answer on stackoverflow.
- Later the primes p1 and p2 are equated to sqrt(n) which are then changed to modify such that p1 * p2 will equal to n.
- The p1 and p2 are changed in such a way that, while the multiplied value compared to actual n is lesser, p2 value is raised by 1 and if its greater p1 value is decreased by one.
- It kinda resembles moving away from sqrt(n) such that p2 moves farther away in increasing manner and p1 moves farther away in decreasing manner. Since the question says the values are primes, whatever values of p1 and p2 breaks out of the while loop, will be primes for sure.
- Now with values of p1 and p2 we can find (p1 -1) and (p2-1) that will help us finding ‘d’ which is the private key required for decryption. The value of ‘d’ is figured out using BigInteger class functions like modinv (Mod Inverse)
- Now with the value of ‘d’ following the modpow(mod power) function of BigInteger class. The flag was decrypted.

- The figures below show the program that was written to implement the above concept.(Includes only main and decryptflag functions. Sqrt function can be checked from the link given.)

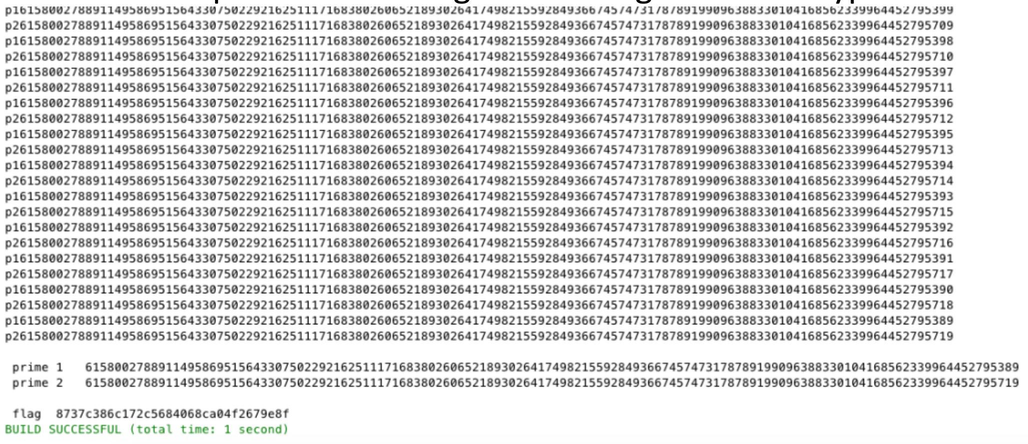


```
 */
package javaapplication2;
import java.util.Scanner;
import java.math.BigInteger;
/**
 *
 * @author saivineethks
 */
public class NewClass {

    public static void main(String[] args)
    {
    String str = "3792099834824176115628645611456939155686624866889813355200736094518084851195
    BigInteger num = new BigInteger(str);
    BigInteger p1 = NewClass.sqrt(num);
    System.out.println(p1);
    BigInteger p2 = NewClass.sqrt(num);

    while(p1.multiply(p2).compareTo(num) != 0)
    {
        if(p1.multiply(p2).compareTo(num) < 0){
            p2 = p2.add(BigInteger.ONE);
            System.out.println("p2" +p2);
          }
        else if(p1.multiply(p2).compareTo(num) > 0 ){
            p1 = p1.subtract(BigInteger.ONE);
            System.out.println("p1"+p1);
        }

    }

    System.out.println("\n prime 1   " + p1  +"\n prime 2   " + p2);
```



```
public static BigInteger decryptflag(BigInteger p1, BigInteger p2)
{
    String encr_flag = "29735694650617087757620198171222655467573508731259757
    BigInteger enc_flag = new BigInteger(encr_flag);

    BigInteger n = p1.multiply(p2);
    BigInteger p11 = p1.subtract(BigInteger.ONE);
    BigInteger p21 = p2.subtract(BigInteger.ONE);
    BigInteger p1p2 = p11.multiply(p21);

    BigInteger e = new BigInteger("65537");

    BigInteger d = e.modInverse(p1p2);

    BigInteger flag = enc_flag.modPow(d, n);



    return flag;
}
```

