Please include the following in your assignment writeup:

- Assignment number, your full name and Student ID.

- Clearly labelled answers to the questions (Core 1, Challenge 99, etc).

- Report is submitted in electronic form as a PDF file.

ASSIGNMENT NUMBER: 1

FULL NAME : James Small

STUDENT ID:300652665

CORE

1.

The most common character in the cipher text is “n”.

2.

Since the plain text is most likely to be in the English language and the most common character in the English is “E” it is most likely that n corresponds to e. This would mean that the plain text was shifted by 9 characters to right.

3.

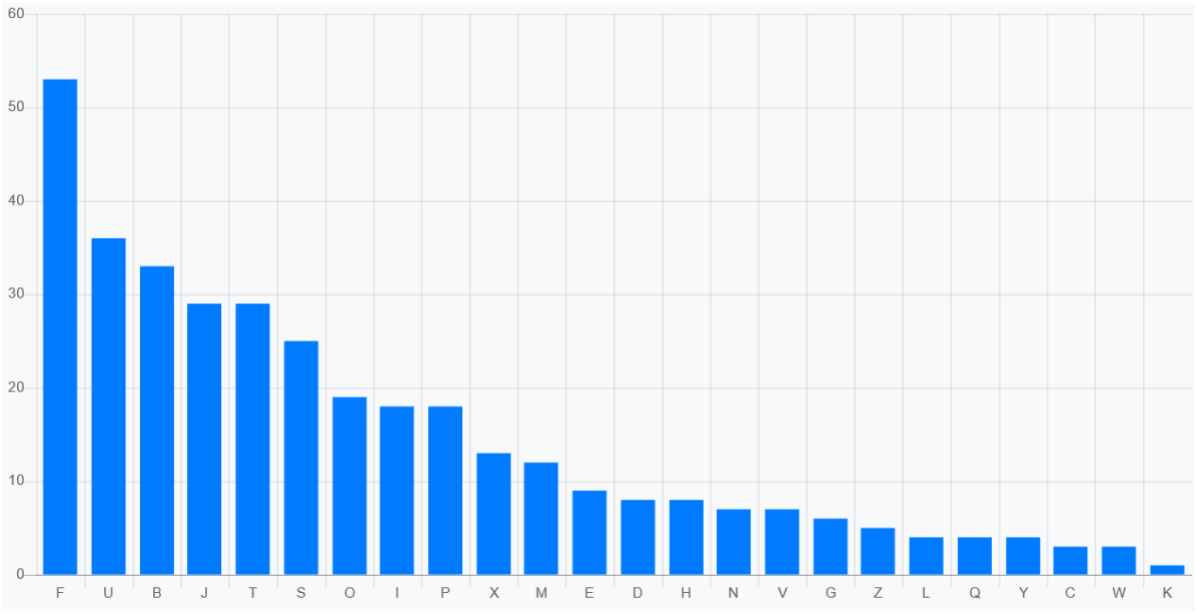
Plain text – used https://www.dcode.fr/caesar-cipher

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 densethinker 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. the splintery metaconsciousness of the solar system has largely sworn off its pre-posthuman cousins dirtside, but its minds sometimes wander nostalgiawise. when that happens, it casually

