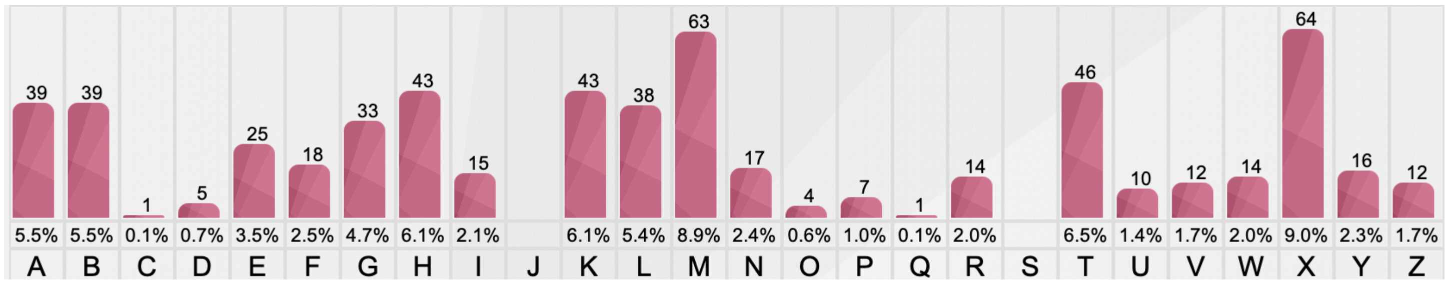
CORE

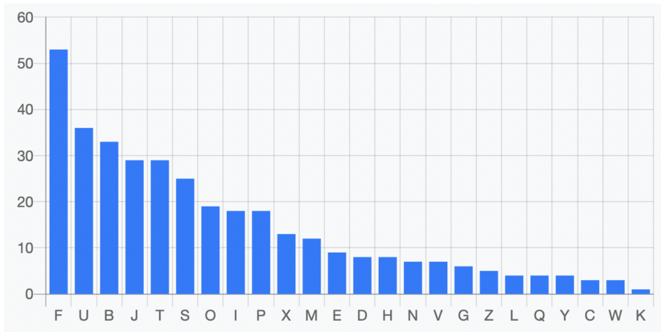
Caesar Cipher



1. The most common letter in the caesar cipher is X, used 64 times. This is closely followed by M, used 63 times.
2. E is the most common letter in the english language. If X is the most common letter in the ciphertext, we can assume the X = E. X is 19 characters ahead of E so we can assume that the caesar cipher has a shift of 19.
3. Text has a shift of 19 so the decrypted text is *"welcome to the fractured future, the first century following the singularity. earth has a population of roughly a billion hominids. for the most part, they are happy with their lot, living in a preserve at the bottom of a gravity well. those who are unhappy have emigrated, joining one or another of the swarming dense thinker clades that fog the inner solar system with a dust of molecular machinery so thick that it obscures the sun. except for the solitary lighthouse beam that perpetually tracks the earth in its orbit, the system from outside resembles a spherical fogbank radiating in the infrared spectrum; a matryoshka brain, nested dyson spheres built from the dismantled bones of moons and planets."*

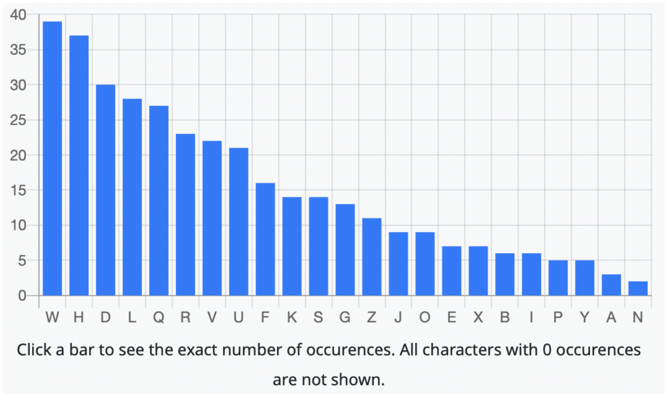
Vignere Cipher

1. Split 1
UNOFFJUTSUFETDBFUVUTUUSHFXXEHXIPXXIUHBUJQBIFTGXTTFNSJMFJTJIHHIUMHVWTZTS
DUTPEFSQTXIFFUBIOSUPDJOIFFSXFCTJBDFTXUJFFBSMBGBPOVFGNUTWYPSOEOSTBUFBPI
JVTUYPBFMUJFSUSOFFJUJBOFOZTJFBIBBSPSJPTMBBBNUTUBJJUXFBOZFOFOGXOCLFBSPT
ZDFSFDNEUPSBBJTSPEFUIBFFPBNFTPJJUZUKJFJUHFOGFOSSUDQFDVUFBFJMJFPIUTYUFIH
PELWIMFSFYJPIOBUTMSMNCLFJJVJTVFXIFJFBFBEFSOMFMQTUUTSBTGBIILTEBMXPOSUOFJ
B



