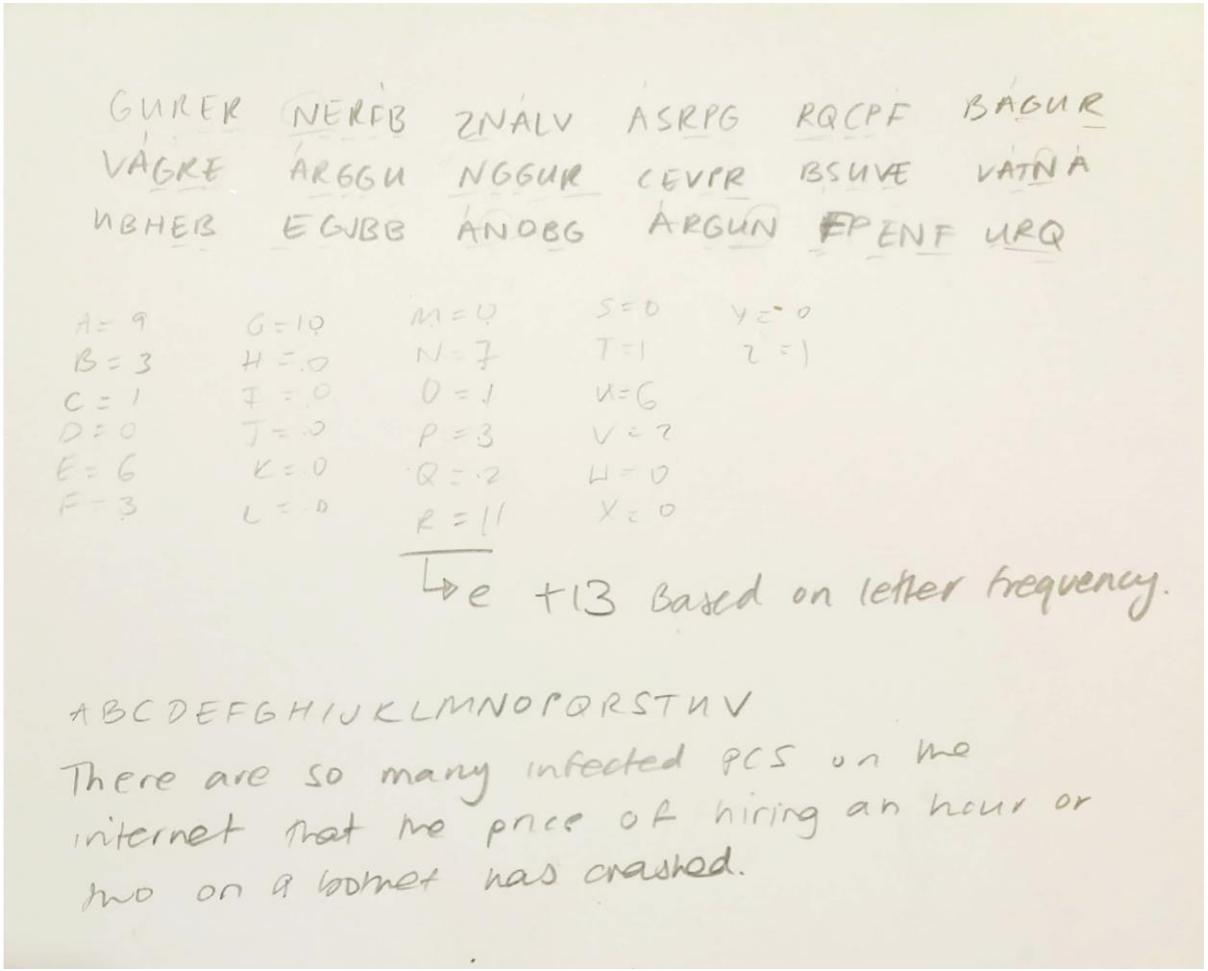
CYBR 171
ASSIGNMENT ONE
300381661

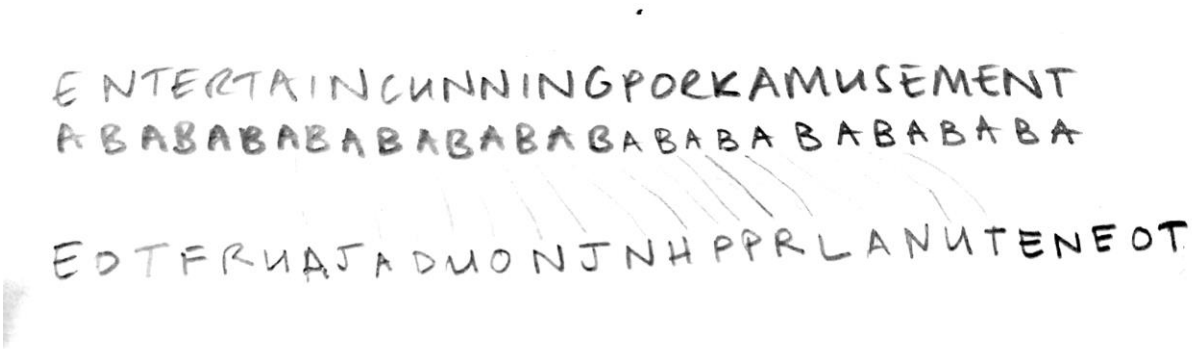
CORE
Cipher Algorithms – Encryption and Decryption