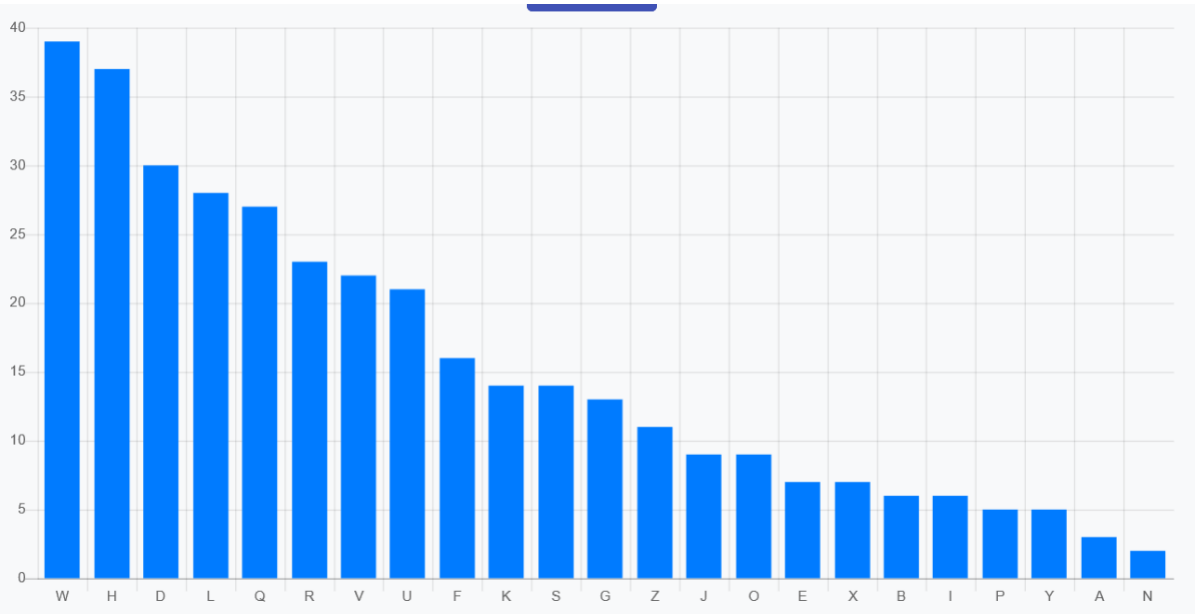
spams earth is rf spectrum with plans for cataclysmically disruptive technologies that emulsify whole industries, cultures, and spiritual systems. a sane species would ignore these get-evolved-quick schemes, but there is always *someone* who will take a bite from the forbidden fruit. there is always someone who unaccountably carries the let is-lick-the-frozen-fence-post gene. there is always a fucking geek who will do it because it is a historical goddamned technical fucking imperative. whether the enlightened, occulting smartcloud sends out its missives as pranks, poison, or care packages is up for debate. asking it to explain its motives is about as productive as negotiating with an ant colony to get it to abandon your kitchen. whatever the motive, humanity would be much better off if the cloud would evolve into something uninterested in communicating with meatpeople--or at least smart enough to let well alone. but until that happy day, there is the tech jury service: defending the earth from the scum of the post-singularity patent office.

4.