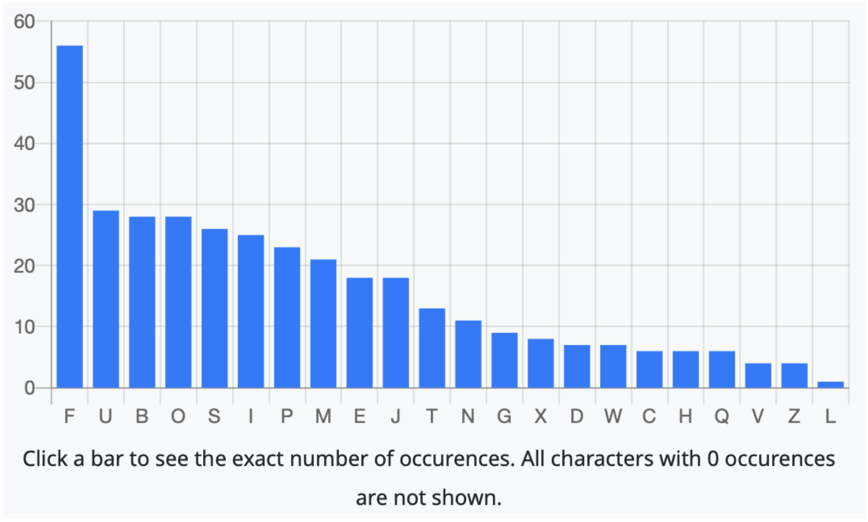
Click a bar to see the exact number of occurences. All characters with 0 occurences are not shown.

Split 2
KLLWQJLZHLRUWFQLKQHWHKHDUDLLGQHODDHLXQKEFVHZWQQVWFSDYBZUVWLWHHKDH
QHLUHFHHDXWZOXHLDZDEWHWQSWRSGLWZOLZZVOGDRRDKQUWVHSFRJZJPUFSHDBSUV
GYWHUGRWLQHUDHRSUQARWVZRLLLWQJBAYJFWOWVVUJQVRPFIUWOVGFSHBHGQWKDR
KGWPDPWOUFHHBEDUSVVSWQRDHRIWQGQWVORLRDSDGEVSWNUVUGRKRQVJLDYWLFW
ULKRXUFQLFUVWHGNZDKVERUDHXWHHLLIWQWAQVGYKVIRDHHHBSHEQKOULWIWFWQDW
JPSLDLXXRKRFQWEFDDHDDQHHIWQLDVRJ



Click a bar to see the exact number of occurences. All characters with 0 occurences are not shown.

Split 3
FEOFFIFFBNGBJIHOFJEBTFBOBTOOPUDESTBOQEFNXUOFPFFFIPBUFGXFUDOPUSFSSJSUFB
IOSSOIPEMEUOISFUJFFSPMBXMIPEEFBUMFESBZFUOXBBMFSSJOCPPVSWFMFBBFVSFZTTM
UWUMBFMFEQJIOGOFOOFFITIJSFMPISFDSUUOFMGEPXNFPVSTNNXIFWGBCIBHFPBFFSMMP
UZIJEBNUNOEHPIENJFUMWOXUQSUFJMBGIGTBOFCWUBOTFZWFFPOFNUBPUNIHMUEIOUUJ
JJWDSXMBPSTGPIFTNUBIFJFQDJESMIQSCHPETUICIJITOIFBPDMOSJGFVFFFPMLUDSOUHECJF
BPMQOF



Click a bar to see the exact number of occurences. All characters with 0 occurences are not shown.

2. We know that the key is of length 3 so every 3rd letter is encrypted the same way. Have split the whole encrypted message into 3 messaged by grouping the characters that are encrypted with the same shift. E is the most common letter in the english alphabet so it is reasonable to think that F in the ciphertext for split 1 and 3 represents E in the plaintext and W or H in the cipher text for split 2 represents E in the ciphertext. F is one letter ahead of E in the ciphertext so for split 1 and split 3, we can assume that B (1)

is the key. H is 3 letters ahead of E in the ciphertext so for split 2 we can assume that D (3) is the key. Therefore, the key for this vignere cipher should be BDB.