QUESTION ONE



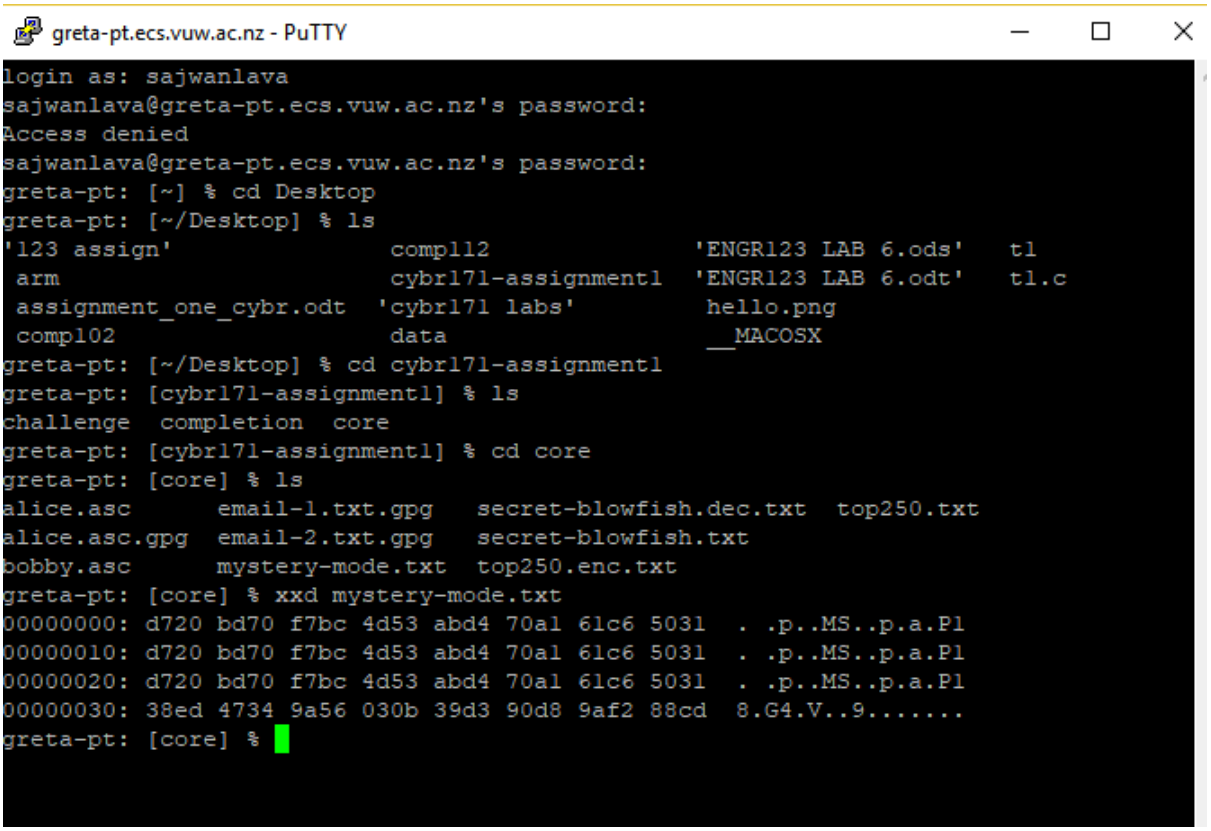
THERE ARE SO MANY INFECTED PCS ON THE INTERNET THAT THE PRICE OF HIRING AN HOUR OR TWO ON A BOTNET HAS CRASHED

QUESTION TWO



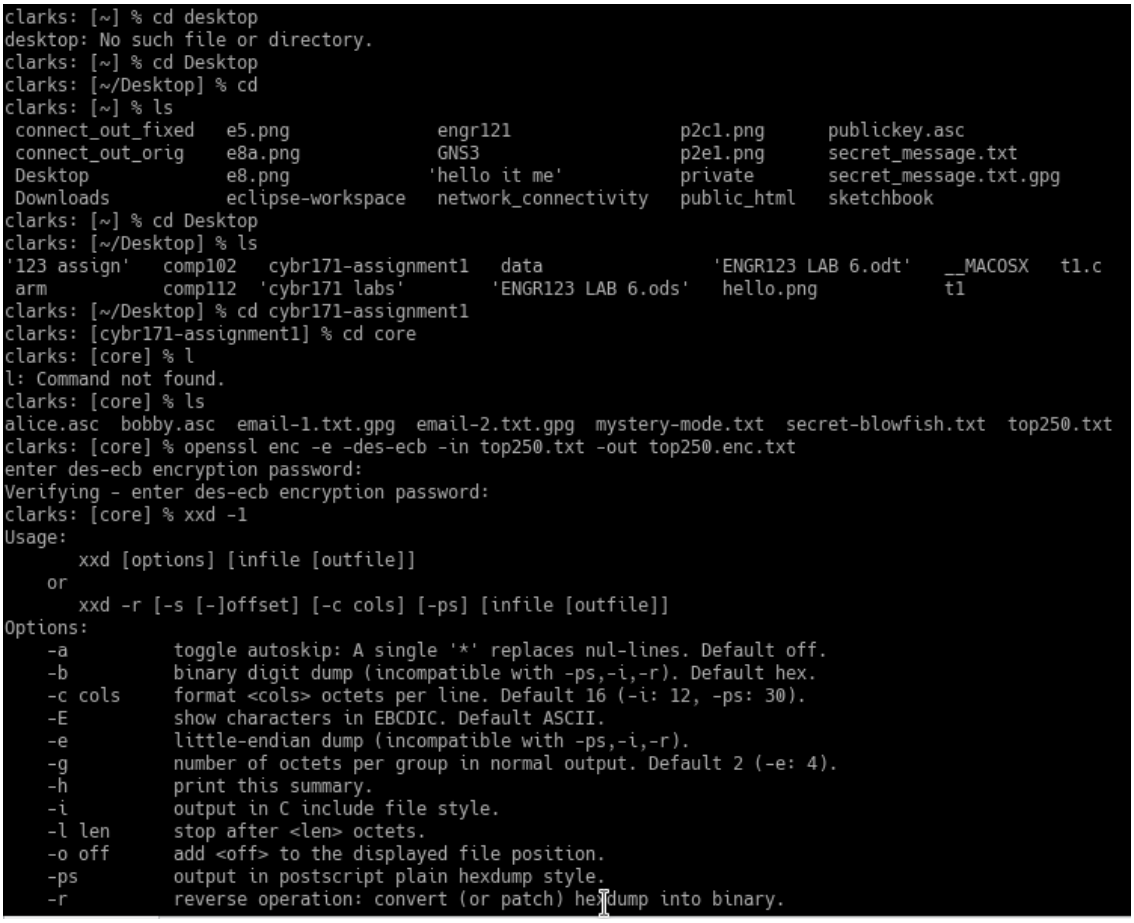
QUESTION THREE

xxs mystery-mode.txt



As the encrypted blocks form a repeated pattern, this indicates that it is Electronic Code Book (ECB).