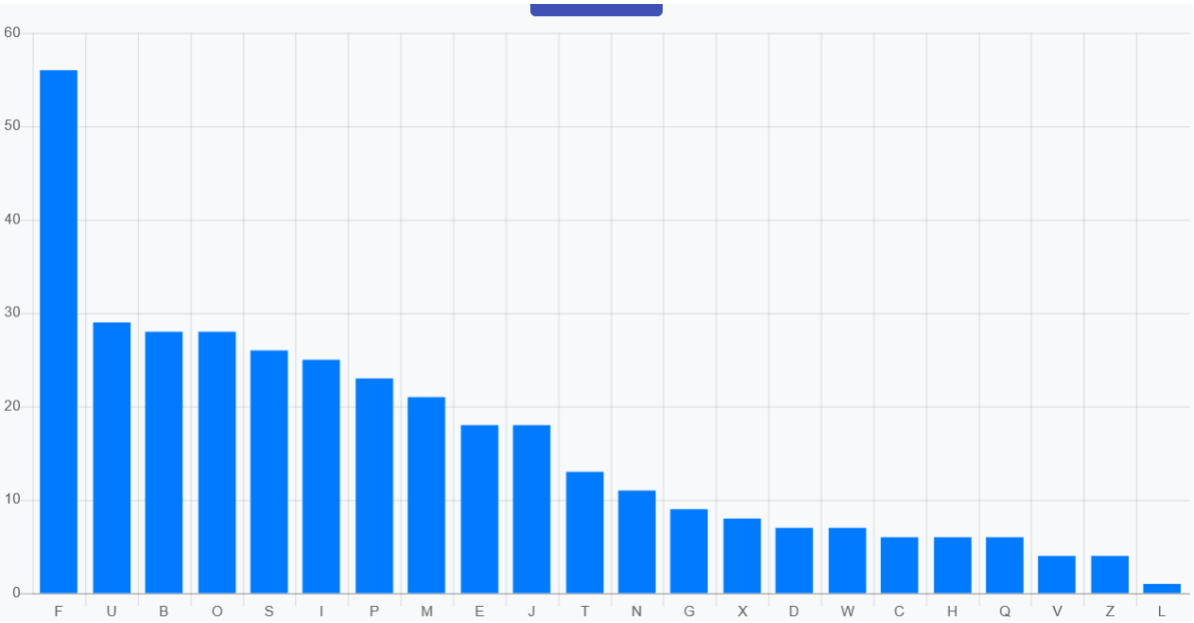
1st split



2nd split



3rd split



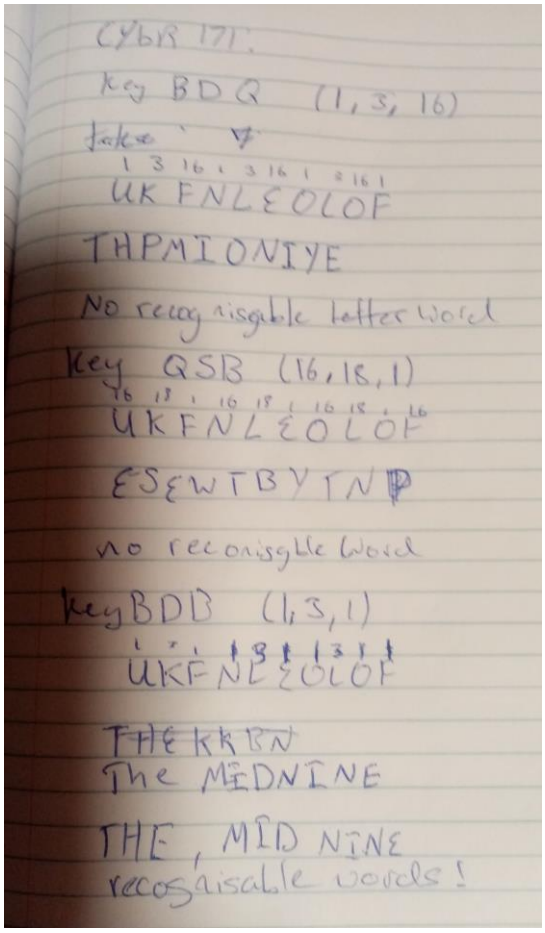
5.1st split = 1(B), 16(Q), in the English language “e” is the most common occurring character so it Is most likely to occur within the two most common characters in the cipher text. The shift value of “f” to e is 1 which correlates to “B”, shift value from U -> e is 16(Q).

2nd split – 18(S), 3(D), in the English language “e” is most common occurring character so its is most likely to occur within the two most common characters in the cipher text. The sift value of W to E is 18(s) and the shift value of H to E is 3(D).

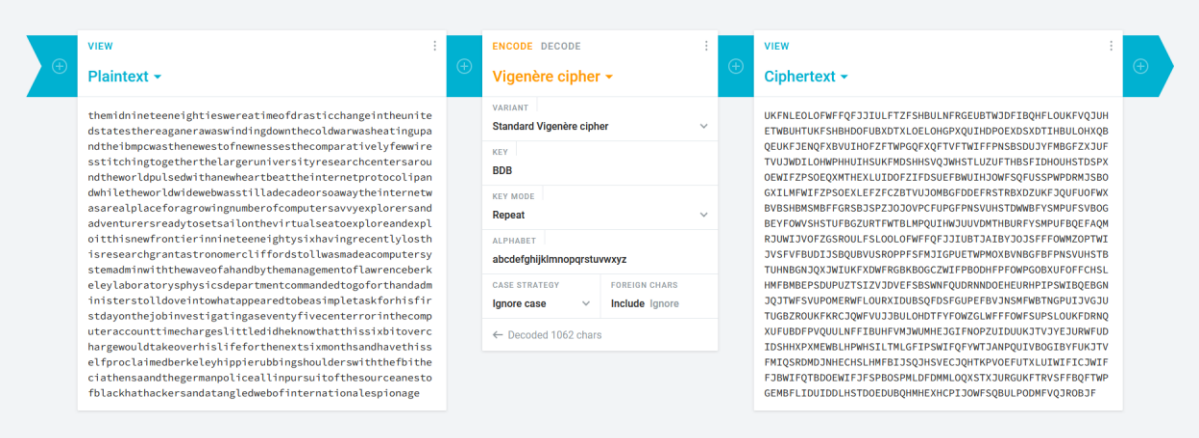
3rd split – 1(B), 16(Q) the 3rd split has the same top two occurring characters as the 1st split however F occurs much more greatly than the others in the 3rd split, and since e is most likely to occur in the top two most common character f will most likely = e with a shift value of 1(b) and from u -> e the shift values is 16(Q).