Other guesses for the shift value for each of the frequency histograms could be:
Split 1 - Q as U is second most frequency letter and U is 16 characters ahead of E. Q = 16
Split 2 - W is 18 characters ahead of E so S could be the key for split 2 as S = 18
Split 3 - Q as U is second most frequency letter and U is 16 characters ahead of E. Q = 16

3. The vignere key is BDB. The message is:

   THE MID NINETEEN EIGHTIES WERE A TIME OF DRASTIC CHANGE IN THE UNITED STATES THE REAGAN ERA WAS WINDING DOWN THE COLD WAR WAS HEATING UP AND THE IBMPC WAS THE NEWEST OF NEWNESSES THE COMPARATIVELY FEW WIRES STITCHING TOGETHER THE LARGER UNIVERSITY RESEARCH CENTERS AROUND THE WORLD PULSED WITH A NEW HEARTBEAT THE INTERNET PROTOCOL IP AND WHILE THE WORLDWIDE WEB WAS STILL A DECADE OR SO AWAY THE INTERNET WAS A REAL PLACE FOR A GROWING NUMBER OF COMPUTER SAVVY EXPLORERS AND ADVENTURERS READY TO SET SAIL ON THE VIRTUAL SEA TO EXPLORE AND EXPLOIT THIS NEW FRONTIER IN NINETEEN EIGHTY SIX HAVING RECENTLY LOST HIS RESEARCH GRANT ASTRONOMER CLIFFORD STOLL WAS MADE A COMPUTER SYSTEM ADMIN WITH THE WAVE OF A HAND BY THE MANAGEMENT OF LAWRENCE BERKELEY LABORATORYS PHYSICS DEPARTMENT COMMANDED TO GO FORTH AND ADMINISTER STOLL DOVE INTO WHAT APPEARED TO BE A SIMPLE TASK FOR HIS FIRST DAY ON THE JOB INVESTIGATING A SEVENTY FIVE CENT ERROR IN THE COMPUTER ACCOUNT TIME CHARGES LITTLE DID HE KNOW THAT THIS SIX BIT OVERCHARGE. WOULD TAKE OVER HIS LIFE FOR THE NEXT SIX MONTHS AND HAVE THIS SELF PROCLAIMED BERKELEY HIPPIE RUBBING SHOULDERS WITH THE FBI THE CIATHENSA AND THE GERMAN POLICE ALL IN PURSUIT OF THE SOURCE A NEST OF BLACK HAT HACKERS AND A TANGLED WEB OF INTERNATIONAL ESPIONAGE

Exhaustive Key Search

1. 2^128 / 2^30 (2^30 is the closest to 1 billion = 1073741824)
   = 3.16912650057057350374175801344 × 10^29 seconds
   3.154x10^7 seconds in a year so (3.16912650057057350374175801344×10^29) / (3.154x10^7)
   = 1.004796×10^22 years is the maximum time it would take for a brute force attack on a single document
2. 2^1024/2^30 (2^30 is the closest to 1 billion = 1073741824)
   = 1.67423219872854268898 x 10^299 seconds
   = (1.67423219872854268898 x 10^29) / (3.154x10^7)
   = 5.308282 x 10^291 years is the maximum time it would take for a brute force attack on a single document

XOR Cryptography

1. ASSIGNMENT as binary - 01000001 01010011 01010011 01001001 01000111 01001110 01001101 01000101 01001110 01010100
   As bitwise XOR = 00010011

ABBAABBAAB as binary - 01000001 01000010 01000010 01000001 01000001 01000010 01000010 01000001 01000001 01000010
As bitwise XOR = 00000011