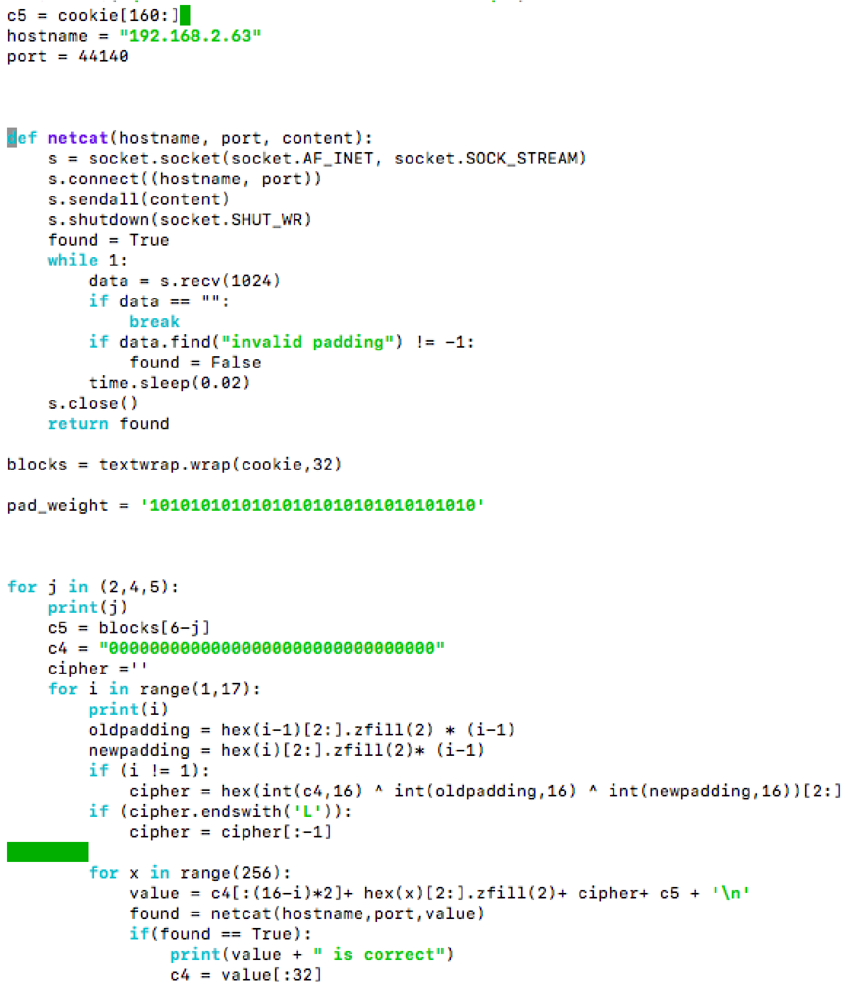
- The output of the code gives the flag that is decrypted.



```
p161580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795399
p261580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795709
p161580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795398
p261580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795710
p161580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795397
p261580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795711
p161580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795396
p261580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795712
p161580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795395
p261580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795713
p161580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795394
p261580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795714
p161580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795393
p261580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795715
p161580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795392
p261580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795716
p161580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795391
p261580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795717
p161580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795390
p261580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795718
p161580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795389
p261580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795719

 prime 1   61580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795389
 prime 2   61580027889114958695156433075022921625111716838026065218930264174982155928493667457473178789199096388330104168562339964452795719

 flag  8737c386c172c5684068ca04f2679e8f
BUILD SUCCESSFUL (total time: 1 second)
```

PADDING ATTACK (QUESTION 6) (CTF username – kssaivineeth15)

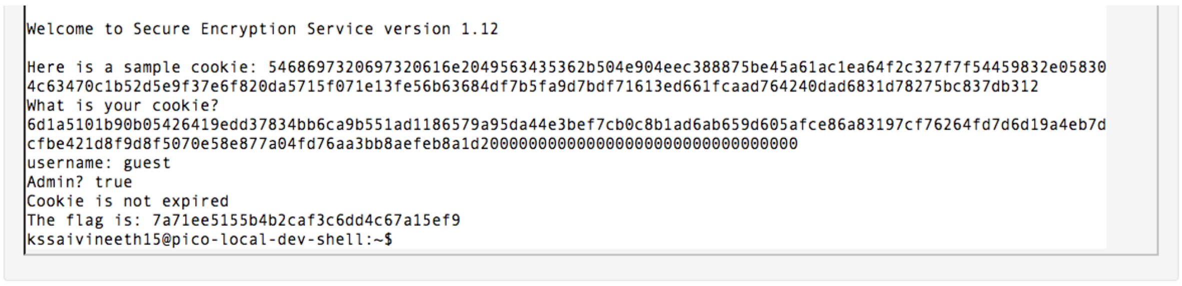
- This has been the most favourite of all questions given.
- For padding attack I used Python to run a socket program that can return the data from the server.
- The padding attack involves breaking a cipher block (C5) by using previous block C4. However we can force P5 to match a padding with 0x01 and then find corresponding C4_prime that can give 'valid padding' from the server.
- The code allowed me decrypt is given below