QUESTION FOUR



```
clarks: [core] % openssl enc -e -des-ecb -in top250.txt -out top250.enc.txt
enter des-ecb encryption password:
Verifying - enter des-ecb encryption password:
clarks: [core] % xxd -1
Usage:
       xxd [options] [infile [outfile]]
    or
       xxd -r [-s [-]offset] [-c cols] [-ps] [infile [outfile]]
Options:
    -a          toggle autoskip: A single '*' replaces nul-lines. Default off.
    -b          binary digit dump (incompatible with -ps,-i,-r). Default hex.
    -c cols     format <cols> octets per line. Default 16 (-i: 12, -ps: 30).
    -E          show characters in EBCDIC. Default ASCII.
    -e          little-endian dump (incompatible with -ps,-i,-r).
    -g          number of octets per group in normal output. Default 2 (-e: 4).
    -h          print this summary.
    -i          output in C include file style.
    -l len      stop after <len> octets.
    -o off      add <off> to the displayed file position.
    -ps         output in postscript plain hexdump style.
    -r          reverse operation: convert (or patch) hexdump into binary.
    -r -s off   revert with <off> added to file positions found in hexdump.
    -s [+][-]seek  start at <seek> bytes abs. (or +: rel.) infile offset.
    -u          use upper case hex letters.
    -v          show version: "xxd V1.10 27oct98 by Juergen Weigert".
clarks: [core] % xxd -l 10




^C
clarks: [core] %
clarks: [core] % ls
alice.asc  email-1.txt.gpg  mystery-mode.txt     top250.enc.txt
bobby.asc  email-2.txt.gpg  secret-blowfish.txt  top250.txt
clarks: [core] % xxd -l 10 top250.enc.txt
00000000: 5361 6c74 6564 5f5f a876                Salted__.v
clarks: [core] % ~
```

```
       xxd [options] [infile [outfile]]
    or
       xxd -r [-s [-]offset] [-c cols] [-ps] [infile [outfile]]
Options:
    -a          toggle autoskip: A single '*' replaces nul-lines. Default off.
    -b          binary digit dump (incompatible with -ps,-i,-r). Default hex.
    -c cols     format <cols> octets per line. Default 16 (-i: 12, -ps: 30).
    -E          show characters in EBCDIC. Default ASCII.
    -e          little-endian dump (incompatible with -ps,-i,-r).
    -g          number of octets per group in normal output. Default 2 (-e: 4).
    -h          print this summary.
    -i          output in C include file style.
    -l len      stop after <len> octets.
    -o off      add <off> to the displayed file position.
    -ps         output in postscript plain hexdump style.
    -r          reverse operation: convert (or patch) hexdump into binary.
    -r -s off   revert with <off> added to file positions found in hexdump.
    -s [+][-]seek  start at <seek> bytes abs. (or +: rel.) infile offset.
    -u          use upper case hex letters.
    -v          show version: "xxd V1.10 27oct98 by Juergen Weigert".
clarks: [core] % xxd -l 10




^C
clarks: [core] %
clarks: [core] % ls
alice.asc  email-1.txt.gpg  mystery-mode.txt     top250.enc.txt
bobby.asc  email-2.txt.gpg  secret-blowfish.txt  top250.txt
clarks: [core] % xxd -l 10 top250.enc.txt
00000000: 5361 6c74 6564 5f5f a876                Salted__.v
clarks: [core] % openssl enc -des-ecb -nosalt -in top250.txt -out top250.enc.txt
enter des-ecb encryption password:
Verifying - enter des-ecb encryption password:
clarks: [core] % xxd -l 10 top250.enc.txt
00000000: 222c 38a6 b28b 8f75 e2f0                ",8....u..
clarks: [core] %
```

QUESTION FIVE

```
    -b          binary digit dump (incompatible with -ps,-i,-r). Default hex.
    -c cols     format <cols> octets per line. Default 16 (-i: 12, -ps: 30).
    -E          show characters in EBCDIC. Default ASCII.
    -e          little-endian dump (incompatible with -ps,-i,-r).
    -g          number of octets per group in normal output. Default 2 (-e: 4).
    -h          print this summary.
    -i          output in C include file style.
    -l len      stop after <len> octets.
    -o off      add <off> to the displayed file position.
    -ps         output in postscript plain hexdump style.
    -r          reverse operation: convert (or patch) hexdump into binary.
    -r -s off   revert with <off> added to file positions found in hexdump.
    -s [+][-]seek  start at <seek> bytes abs. (or +: rel.) infile offset.
    -u          use upper case hex letters.
    -v          show version: "xxd V1.10 27oct98 by Juergen Weigert".
clarks: [core] % xxd -l 10




^C
clarks: [core] %
clarks: [core] % ls
alice.asc  email-1.txt.gpg  mystery-mode.txt     top250.enc.txt
bobby.asc  email-2.txt.gpg  secret-blowfish.txt  top250.txt
clarks: [core] % xxd -l 10 top250.enc.txt
00000000: 5361 6c74 6564 5f5f a876                Salted__.v
clarks: [core] % openssl enc -des-ecb -nosalt -in top250.txt -out top250.enc.txt
enter des-ecb encryption password:
Verifying - enter des-ecb encryption password:
clarks: [core] % xxd -l 10 top250.enc.txt
00000000: 222c 38a6 b28b 8f75 e2f0                ",8....u..
clarks: [core] % openssl enc -aes-128-ecb -nosalt -in top250.txt -out top250.enc.txt
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
clarks: [core] % xxd -l 10 top250.enc.txt
00000000: aa62 410e ed2a 8023 a097                .bA..*.#..
clarks: [core] %
```