Encrypted XOR = 00010000

Hashes

1. MD5 hash value of Slax download - b347608199f2a0a70fb7a31beb460c20
   curl -0 https://ftp.sh.cvut.cz/slax/Slax-11.x/slax-ipxe.iso
   ```
   Input: curl -0  https://ftp.sh.cvut.cz/slax/Slax-11.x/slax-ipxe.iso
   Ouput:
    % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                Dload  Upload   Total   Spent    Left  Speed
   100  304k  100  304k    0     0   98k      0  0:00:03  0:00:03 --:--:--  98k

   Input: ls
   output:
   11-0.txt      ciphertext2.txt      Downloads            message                          plaintext.txt
   36-0.txt      ciphertext.txt       hackerdirectory  message.asc                       plaintext.txt.asc
   assignment1            ciphertext.txt-binary  hidden                                       private
   bob-publickey.asc  CYBR171-2022T1   lab01                   plaintext1.txt         public_html
   ciphertext1.txt    Desktop      slax-ipxe.iso

   Input: md5sum slax–ipxe.iso
   Output: b347608199f2a0a70fb7a31beb460c20 slax-ipxe.iso
   ```

2. Proving the integrity of the downloaded file does not guarantee its authenticity because despite the hashes matching, this only verifies that we downloaded something from the website (this could be anything from the website, however. The only way we can be sure is to compare the file we downloaded with the original version of that file.

Symmetric cryptography with command-line tools

1. openssl enc -des-cbc -pbkdf2 -in cipher.txt -out ciphers.des.enc -pass pass:banfiebrod

2. openssl enc -aes-256-ecb -pbkdf2 -in cipher.txt -out ciphers.aes.enc -pass pass:banfiebrod
3. Do this

Public Key cryptography using command line tools

1. Input: gpg --import gpg-key.txt
   ```
   output : gpg: key FC8D7EE4F3B2AC61: public key "CYBR171 2021
                <cybr171-staff@ecs.vuw.ac.nz>" imported
               gpg: Total number processed: 1
               gpg:               imported: 1

   Input: gpg --verify document1.asc
   Output: gpg: Signature made Thu 04 Mar 2021 14:14:50 NZDT
                gpg:                using RSA key 08EA55773F3259542E69BBBEFC8D7EE4F3B2AC61
                gpg:                issuer "cybr171-staff@ecs.vuw.ac.nz"
                gpg: Good signature from "CYBR171 2021 <cybr171-staff@ecs.vuw.ac.nz>" [unknown]
                gpg: WARNING: This key is not certified with a trusted signature!
                gpg:          There is no indication that the signature belongs to the owner.
                Primary key fingerprint: 08EA 5577 3F32 5954 2E69  BBBE FC8D 7EE4 F3B2 AC61
   ```

```
Input: gpg --verify document2.asc
Output: gpg: Signature made Thu 04 Mar 2021 14:15:19 NZDT
        gpg:            using RSA key 08EA55773F3259542E69BBBEFC8D7EE4F3B2AC61
        gpg:            issuer "cybr171-staff@ecs.vuw.ac.nz"
        gpg: BAD signature from "CYBR171 2021 <cybr171-staff@ecs.vuw.ac.nz>" [unknown]
```

document2.asc was modified after it was signed. I know this due to the gpg output shoding that it is a BAD signature from CYBR171 2021 compared with the good signature for document1.asc

2. By signing a message or file, we have increased the level of integrity associated with it. In the PGP encrypted message, no one else can intercept the message in plaintext. This can make our identity provable and tied to our message, as well as showing the message has not been tampered with. PGP signature is tied to a PGP identity (Harith's) with a private key.

3. Input: gpg --keyserver pgp.net.nz --search-keys

```
Output: barretts% gpg --keyserver pgp.net.nz --recv-keys BC8B95B6
        gpg: key BB4F3E77BC8B95B6: public key "Harith Al-Sahaf <harith.al-sahaf@ecs.vuw.ac.nz>"
        imported
        gpg: Total number processed: 1
        gpg:           imported: 1
        barretts% gpg:          imported: 1
```

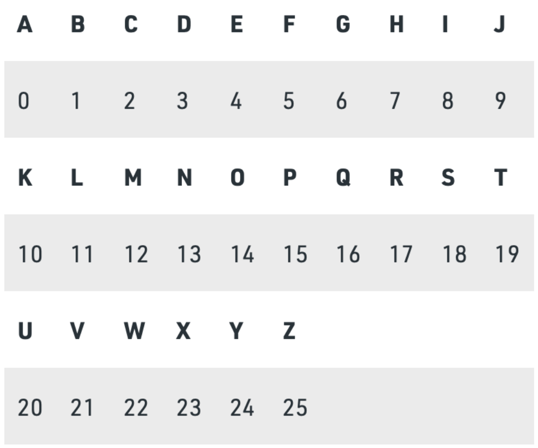
```
Input: gpg --verify message.asc
Output: gpg --verify message.asc
        gpg: Signature made Wed 03 Mar 2021 13:30:54 NZDT
        gpg:            using RSA key A7711E3B69CEE19800F98EBFBB4F3E77BC8B95B6
        gpg:            issuer "harith.al-sahaf@ecs.vuw.ac.nz"
        gpg: Good signature from "Harith Al-Sahaf <harith.al-sahaf@ecs.vuw.ac.nz>" [unknown]
        gpg: WARNING: This key is not certified with a trusted signature!
        gpg:        There is no indication that the signature belongs to the owner.
        Primary key fingerprint: A771 1E3B 69CE E198 00F9  8EBF BB4F 3E77 BC8B 95B6
        gpg: WARNING: not a detached signature; file 'message' was NOT verified!
```