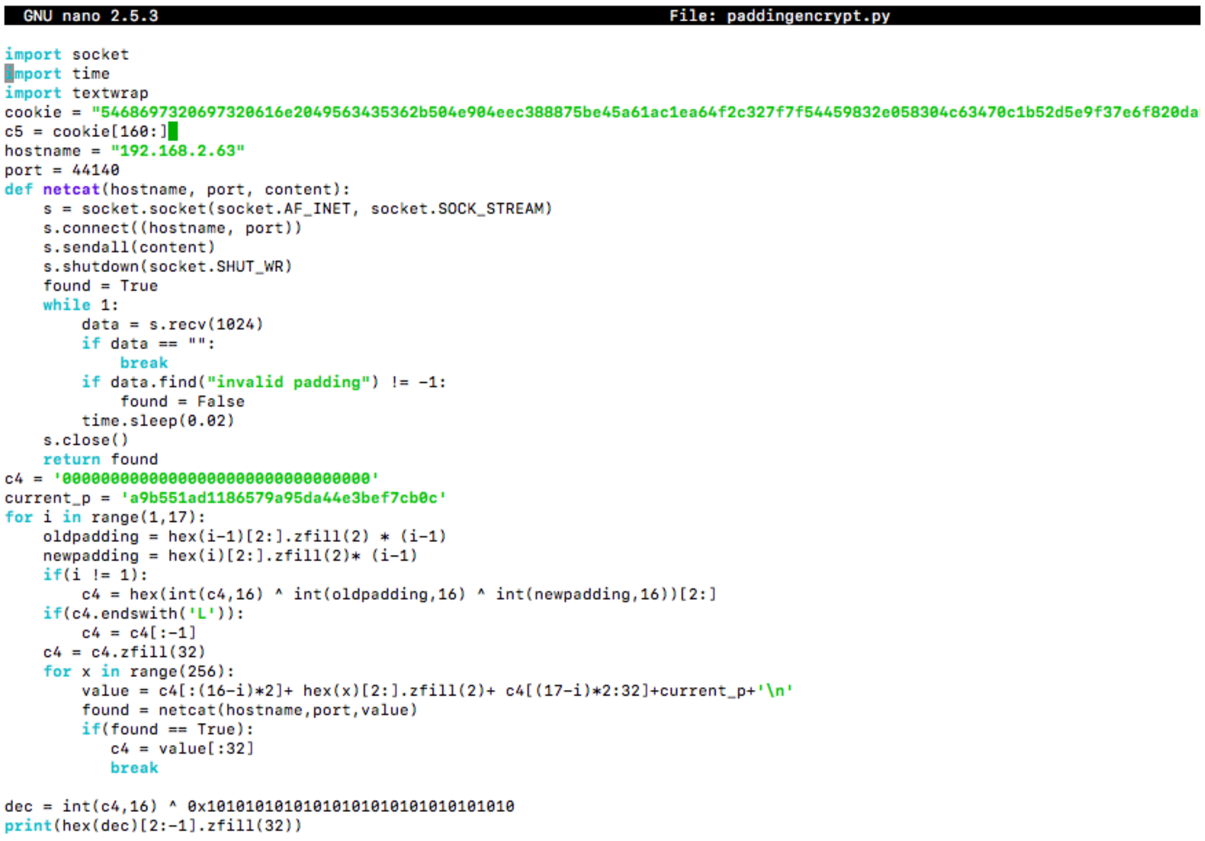
```
c5 = cookie[160:]
hostname = "192.168.2.63"
port = 44140


def netcat(hostname, port, content):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((hostname, port))
    s.sendall(content)
    s.shutdown(socket.SHUT_WR)
    found = True
    while 1:
        data = s.recv(1024)
        if data == "":
            break
        if data.find("invalid padding") != -1:
            found = False
        time.sleep(0.02)
    s.close()
    return found

blocks = textwrap.wrap(cookie,32)

pad_weight = '10101010101010101010101010101010'


for j in (2,4,5):
    print(j)
    c5 = blocks[6-j]
    c4 = "00000000000000000000000000000000"
    cipher =''
    for i in range(1,17):
        print(i)
        oldpadding = hex(i-1)[2:].zfill(2) * (i-1)
        newpadding = hex(i)[2:].zfill(2)* (i-1)
        if (i != 1):
            cipher = hex(int(c4,16) ^ int(oldpadding,16) ^ int(newpadding,16))[2:]
        if (cipher.endswith('L')):
            cipher = cipher[:-1]

        for x in range(256):
            value = c4[:(16-i)*2]+ hex(x)[2:].zfill(2)+ cipher+ c5 + '\n'
            found = netcat(hostname,port,value)
            if(found == True):
                print(value + " is correct")
                c4 = value[:32]
```