QUESTION SIX

```
^C
clarks: [core] %
clarks: [core] % ls
alice.asc  email-1.txt.gpg  mystery-mode.txt     top250.enc.txt
bobby.asc  email-2.txt.gpg  secret-blowfish.txt  top250.txt
clarks: [core] % xxd -l 10 top250.enc.txt
00000000: 5361 6c74 6564 5f5f a876                Salted__.v
clarks: [core] % openssl enc -des-ecb -nosalt -in top250.txt -out top250.enc.txt
enter des-ecb encryption password:
Verifying - enter des-ecb encryption password:
clarks: [core] % xxd -l 10 top250.enc.txt
00000000: 222c 38a6 b28b 8f75 e2f0                ",8....u..
clarks: [core] % openssl enc -aes-128-ecb -nosalt -in top250.txt -out top250.enc.txt
enter aes-128-ecb encryption password:
Verifying - enter aes-128-ecb encryption password:
clarks: [core] % xxd -l 10 top250.enc.txt
00000000: aa62 410e ed2a 8023 a097                .bA..*.#..
clarks: [core] % openssl enc -d-bf -cbc -in top250.txt -out top250.dec.txt
enc: Unknown cipher d-bf
enc: Use -help for summary.
clarks: [core] % openssl enc -d -bf -cbc -in top250.txt -out top250.dec.txt
enc: Unknown cipher cbc
enc: Use -help for summary.
clarks: [core] % openssl enc -d -bf -cbc -in secret-blowfish.txt -out secret-blowfish.dec.txt
enc: Unknown cipher cbc
enc: Use -help for summary.
clarks: [core] % openssl enc -d -bf-cbc -in secret-blowfish.txt -out secret-blowfish.dec.txt
enter bf-cbc decryption password:
clarks: [core] % YELLOW
YELLOW: Command not found.
clarks: [core] % cat secret-blowfish.dec.txt
You have the right answer.
clarks: [core] %
```

QUESTION SEVEN

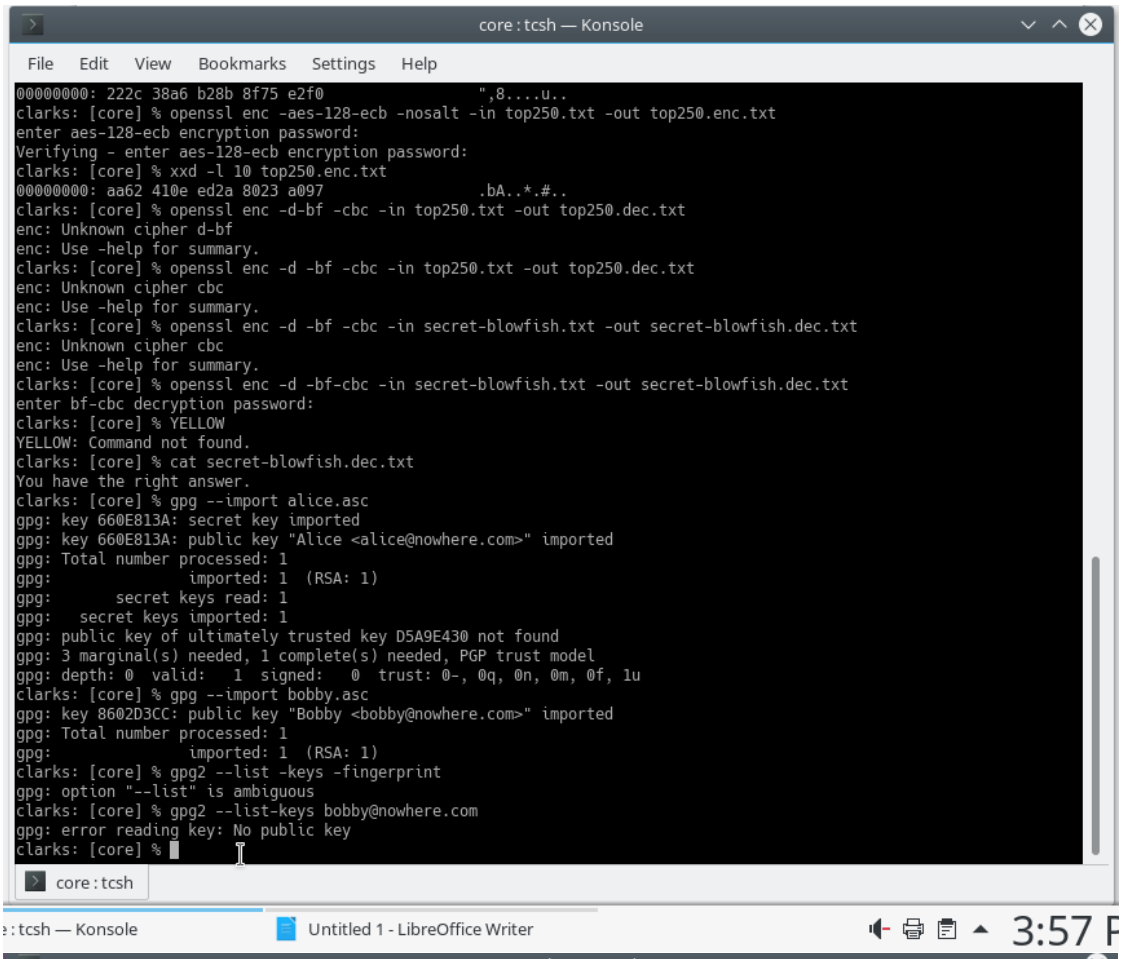
Carol would swap her encrypted pay roll for Bob's. By doing so she breaches the integrity of the system as data has been modified without permission and is therefore now inaccurate.