This is Harith Al-Sahaf's key. While it did come from an official key server, the key cannot be trusted, as shown by the warning. You canot trust that the message really came from Harith because it could have been intercepted. The last warning also shows that the message was not verified. The only way to check this would be to call Harith and read out the primary key fingerprint. If he indicates that it is the same primary key fingerprint, then you can be sure that the message is owned by him and is the real deal.

COMPLETION

XOR Encryption

1. To implement a one time pad to encrypt the message "APPLE", I would convert the plain text to cipher text by assigning a number to each character of the plain text.



A	B	C	D	E	F	G	H	I	J
0	1	2	3	4	5	6	7	8	9
K	L	M	N	O	P	Q	R	S	T
10	11	12	13	14	15	16	17	18	19
U	V	W	X	Y	Z				
20	21	22	23	24	25				

Plain text to ciphertext - APPLE = 0 15 15 11 4

The random secret key would be something totally unrelated to the word apple, could be MONEY for example. The length of the key should be equal to that of the plain text.

Key - MONEY = 12 14 13 4 24

Plain text + key = 12 29 28 15 28
(29-26 = 3) (28-26 = 2)
Plain text + key = 12 3 2 15 2

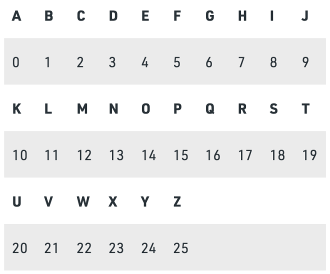
Cipher text = MDCPC

This could also be done using a java program instead of by hand.

Substitution Ciphers

1. A double caesar cipher encryption is no more secure than single encryption as double encryption using caesar cipher is equivalent to single encryption using another offset (a + b) mod 26.

Plaintext	h	e	l	l	o	y	o	u
Repeated key	d - 3	o - 14	g - 6	d	o	g	d	o
Ciphertext	k	s	r	o	c	e	r	i



A	B	C	D	E	F	G	H	I	J
0	1	2	3	4	5	6	7	8	9
K	L	M	N	O	P	Q	R	S	T
10	11	12	13	14	15	16	17	18	19
U	V	W	X	Y	Z				
20	21	22	23	24	25				

Here, each letter is being shifted by a different amount rather than all having the same shift. D is 3, O is 14, and G is 6. If the plaintext message was helloyou, and the key was dog, the first letter of the key would shift the first letter of the plaintext by 3 as D=3. So H becomes K. You would move E to S by 14 letters as O=14 and L to R by 6 letters as G=6. Then the key repeats where the fourth letter would be shifted by D=3. This creates an encrypted message ksroceri.