- I used the function netcat for socketing into the IP of the padding attack question. I used time to give some time for the server to respond. Based on the response the corresponding value from 'x loop' will be considered for next stage of processing. This way I created a loop that could crack C(n-1)_prime for each C(n). Now the C(n-1)_prime is XORed with '0x10101010101010101010101010101010'(16 '0x10'combination to get actual Dec(C(n)). The Dec(C(n)) is later XORed with the actual C(n-1) to get the plain text for the particular block.

- A similar concept has been followed for encryption. During decryption the plain text was found out to be.
    - {“username” : “guest”, “expires”: “2000-01-07”, “is_admin” : “false”} – decrypted message
    - Therefore the required plain text would be: {“username” : “guest”, “expires”: “2020-01-07”, “is_admin” : “true”}.
- The plain_text_req is hexed to give P_new (plain text new). This plain text was 160 words long (80 bytes). plain_hex = '7b22757365726e616d65223a202267756573742c2022657870697265 73223a2022323031392d30312d3037222c202269735f61646d696e223a2 02274727565227d0e0e0e0e0e0e0e0e0e0e0e0e0e0e'. ‘0x0e’ represents the padding required to complete them into multiple of 16 byte blocks.
- This plaintext was re encrypted using the algorithm given below. Initially it was giving me erros for many days. After debugging I realised that using Hex function eliminates the ‘zeroes’ that appear in the beginning. Therefore I used *zfill* function to fill up the eliminated zeroes and then successfully I got the flag. I would like to thank professor ,TAs and class mates for helping me understand the high level concept of encryption algorithm.



```
Welcome to Secure Encryption Service version 1.12

Here is a sample cookie: 5468697320697320616e2049563435362b504e904eec388875be45a61ac1ea64f2c327f7f54459832e05830
4c63470c1b52d5e9f37e6f820da5715f071e13fe56b63684df7b5fa9d7bdf71613ed661fcaad764240dad6831d78275bc837db312
What is your cookie?
6d1a5101b90b05426419edd37834bb6ca9b551ad1186579a95da44e3bef7cb0c8b1ad6ab659d605afce86a83197cf76264fd7d6d19a4eb7d
cfbe421d8f9d8f5070e58e877a04fd76aa3bb8aefeb8a1d200000000000000000000000000000000
username: guest
Admin? true
Cookie is not expired
The flag is: 7a71ee5155b4b2caf3c6dd4c67a15ef9
kssaivineeth15@pico-local-dev-shell:~$
```



```
 GNU nano 2.5.3                                   File: paddingencrypt.py

import socket
import time
import textwrap
cookie = "5468697320697320616e2049563435362b504e904eec388875be45a61ac1ea64f2c327f7f54459832e058304c63470c1b52d5e9f37e6f820da
c5 = cookie[160:]
hostname = "192.168.2.63"
port = 44140
def netcat(hostname, port, content):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((hostname, port))
    s.sendall(content)
    s.shutdown(socket.SHUT_WR)
    found = True
    while 1:
        data = s.recv(1024)
        if data == "":
            break
        if data.find("invalid padding") != -1:
            found = False
        time.sleep(0.02)
    s.close()
    return found
c4 = '00000000000000000000000000000000'
current_p = 'a9b551ad1186579a95da44e3bef7cb0c'
for i in range(1,17):
    oldpadding = hex(i-1)[2:].zfill(2) * (i-1)
    newpadding = hex(i)[2:].zfill(2)* (i-1)
    if(i != 1):
        c4 = hex(int(c4,16) ^ int(oldpadding,16) ^ int(newpadding,16))[2:]
    if(c4.endswith('L')):
        c4 = c4[:-1]
    c4 = c4.zfill(32)
    for x in range(256):
        value = c4[:(16-i)*2]+ hex(x)[2:].zfill(2)+ c4[(17-i)*2:32]+current_p+'\n'
        found = netcat(hostname,port,value)
        if(found == True):
           c4 = value[:32]
           break

dec = int(c4,16) ^ 0x10101010101010101010101010101010
print(hex(dec)[2:-1].zfill(32))
```