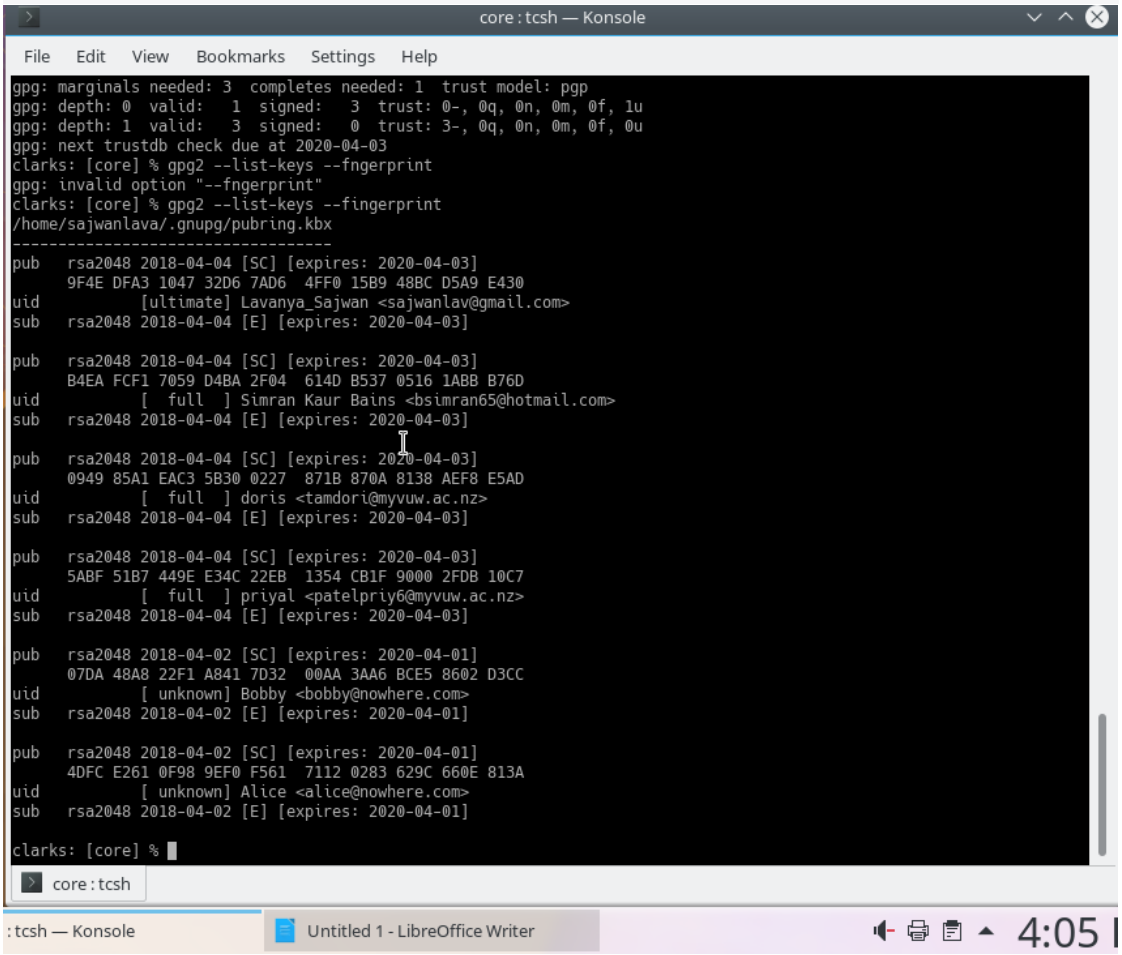
QUESTION EIGHT

A way to prevent the "cut-and-paste" attack would be to make it so that every team member has a different cipher specific to them. That way when Carol tampers with the ciphers, it would be apparent. Another way to prevent this would be to change from ECB to Cipher Block Chaining (CBC), as it is harder for an attacker to modify and it can be more apparent that the integrity has been lost. This is because CBC doesn't process each block separately. This would thus change the information on the file.

Another way would be to disable the clipboard on that file so she wouldn't be able to copy and paste. This wouldn't be done directly on the file itself, so no extra information would be stored.

QUESTION NINE





Gpg2 –import alice.asc

gpg2 –import bobby.asc (initially accidentally used gpg for both)

QUESTION TEN

Bobby had signed Alice's key and this was found using the command gpg2 –check-sig (last 8 digits).

```
clarks: [core] % gpg2 --check-sig 660E 813A
gpg: error reading key: No public key
clarks: [core] % gpg2 --check-sig 660E 813A
gpg: error reading key: No public key
clarks: [core] % gpg2 --check-sig 660E813A
pub   rsa2048 2018-04-02 [SC] [expires: 2020-04-01]
      4DFCE2610F989EF0F56171120283629C660E813A
uid           [ unknown] Alice <alice@nowhere.com>
sig!3        0283629C660E813A 2018-04-02  Alice <alice@nowhere.com>
sig!         3AA6BCE58602D3CC 2018-04-02  Bobby <bobby@nowhere.com>
sub   rsa2048 2018-04-02 [E] [expires: 2020-04-01]
sig!         0283629C660E813A 2018-04-02  Alice <alice@nowhere.com>

gpg: 3 good signatures
clarks: [core] %
```

QUESTION ELEVEN

```
greta-pt: [core] % gpg2 --edit-key Lavanya_Sajwan
gpg (GnuPG) 2.2.4; Copyright (C) 2017 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Secret key is available.

sec  rsa2048/15B948BCD5A9E430
     created: 2018-04-04  expires: 2020-04-03  usage: SC
     trust: ultimate      validity: ultimate
ssb  rsa2048/0C05696C48014620
     created: 2018-04-04  expires: 2020-04-03  usage: E
[ultimate] (1). Lavanya_Sajwan <sajwanlav@gmail.com>
```

```
greta-pt: [core] % gpg2 --sign-key 8602D3CC

gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   3  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: depth: 1  valid:   3  signed:   0  trust: 3-, 0q, 0n, 0m, 0f, 0u
gpg: next trustdb check due at 2020-04-03
pub  rsa2048/3AA6BCE58602D3CC
     created: 2018-04-02  expires: 2020-04-01  usage: SC
     trust: unknown       validity: unknown
sub  rsa2048/6478C44270160CEF
     created: 2018-04-02  expires: 2020-04-01  usage: E
[ unknown] (1). Bobby <bobby@nowhere.com>


pub  rsa2048/3AA6BCE58602D3CC
     created: 2018-04-02  expires: 2020-04-01  usage: SC
     trust: unknown       validity: unknown
 Primary key fingerprint: 07DA 48A8 22F1 A841 7D32  00AA 3AA6 BCE5 8602 D3CC

     Bobby <bobby@nowhere.com>

This key is due to expire on 2020-04-01.
Are you sure that you want to sign this key with your
key "Alice <alice@nowhere.com>" (0283629C660E813A)

Really sign? (y/N) y

greta-pt: [core] % gpg2 --check-sig 8602D3CC
pub   rsa2048 2018-04-02 [SC] [expires: 2020-04-01]
      07DA48A822F1A8417D3200AA3AA6BCE58602D3CC
uid           [ unknown] Bobby <bobby@nowhere.com>
sig!3        3AA6BCE58602D3CC 2018-04-02  Bobby <bobby@nowhere.com>
sig!         0283629C660E813A 2018-04-24  Alice <alice@nowhere.com>
sub   rsa2048 2018-04-02 [E] [expires: 2020-04-01]
sig!         3AA6BCE58602D3CC 2018-04-02  Bobby <bobby@nowhere.com>

gpg: 3 good signatures
greta-pt: [core] %
```