6.Possible keys:

I first took some of the starting characters out of the cipher text and then tried to decrypt to see if the key gave back recognisable words from the English language.



Once I found a key that worked I then took the whole cipher text and tried to decrypt it with the key that worked. By using https://cryptii.com/pipes/vigenere-cipher



I found that the key BDB worked, so the key to decrypt the Vigenère cipher text is **BDB** (shift values of 1, 3, 1)

“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 IBM PC was the newest of newness. 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 World Wide 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 Laboratory's 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 CIA, the NSA, and the German police all in pursuit of the source: an nest of black hat hackers and a tangled web of international espionage.” (Decrypted Text)

7.

Used https://www.wolframalpha.com/input for Q 7-8

128-bit encryption = $2^{128} \approx 3.4*10^{38}$

With $1*10^{10}$ operations pre second

$3.4*10^{38}/1*10^{10} = 3.4*10^{28}$ seconds to brute force AES 128-bit encryption.

8.

1028-bit RSA encryption = $2^{1024} \approx 1.797*10^{308}$

At $1*10^{10}$ operations per second or $1*10^{10} * (60*60*24*365) = 3.1536*10^{17}$ operations per year

Time to brute force in years -> $1.797*10^{308}/3.1536*10^{17} = 5.698*10^{290}$ years

9.

ASSIGNMENT = 01000001 01010011 01010011 01001001 01000111 01001110 01001101 01000101 01001110 01010100

BAABBAABBA = 01000010 01000001 01000001 01000010 01000010 01000001 01000001 01000010 01000010 01000001

ASSIGNMENT encrypted using the key in binary = 00000011 00010010 00010010 00001011 00000101 00001111 00001100 00000111 00001100 00010101

10.

barretts /home/smalljame/CYBR171/assignment01/q1011 % curl -O https://ftp.sh.cvut.cz/slax/Slax-11.x/slax-ipxe.iso

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  908k  100  908k    0     0   166k      0  0:00:05  0:00:05 --:--:--  214k
```

barretts /home/smalljame/CYBR171/assignment01/q1011 % curl -O https://ftp.sh.cvut.cz/slax/Slax-11.x/md5.txt

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   613  100   613    0     0    289      0  0:00:02  0:00:02 --:--:--   289
```

barretts /home/smalljame/CYBR171/assignment01/q1011 % md5sum slax-ipxe.iso

b347608199f2a0a70fb7a31beb460c20 slax-ipxe.iso

barretts /home/smalljame/CYBR171/assignment01/q1011 % cat md5.txt

b347608199f2a0a70fb7a31beb460c20 slax-ipxe.iso

7439117a5336b2966d673f02601b16a6 slax-32bit-11.2.0.iso

ed56aee444b4c2b98043c7c0d74afb2b slax-64bit-11.2.0.iso

305af79378082c5102ec2918666eb806 slax-32bit-11.2.1.iso

619d72e9d2498ff882e85946614ef3bf slax-64bit-11.2.1.iso

042c2154a766bcecbf49d82ec32a658a slax-32bit-11.3.0.iso

9d94c1796ba4c79fb05bb9f35c3fe188 slax-64bit-11.3.0.iso

d761e0c695517ceb99a1a51b05fd6777 slax-32bit-11.4.0.iso

132147e214fc32e39ea73f3a5438cedc slax-64bit-11.4.0.iso

651055d0030de9775297b04c5049602a slax-32bit-11.6.0.iso

85621ae3f6633017a59dc1ec79919d76 slax-64bit-11.6.0.iso

barretts /home/smalljame/CYBR171/assignment01/q1011 %

barretts /home/smalljame/CYBR171/assignment01/q1011 %

the slax file has been verified and its integrity is sound by comparing the hashes of the file to the documented hash of the original.

11.