STRIDE ANALYSIS (QUESTION 7) (CTF username – kssaivineeth15)

- We can consider that the data from the power meter reaches multiple routers and gateways to reach the internet and later reaches to its server and saves the data.
- Performing STRIDE analysis we can find that
- SPOOFING ATTACKS:
  - It is a situation in which any adversary or program can impersonate another person on the network by falsifying data for to gain advantage.
  - In this situation the system has to have an AUTHENTICATION method to validate the identity of the user
  - Lack of it , any MITM (man in the middle) attack would be possible where the adversary can sniff for packets in the network and perform TCP/IP spoofing . The attacker can change the identity of the user on the packet and allow the packet to reach in the name of a different user. This would be a huge problem for the company to deal. The attacker can compromise identity with this method.
  - DNS spoofing is also possible, where the adversary (MITM) can spoof the DNS of the power company server to one of his own servers and collect the data, preventing the proper functioning of the power company. He can also attack the customer of the company by pretending to send bill information from the power company server, but connect the payment gateway to his/her own account.
  - For these cases packet filtering can be implemented that can be scanned for inconsistencies and filtered accordingly.
- TAMPERING ATTACKS:
  - Parameter tampering is possible, where the data of the packets can be tampered. The attacker feeds the packet with his own message(something like an increased power meter value). The power company server is deceived into thinking the contents of the message are authentic and bill the person for the spoofed meter readings data instead of the original/actual one.
  - Attacker would have the power to modify data including its destination address. So he can change the destination address to his own server and collect all the meter readings, which does not allow the packets to never the power company server.

- REPUDIATION
  - Repudiation stands for not being able to trace back information to its origin. The system should support non-repudiation and therefore cannot allow any user to get unbilled for a higher usage of power. *non-repudiation* involves associating actions or changes with a unique individual.
  - Phishing attack by an adversary on the meter reading device can create problems to the digital security of the system.
  - Non Repudiation is attacked if the user authentication details or being shared or stolen. Any adversary with this information can spoof the identity.
  - The meter reading area in the customer’s house should have the ability to trace every data it gets back to its origin. Else any adversary can spoof data into the machine.
  - Hashing maintains data integrity and origin of data can be maintained by using a PUBLIC key infrastructure. In which the data is always authenticated and can be traced back to its origin
  - Trusted third parties can be called for to authenticate public keys, failure to which any adversary can send his own public key to the customer and allow the customer to encrypt messages to the adversary(wrong destination).
  - Digital Signatures can be implemented, so that the user cannot repudiate on the power consumed by him. It gives a guaranteed claim that the user was using it.
  - Strict policies have to be implemented in case of stolen passwords or corrupt trusted parties.
- INFORMATION PRIVACY
  - By sniffing for packets on the unsecure channel between server and the customer area, the attacker can get hold of some personal data like name, bank account number, house address, date of birth, mobile number etc.
  - This information proves costly for the user. As any of this information can be used for indentity theft. End to End encryption would be required to counter this.
  - Even knowing the power consumption behaviour of the user, can be used to infer what appliances the customer is using. For example, an Air conditioner has a bigger power profile than a Fan or an LED.

    - Data breach is the biggest security concern of any large scale company.
    - If the servers of the power company are unsecure, then the attacker can collect all the users data which would be very valuable for the dark market
- DENIAL OF SERVICE
    - Denial of service is a cyber attack where the adversary makes a server unavailable for its users by FLOODING the target machine or resource with superfluous requests.
    - This is mostly done to distract the security personel of a company, while the attacker performs a data breach or any other attack on other servers of the company.
    - In our scenario, the attacker can perform DoS attack on the power company server, which prevents the packets from the customer side reach the company server, some important data can be lost in this process.
    - The DoS attack can be performed by hacking one or many of the power meters in the network and allowing them to flood data packets to the main server of power company.
- ELEVATION OF PRIVILEGE
    - Elevation of privilege involves exploiting a bug or a loophole in the operating system and gain higher access.
    - The attacker can perform jailbreak attack or find out a loophole in the power company main server and escalate to an administrator privilege.
    - As an admin, the attacker can include or remove users, modify meter readings data, allow the server to shut down, practically every kind of malicious activity is possible once privilege escalations happen.
    - If the user himself is an adversary he can escalate his privilege in the network and cause harm to other users in the network.
    - To prevent this, proper access control policies should be followed,
    - cross site scripting allows admin access in some web based servers.