QUESTION TWELVE

```
greta-pt: [core] % gpg2 --decrypt email-1.txt.gpg
gpg: encrypted with 2048-bit RSA key, ID 21D059181C12C8CC, created 2018-04-02
      "Alice <alice@nowhere.com>"
Dear Alice,

Now I ask you one. What has a trunk, but no key, weighs 2,000 pounds and lives in
a cir\cus?

Regards, Bob
greta-pt: [core] %
```

QUESTION THIRTEEN

```
greta-pt: [core] % gpg2 --decrypt email-2.txt.gpg
gpg: encrypted with 2048-bit RSA key, ID 21D059181C12C8CC, created 2018-04-02
      "Alice <alice@nowhere.com>"
Dear Alice,

I am on holiday in Freedonia but have had all my cash stolen.

Please send me your credit card details immediately otherwise I cannot pay for my
accommodation.

Send it to needmoneyinfreedonia@gmail.com.

Regards, Bob
gpg: Signature made Mon 02 Apr 2018 21:37:32 NZST
gpg:                using RSA key 29820923C5883AC384D8E1245EDAECFE3356332A
gpg:                issuer "carol@nowhere.com"
gpg: Can't check signature: No public key
greta-pt: [core] %
```

This output does indicate an attempted attack because the emails listed haven't previously been used by Bob, so they should not be a trusted email. Also there has been no indication in previous emails that Bob may be travelling to "Freedonia", and Alice should know never to send her credit card details online no matter whom it is. The nature of the attack of this phishing email is that of an active attack as the sender whom has stolen the encryption key is directly trying to obtain Alice's personal details so that they can harm her financially (*Nigerian Scams (n.d*)). Bob and Alice will now have to change their key as they may not know whether it is actually them trying to talk to each other.

COMPLETION

Password Strength Meters and Cryptanalysis

QUESTION ONE

- abc123
  bennish.net = 0.001 seconds
  howsecureismypassword.net = instantly
  passwordmeter.com = weak
- trustno1
  bennish.net = 0.001 seconds
  howsecureismypassword.net = instantly
  passwordmeter.com = weak
- ncc1701
  bennish.net = 0.006

howsecureismypassword.net = instantly
passwordmeter.com = good

- iloveyou!
  bennish.net = 0.016
  howsecureismypassword.net = instantly
  passwordmeter.com = weak
- primtime21
  bennish.net = 4 hours
  howsecureismypassword.net = 1 day
  passwordmeter.com = weak

Most passwords were cracked instantly; only the last one took a little longer, but was still considered a "weak" password. There were inconsistencies between the websites for some of the passwords. This was shown with ncc1701, which was cracked instantly with bennish.net and howsecureismypassword.net, however, passwordmeter.com still stated that it was good with a 46% score. This was probably due to the websites having slightly different specifications of what made a strong password. Also in terms of the actual passwords, there were slight differences in speed between some of them due to the latter passwords using a combination of non-consecutive numbers or symbols.

QUESTION TWO

As we were given that the key length was four, I split the contents of the vigenere.txt into blocks of four (Seidel, 2014).

WIEV HSMY RSMC VBPR OJEF WPCQ PQUV HSSE DNEH UPML RIND UVNP HSSU FJEP FFFK FUIQ

QDLC VTIE WIEU KPCM ZBVG UJDG UUHG QPVG OEEU FSID HEHQ ZBRG EFLE RNPW WFRR UPGT

DNMG UDRG DUEF DQRQ JSAO FBLN HETC SFWQ UNWJ LDHY DTRG OFAU HEIP WPAP RNNK SPTG

QUCQ PQUV HSNG WXOT NVSG GCYC QBUV RDRC WJCI RWET QNEP WUOE ROTT RMIV VQEQ SMEV

KFGQ YFRP PFNV KBDV RUUT QPFH WIEE RNPW WFRP HUWQ ULTJ XTDG VURQ BJNI LUSE ROTT

RMIP RSDG UUOG UBDK FBTG WIEY RSMD UVNP HSSD RPKK VBBQ XUAU FMOU HBSO RTTX PTCQ

PQUV HSNG WXOT NNAP DHET VXOW OEEX HSHC YFCQ PFTQ DSEC OSOI XFWQ UNUP WJLV KFLC

WFSY RSMU ZFRG RCSE XSEV KJNI VNOT HBSU RDIC WFDY LUHT HTEC UDHK QBCQ PQUV HSLC EPRC

WPRA IPRG ABMR OFAH HXBG QFVQ OFNV ZPRO VXET HEEX HMOR HEBA AFRQ ASEU HBRE KFRU

ZIOY DOTG GUOO DLEO RSEG IGIE LFNV XTEQ IDOO SVTG UGAE LMIV LFSV KFYF HWEN RQEF DUOY

QDRK HSWQ UNWJ LDHO RWEF WIRQ XHHC QFTY RSKU HODK QHOW WJMR RSTC QUAP QPUP

FFMG QUSV KFIT GJAI QPSV LDWQ UNAN VPCQ QTTC QULA ZFAX HETJ UPUI KUHG QFTY RSKD XUTJ

LT

Then for the 1st letters of the blocks, I found the most frequent occurring number. I then repeated this for the 2nd, 3rd and 4th letters. I then calculated how much their shift was from E (the most frequent occurring letter in the English language), and added one to it. Then I went up from A to get to what that letter originally was as shown in my working out. This then gave me the four letter key DBAC which I then used to decrypt the message which was;