Verifying the integrity of the file proves that the document has not been tampered with in any way while in transit because if the file is changed its current hash will change completely no matter how small it is tampered with and will not match the original hash, however this does not prove it was sent from the correct person because in no way can it prove that it is them just by comparing hashes. Someone could tamper with the file then hash it and send it to the receiver and by just comparing hashes it does not prove they are the original sender nor could u tell its already been tampered with.

12.

barretts /home/smalljame/CYBR171/assignment01 % ls

alice.txt bob.txt carol.txt cat-b.jpg ciphers.txt document1.asc maori-battalion.txt message-bob-to-alice.txt slax viginere.txt

assignment1.zip caesar-cipher.txt cat-a.jpg cat-c.jpg cybr171.pub.key document2.asc message.asc payroll.bf.enc treasure.bf.enc

barretts /home/smalljame/CYBR171/assignment01 % mkdir ques(12)

zsh: unknown file attribute: 1

barretts /home/smalljame/CYBR171/assignment01 % mkdir q12

barretts /home/smalljame/CYBR171/assignment01 % cp ciphers.txt q12

barretts /home/smalljame/CYBR171/assignment01 % cd q12

barretts /home/smalljame/CYBR171/assignment01/q12 % ls

ciphers.txt

barretts /home/smalljame/CYBR171/assignment01/q12 %

barretts /home/smalljame/CYBR171/assignment01/q12 % openssl enc -des-cbc -pbkdf2 -in ciphers.txt -out ciphers.des.enc -pass pass:smalljame -provider legacy -provider default

barretts /home/smalljame/CYBR171/assignment01/q12 % ls

ciphers.des.enc ciphers.txt

barretts /home/smalljame/CYBR171/assignment01/q12 %

13.

barretts /home/smalljame/CYBR171/assignment01/q13 % openssl enc -aes-256-ecb -pbkdf2 -in ciphers.txt -out ciphers.aes.enc -pass pass:smalljame

barretts /home/smalljame/CYBR171/assignment01/q13 % ls

ciphers.aes.enc ciphers.txt

barretts /home/smalljame/CYBR171/assignment01/q13 %

14.

barretts /home/smalljame/CYBR171/assignment01/q14 % ls

treasure.bf.enc

barretts /home/smalljame/CYBR171/assignment01/q14 % openssl enc -blowfish -d -pbkdf2 -in treasure.bf.enc -out treasure-d.txt -pass pass:lucre -provider legacy -provider default

barretts /home/smalljame/CYBR171/assignment01/q14 % ls

treasure.bf.enc treasure-d.txt

barretts /home/smalljame/CYBR171/assignment01/q14 % cat treasure-d.txt

I have deposited in the county of Bedford, about four miles from Buford's, in an excavation or

vault, six feet below the surface of the ground, the following articles: The deposit consists

of two thousand nine hundred and twenty one pounds of gold and five thousand one hundred pounds

of silver; also jewels, obtained in St. Louis in exchange for silver to save transportation.

The above is securely packed in iron pots, with iron covers. The vault is roughly lined with

stone, and the vessels rest on solid stone, and are covered with others.

barretts /home/smalljame/CYBR171/assignment01/q14 %

15.

Document2 has been tampered with and can not be trusted

barretts /home/smalljame/CYBR171/assignment01/q1516 % ls

cybr171.pub.key document1.asc document2.asc

barretts /home/smalljame/CYBR171/assignment01/q1516 % gpg --import cybr171.pub.key

gpg: key C54C701B32A6E1C9: public key "CYBR171 <cybr171-staff@ecs.vuw.ac.nz>" imported

gpg: Total number processed: 1

gpg: imported: 1

barretts /home/smalljame/CYBR171/assignment01/q1516 % gpg --verify document1.asc

gpg: Signature made Mon 13 Mar 2023 09:46:57 NZDT

gpg: using RSA key 1A36CCAA172B659F8885FE6DC54C701B32A6E1C9

gpg: Good signature from "CYBR171 <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: 1A36 CCAA 172B 659F 8885 FE6D C54C 701B 32A6 E1C9

barretts /home/smalljame/CYBR171/assignment01/q1516 % gpg --verify document2.asc

gpg: Signature made Mon 13 Mar 2023 09:46:57 NZDT

gpg: using RSA key 1A36CCAA172B659F8885FE6DC54C701B32A6E1C9

gpg: BAD signature from "CYBR171 <cybr171-staff@ecs.vuw.ac.nz>" [unknown]

barretts /home/smalljame/CYBR171/assignment01/q1516 %

16.