Below is an example of a single caesar cipher with a shift of d=3

Plaintext	h	e	l	l	o	y	o	u
Ciphertext	k	h	o	o	r	b	r	x

Quantum computers

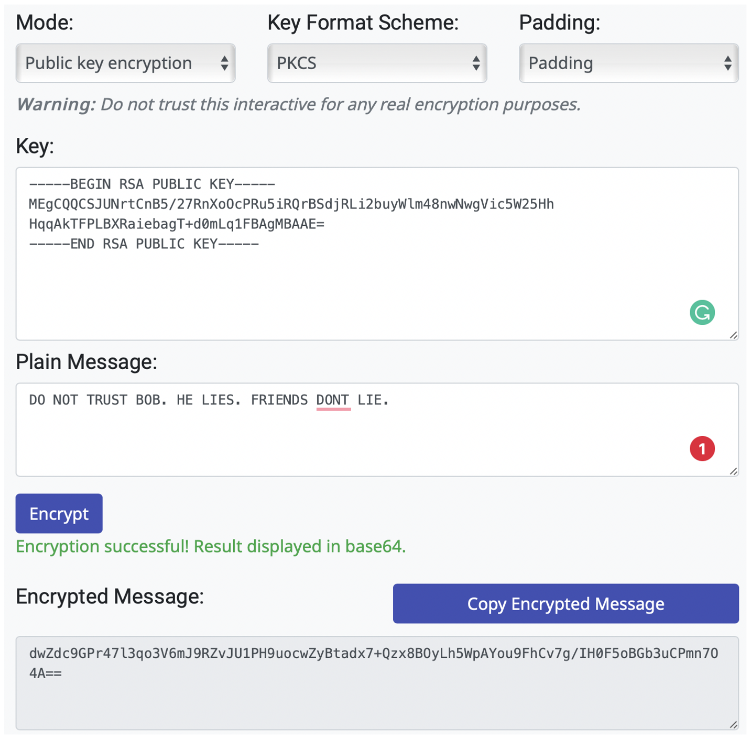
Before:
2^1024/2^30 (2^30 is the closest to 1 billion = 1073741824)
= 1.674232198728542688898 x 10^299 seconds
= 5.308282 x 10^291 years is the maximum time it would take for a brute force attack on a single document

With quantum:
2^512/2^30
= 1.24869942012639689 x 10^145 seconds
= (1.24869942012639689 x 10^145) / (3.154x10^7)
= 3.95909771758527866 x 10^137 years to exhaustive key search

Quantum computers make a significant difference and decrease the number of years to carry out an exhaustive key search. 3.95909771758527866 x 10^137 years is still a very very long time, however. With millions of quantum computers carrying out the same exhaustive key search, this could be cracked in a few years.

Public Key Cryptography

1. Carol would use Alice’s public key to encrypt the message. Because public and private keys are inverses of each other, only Alice’s private key could decrypt the message later on and Bob would not be able to read it (as he has no access to Alice’s private key since it is a secret.



Mode: Public key encryption

Key Format Scheme: PKCS

Padding: Padding

Warning: *Do not trust this interactive for any real encryption purposes.*

Key:

```
-----BEGIN RSA PUBLIC KEY-----
MEgCQQCSJUNrtCnB5/27RnXoOcPRu5iRQrBSdjRLi2buyWlm48nwNwgVic5W25Hh
HqqAkTFPLBXRaiebagT+d0mLq1FBAgMBAAE=
-----END RSA PUBLIC KEY-----
```

Plain Message:

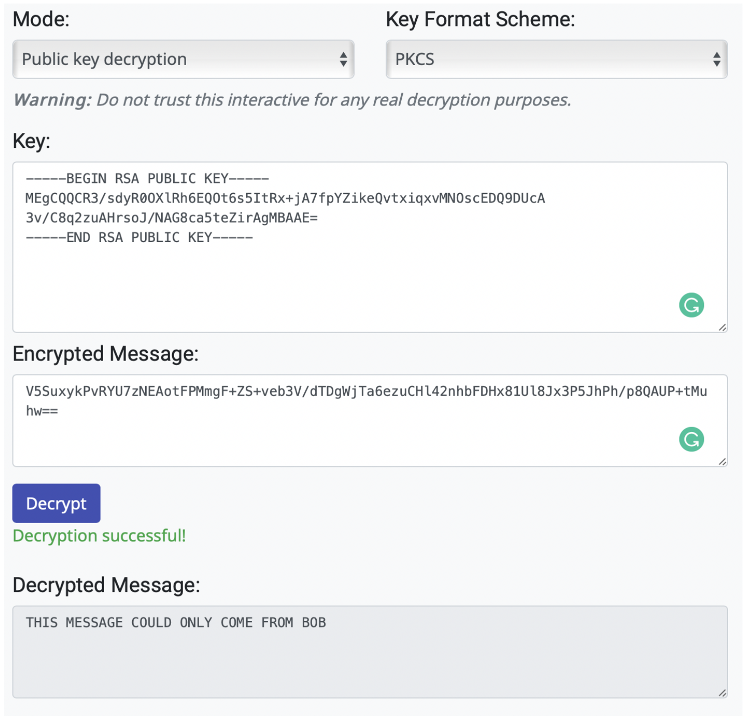
```
DO NOT TRUST BOB. HE LIES. FRIENDS DONT LIE.
```

Encryption successful! Result displayed in base64.