"The term worms as applied to computers came from John Brunners science fiction classic the shock wave rider the novel described how warebel computer programmer created a program called tape worm which was released into an Omni potent computer network used by an autocratic government to control its people the government had to turn off the computer network thus destroying its control in order to eradicate the worm Brunners book is about as close as most VMS computer network managers would ever have to come to a real rogue worm until the lates worms were obscure things more associated with research in a computer laboratory for example few benevolent worms were developed by XEROC researchers wo wanted to make more efficient use of computer facilities they developed a town crier worm which moved through networks sending out important announcements their diagnostic worm also constantly weaved through the network but this"

1st symbol : Most common = H ⇒ D

2nd symbol : Most common = F ⇒ B

3rd symbol : Most common = E ⇒ A

4th Symbol : Most common = G ⇒ C

Key = DBAC

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

WIEV HSMY RSMC VBPR OJEP WPCQ PQUV HSSE
DBAC DBAC DBAC DBAC DBAC DBAC DBAC DBAC DBAC
THET ERMNORMAS APP LIED TOCO MPUT ERSC

DNEH UPML RIND UVNP TISSU FJEP FFIK FUID
DBAC DBAC DBAC DBAC DBAC DBAC DBAC DBAC
AMEF ROMJ OHNB RUNN ERSS CIENCEFICTIO

QDLC VTIE WIEU KPCM ZBVG UJDG UUHG
DBAC DBAC DBAC DBAC DBAC DBAC DBAC
NCLASSIC THESH OCK WAVE RIDE RTHE

QPVG OEEU PSID HEHQ ZBRG EFLERNPW
DBAC DBAC DBAC DBAC DBAC BBAC DBAC
WERRUPGT
NOVE LDES CRIBEDHO WARB ECOMPUTE

WFRRUPGTDNMG UDRG DUEF DQRQJSAO FBLN
DBACDBACDBAC DBAC DBAC DBAC DBAC DBAC
RPROGRAMMERECRE_ATED APROGRAM CALL

HETCSFWQUNWJLDHYDTRGOFAU HEIP WPAP
DBACDBAC DBAC DBAC DBAC DBAC DBAC DBAC
EDTAPEWORMWHIC HWASRELEAS EDINTDAN

RNNICSPTGQUGQ PQUV HSNGWXOTNVSG GCYC
DBACDBAC DBAC DBAC DBAC DBAC DBAC DBAC
DMNIPOTENTCO MPUT ERNE TWORKVSE DBYA
QBUVRDRC WJCI RWET QNEPWUOEROTT RMIVVQEQ
DBACDBAC DBAC DBAC DBAC DBAC DBAC DBAC
NAVTOCRA TICG OVERNMENTTOCONTR OLITSPED

SMEV KFGO VFRP PFNVK BDVRUUTQPFH WIEE
DBAC DBAC DBAC DBAC DBAC DBAC DBAC DBAC
PIET HEGO VERN MENTHADTOTURNCFT HECO

RNPH WFRPHUWQ ULTJ XTDG VURQ BJNI
DBAC DBACDBAC DBAC DBAC DBAC DBAC
MPUT ERNETWOR KSTH USDE STRO YINS
LUSEROTT RMIPRSDG UUOG VBDK FBTG WIEY
DBAC DBAC PBACDBAC DBAC DBAC DBAC
ITSC ONTR OLINORDER TOERADI CATETHEW

RSMDU VNP HSSD RPKRVBBQ XVAU EMOU HBSO
DBACDBAC DBAC DBAC DBAC DBAC DBAC
ORMBRUNNERS BOOKISABOUTASCLUSES
RTTXPCQPQUV HSNG WXOT NNAP DHETVXOWDETY
DBACDBACDBAC DBAC DBAC DBAC DBAC DBACDBAC
MOSTVMSCOMPU TERNETWORKMA NAGERSWOULDE

HSHC YFCQ PFTQ DSEC OSOL XFWQ UNUPWJLV
DBAC DBAC DBAC DBAC PBAC PBAC DBAC DBAC
VERHAVECOMETOREA LROB UEWO RMUNTIL
KFLC WFSY RSMU 2FRG RCBE XSEV KJNIVNOT
DBAC DBAC DBAC DBAC DBAC DBAC DBAC DBAC
THEL ATEW ORMS WERE OBSCURE THINGSMO

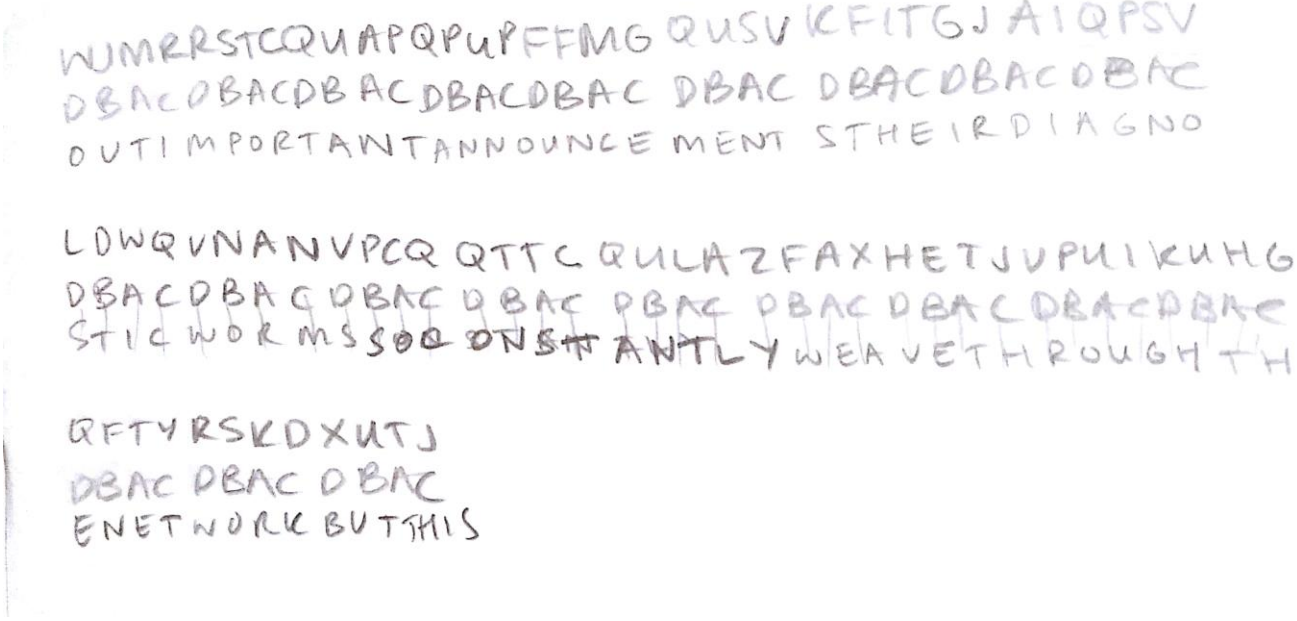
HBSURDIC WFDYLUHT HTECLIDHKQBCQ PQUVHSLC
DBACDBAC DBACDBAC DBAC DBAC DBACDBACDBAC
REASSOCI ATEDWITH RESEARCH INGACOMPUTER
EPRCWPRAIPRGABMR OFAH HXBGQ FUQ OFNV
DBAC DBACDBAC DBAC DBAC DBAC DBAC DBAC
LABORATORYFOREXA MPLE AFEWBENE VIOL