The PGP signature must contain the signed file name, the signature creation date, and the RSA key ID used to sign the file

17.

barretts /home/smalljame/CYBR171/assignment01/q17 % gpg --keyserver pgp.net.nz --recv-keys 1C6DC77C

gpg: key C615B1761C6DC77C: "Harith Al-Sahaf <harith.al-sahaf@ecs.vuw.ac.nz>" not changed

gpg: Total number processed: 1

gpg: unchanged: 1

barretts /home/smalljame/CYBR171/assignment01/q17 % cd ..

barretts /home/smalljame/CYBR171/assignment01 % ls

alice.txt bob.txt carol.txt cat-b.jpg ciphers.txt document1.asc maori-battalion.txt message-bob-to-alice.txt q1011 q13 q1516 slax viginere.txt

assignment1.zip caesar-cipher.txt cat-a.jpg cat-c.jpg cybr171.pub.key document2.asc message.asc payroll.bf.enc q12 q14 q17 treasure.bf.enc

barretts /home/smalljame/CYBR171/assignment01 % cp message.asc q17

barretts /home/smalljame/CYBR171/assignment01 % cd q 17

cd: string not in pwd: q

barretts /home/smalljame/CYBR171/assignment01 % cd q17

barretts /home/smalljame/CYBR171/assignment01/q17 % gpg --verify message.asc

gpg: Signature made Mon 13 Mar 2023 10:33:21 NZDT

gpg: using RSA key BE655EE79B2C4E7522BADD16C615B1761C6DC77C

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: BE65 5EE7 9B2C 4E75 22BA DD16 C615 B176 1C6D C77C

barretts /home/smalljame/CYBR171/assignment01/q17 %

The key can belong to anyone pretending to be Harith Al-Sahaf because it Is a RSA public keys a RSA public key.

You cannot trust this key even though it came from an official key server because anyone can upload a key in anyone’s name or email to a key server.

You cannot trust the message really come from who it claims to be because they can say that they are some ous.

The only that you can trust both the key used and they are whom they say they are on the message is to either establish a web of trust with them by getting 2 or more trusted users to vouch for the the key and message or to meet the other user in person and physically check what there credentials and key are.

COMPLETION

18.

To make a one time pad you need to generate the key values completely randomly, I would do this by recording the values of atmospheric noise (which is completely random due to complexity), the value length should be no more than the length of the plaintext.

I would then convert the values using a number to binary converter to get the key
this would be a true random key that shouldn't occur again because the variable in that order a randomly generated by the atmospheric noise and unless the hacker knew the originating start of all the noise the key will not exist again.

Then encrypt the plaint text(in binary form) with the key using XOR encryption.

19.

One-time cearser encryption:

Plaintext = "HELLO FELLOW HUMAN"

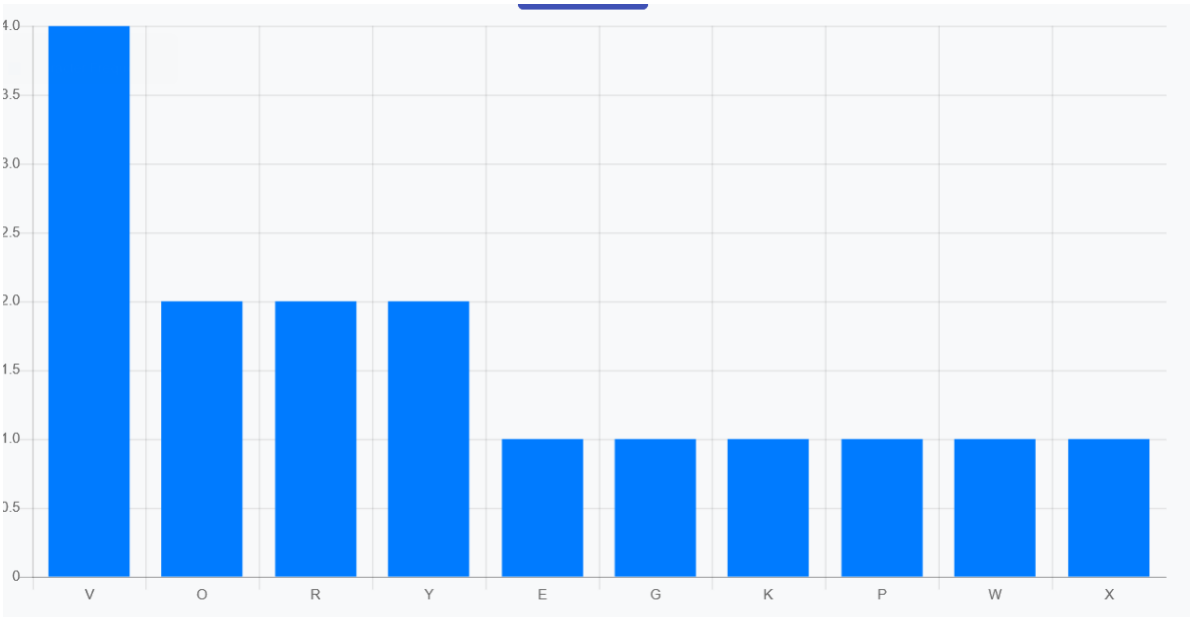
SHIFT VALUE = "10"