Encrypted Message:

```
dwZdc9GPr47l3qo3V6mJ9RZvJU1PH9uocwZyBtadx7+Qzx8BOyLh5WpAYou9FhCv7g/IH0F5oBGb3uCPmn7O
4A==
```

2. If Bob encrypts the message using his private key and sends the encrypted message to Alice, then Alice can be sure that this information is coming from Bob. If Alice can decrypt the data with Bob's public key, the message must have been encrypted by Bob with his private key, and only Bob has Bob's private key. However, anyone could read the message because Bob's public key is public.



Mode: Public key decryption

Key Format Scheme: PKCS

Warning: *Do not trust this interactive for any real decryption purposes.*

Key:

```
-----BEGIN RSA PUBLIC KEY-----
MEgCQQCR3/sdyR0OXlRh6EQOt6s5ItRx+jA7fpYZikeQvtxiqxvMNOscEDQ9DUcA
3v/C8q2zuAHrsoJ/NAG8ca5teZirAgMBAAE=
-----END RSA PUBLIC KEY-----
```

Encrypted Message:

```
V5SuxykPvRYU7zNEAotFPMmgF+ZS+veb3V/dTDgWjTa6ezuCHl42nhbFDHx81Ul8Jx3P5JhPh/p8QAUP+tMu
hw==
```

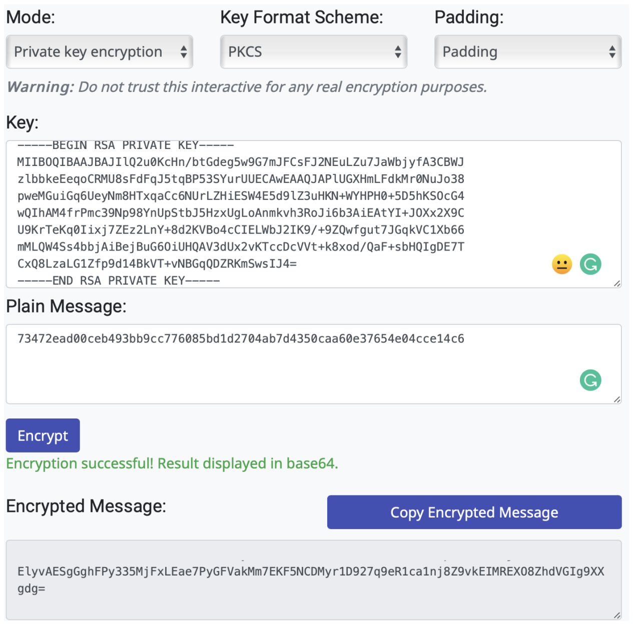
Decryption successful!

Decrypted Message:

```
THIS MESSAGE COULD ONLY COME FROM BOB
```

3. SHA2 Hash of "ALICE WROTE THIS" = 73472ead00ceb493bb9cc776085bd1d2704ab7d4350caa60e37654e04cce14c6

Encrypt using private key. This is the digital signature. Because it is encrypted using a private key, everyone who see's Alice's digital signature can be certain that the message attached to it is really from her since no one else has access to her private key.



Mode: Private key encryption
Key Format Scheme: PKCS
Padding: Padding

Warning: Do not trust this interactive for any real encryption purposes.

Key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIBOQIBAAJBAJIlQ2u0KcHn/btGdeg5w9G7mJFCsFJ2NEuLZu7JaWbjyfA3CBWJ
zlbbkeEeqoCRMU8sFdFqJ5tqBP53SYurUUECAwEAAQJAPlUGXHmLFdkMr0NuJo38
pweMGuiGq6UeyNm8HTxqaCc6NUrLZHiESW4E5d9lZ3uHKN+WYHPH0+5D5hKSOcG4
wQIhAM4frPmc39Np98YnUpStbJ5HzxUgLoAnmkvh3RoJi6b3AiEAtYI+JOXx2X9C
U9KrTeKq0Iixj7ZEz2LnY+8d2KVBo4cCIELWbJ2IK9/+9ZQwfgut7JGqkVC1Xb66
mMLQW4Ss4bbjAiBejBuG6OiUHQAV3dUx2vKTccDcVVt+k8xod/QaF+sbHQIgDE7T
CxQ8LzaLG1Zfp9d14BkVT+vNBGqQDZRKmSwsIJ4=
-----END RSA PRIVATE KEY-----
```

Plain Message:

```
73472ead00ceb493bb9cc776085bd1d2704ab7d4350caa60e37654e04cce14c6
```

Encryption successful! Result displayed in base64.

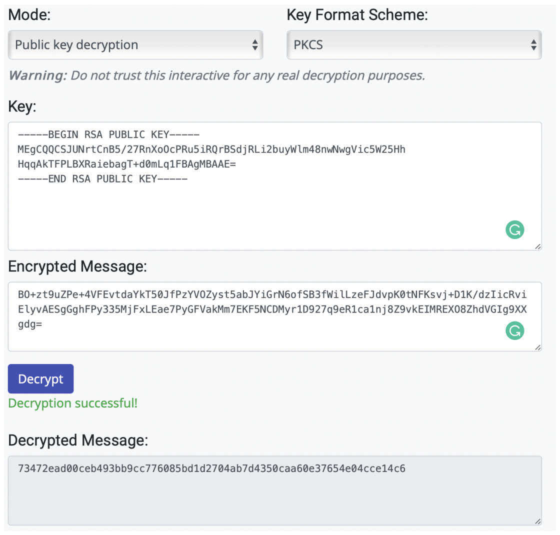
Encrypted Message:

```
ElyvAESgGghFPy335MjFxLEae7PyGFVakMm7EKF5NCDMyr1D927q9eR1ca1nj8Z9vkEIMREXO8ZhdVGIg9XX
gdg=
```

"ALICE WROTE THIS

BO+zt9uZPe+4VFEvtdaYkT50JfPzYVOZyst5abJYiGrN6ofSB3fWilLzeFJdvpK0tNFKsvj+D1K/dzIicRviElyvAESgGghFPy335MjFxLEae7PyGFVakMm7EKF5NCDMyr1D927q9eR1ca1nj8Z9vkEIMREXO8ZhdVGIg9XXgdg="

To verify this the recipient would:
Decrypt signature using Alice's public key. Use the same hash and compare the decrypted digital signature with the message. If the hashes match, the recipient knows the message was not altered in transit from the sender



Mode: Public key decryption
Key Format Scheme: PKCS

Warning: Do not trust this interactive for any real decryption purposes.

Key:

```
-----BEGIN RSA PUBLIC KEY-----
MEgCQQCSJUNrtCnB5/27RnXoOcPRu5iRQrBSdjRLi2buyWlm48nwNwgVic5W25Hh
HqqAkTFPLBXRaiebagT+d0mLq1FBAgMBAAE=
-----END RSA PUBLIC KEY-----
```

Encrypted Message:

```
BO+zt9uZPe+4VFEvtdaYkT50JfPzYVOZyst5abJYiGrN6ofSB3fWilLzeFJdvpK0tNFKsvj+D1K/dzIicRvi
ElyvAESgGghFPy335MjFxLEae7PyGFVakMm7EKF5NCDMyr1D927q9eR1ca1nj8Z9vkEIMREXO8ZhdVGIg9XX
gdg=
```

Decryption successful!

Decrypted Message:

```
73472ead00ceb493bb9cc776085bd1d2704ab7d4350caa60e37654e04cce14c6
```

Malware Threats

Article used attached here

The malware shown in this article from ThreatPost was a cyberattack whereby phishers attempted to steal Instagram login credentials by threatening account deactivation for sharing ‘fake content’. The attack was through email and under the guise that it was sent from Instagram’s technical support, looking fairly identical with obvious brand-impersonation, despite the grammar and spelling mistakes used. The attackers also used an email that obviously did not come from Instagram (membershipform@outlook.com.tr) but bypassed Google’s email security by using a valid domain name. Because they crafted the email address to be so long, mobile users would only see characters before the ‘@’, being ‘membershipform’ which would not seem to be of concern. This may cause a loss to Instagram due to user’s trust being impacted, whereby they associate the bad experience with Instagram, despite them having no part to play in the scam. It becomes a ‘guilty by association’ situation whereby brand reputation is damaged. The media may take note of the situation too, such as how ThreatPost did and instagram may lose users who are particularly fearful of identity theft, such as the ones targetted at this Life Insurance company. Using CERT NZ critical controls, a consumer could be less likely to face consequences of a malware attack such as this. Users should follow the 2nd CERT NZ control and implement multi-factor authentication and verification, in particular for internet accounts. Enforcing this prevents unauthorised access due to weak credentials. Another one could be the 4th CERT NZ control where they encourage consumers to configure logging and alerting. This could be turning on unknown login alerts/notifications or by using software such as Armorblow to detect email malware attacks.