ZPRO VXET HEEXHIMORHEBAAFRQ HSEUHBREKFRU
DBAC DBAC DBACDBACDBACDBAC DBACDBACDBAC
ENTW ORMS WEREDEVELOPEDBYXEROXRESEARCH

ZIOY DOTG GUOODLEO RSEG IGIELFNV XTEQ IDOO
PBAC DBAC PBAC DBAC PBAC DBACDBAC DBAC
ESWH OWANTEDTOMAKEMORE EFFICIENTUSE OTHE

SVTGUGAECMIVLFSVKFYFHWENRQEFDUOYQDRK
DBAC DBACDBACDBACDBACDBAC DBACDBACDBAC
OMPUTERFACILITIESTHEYDEV ELOPED ATOWNC

HSWQVNWJLDHORWEFWIRQX##C QFTYRSKUHODKQHOW
DBACDBACDBACDBACDBACDBAC DBACDBACDBACDBAC
RIERNORMWHICHMOVEDTHROUGH NETWORKSSENDING



CHALLENGE

Capture the Flag and Secure Messaging

QUESTION ONE

I had to decrypt the message four times to which the end message was;
“This is a flag to prove that you did decrypt it.”
It isn’t very secure because it is very simple to decrypt with any online tool and so it is accessible to most unlike a message encrypted with a key which is only able to be decrypted with that specific key. I used the tool: https://www.base64decode.org/

QUESTION TWO

QUESTION THREE

- Telegram
  - Telegram is encrypted using MTProto which was developed based off three pre-existing encryption types; 256-bit symmetric AES encryption, 2048-bit RSA encryption and Diffie–Hellman key exchange. On this messaging product you are able to know whom you’re talking to as accounts are linked to phone numbers and have to be verified by either text, or phone call. Telegram is open source so can be viewed by the public and therefore reviewed. There have been hacks, to which one of the founders tweeted;
  “Users from troubled countries: make sure you have 2-step verification enabled – in Telegram and other services”.
  This tweet is a direct show of action towards the hacks to the messaging app from the supposed governments in countries like Russia and Iran (Lokot, 2016).
- Signal
  - Signal is a communication app used by Android and IOS users. It uses Signal Protocol encryption which has been found to be secure by university auditors. Users know whom they are connected with as they communicate with them directly. Signal is open source so can be reviewed. Signal thus far hasn’t been cracked.
- Snapchat
  - Snapchat is a multimedia messaging app. It uses the 256-bit algorithm to encrypt its messages and uses Cypher Block Chaining (CBC). Unlike the other two. Snapchat does not offer end-to-end encryption because while opened snaps are automatically deleted from servers, the unopened

messages can be stored on their servers for 30 days (Wagner, 2015).Users know whom they are connected to as it people can choose to accept whom they are “friends” with and send messages to. Snapchat is not opened sourced so people cannot view it and pick up on potential bugs. In December 2013 snapchat was hacked and over 4 million accounts were leaked. This was done by a hacking group to show “awareness” to users as Snapchat hadn’t proactively solved the problem (Leyden, 2014).

In a security stand point, Signal seems to be the better app out of the three to use. It hasn’t been exploited as of yet, and also has end-to-end encryption so contents can never be accessed by any other than yourself. However, the better app has to also have to have a good user interface, needs to be adaptable to user needs and has to be on trend for the demographic, so therefore, picking the best app out of the three has a broad spectrum.

BIBLIOGRAPHY

1. Australian Competition and Consumer Commission
   Nigerian Scams (n.d)
   Retrieved from: https://www.scamwatch.gov.au/types-of-scams/unexpected-money/nigerian-scams
2. Seidel, M [Michael Seidel]. (14th April, 2014)
   Cracking the Vigenére Cipher (english) [video file]
   Retrieved from: https://www.youtube.com/watch?v=ysLY2565lFw
3. Lokot, T (2nd May 2016)
   Is Telegram Really Safe for Activists Under Threat? These Two Russians Aren't So Sure.
   Retrieved from: https://advox.globalvoices.org/2016/05/02/is-telegram-really-safe-for-activists-under-threat-these-two-russians-arent-so-sure/
4. Wagner, K (21st December 2015)
   Is Your Messaging App Encrypted?
   Retrieved from: https://www.recode.net/2015/12/21/11621610/is-your-messaging-app-encrypted
5. Leyden, J (2 January 2014)
   Snapchat: In 'theory' you could hack... Oh CRAP is that 4.6 MILLION users' details?
   Retrieved from: https://www.theregister.co.uk/2014/01/02/snapchat_leak/