Cipher text once = ROVVY POVVYG REWKX

Shift value = "6"

Cipher text second = XUBBE VUBBEM XKCQD

BREAKING ONCE ENCRYTION:



Most occurring letters in the cipher text is v and second is o

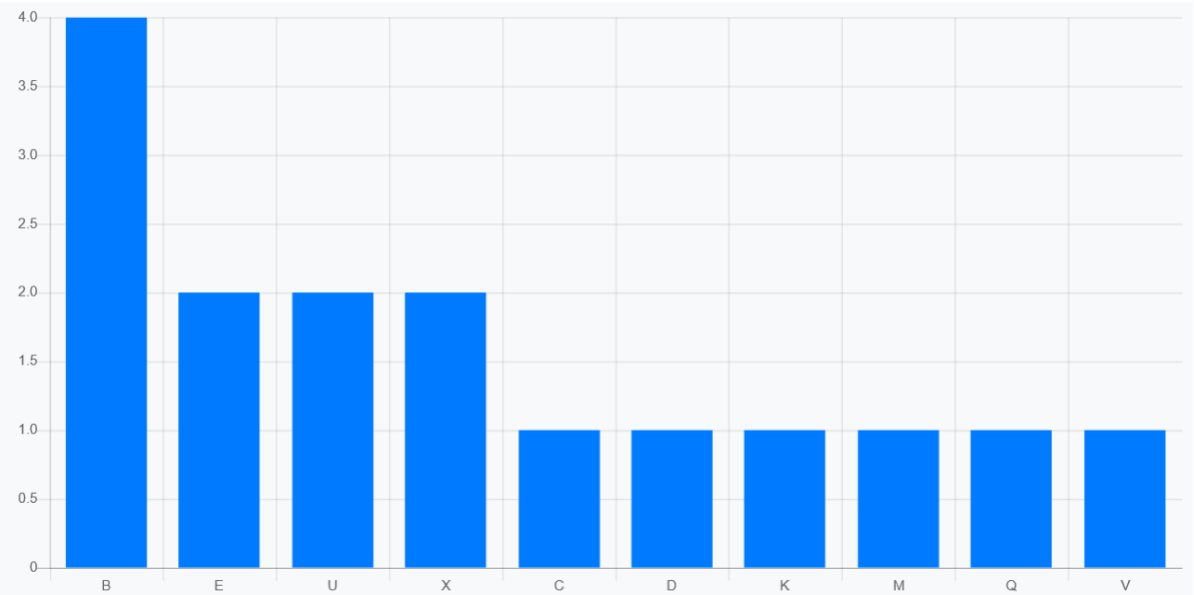
Since e is most common in English it will most likely occur within top two

Likely shift value is 17 for v - > e or 10 for o ->e

Put both into a decoder

And the shift value of 10 gave a plaintext with English words “HELLO FELLOW HUMAN”

Breaking double encryption:



Even though the cipher text letters change the rate of occurrence of what each cipher letter relates to there plaintext letter does not meaning you could encrypt the plaintext multiple time but you could still break it with one shift value.

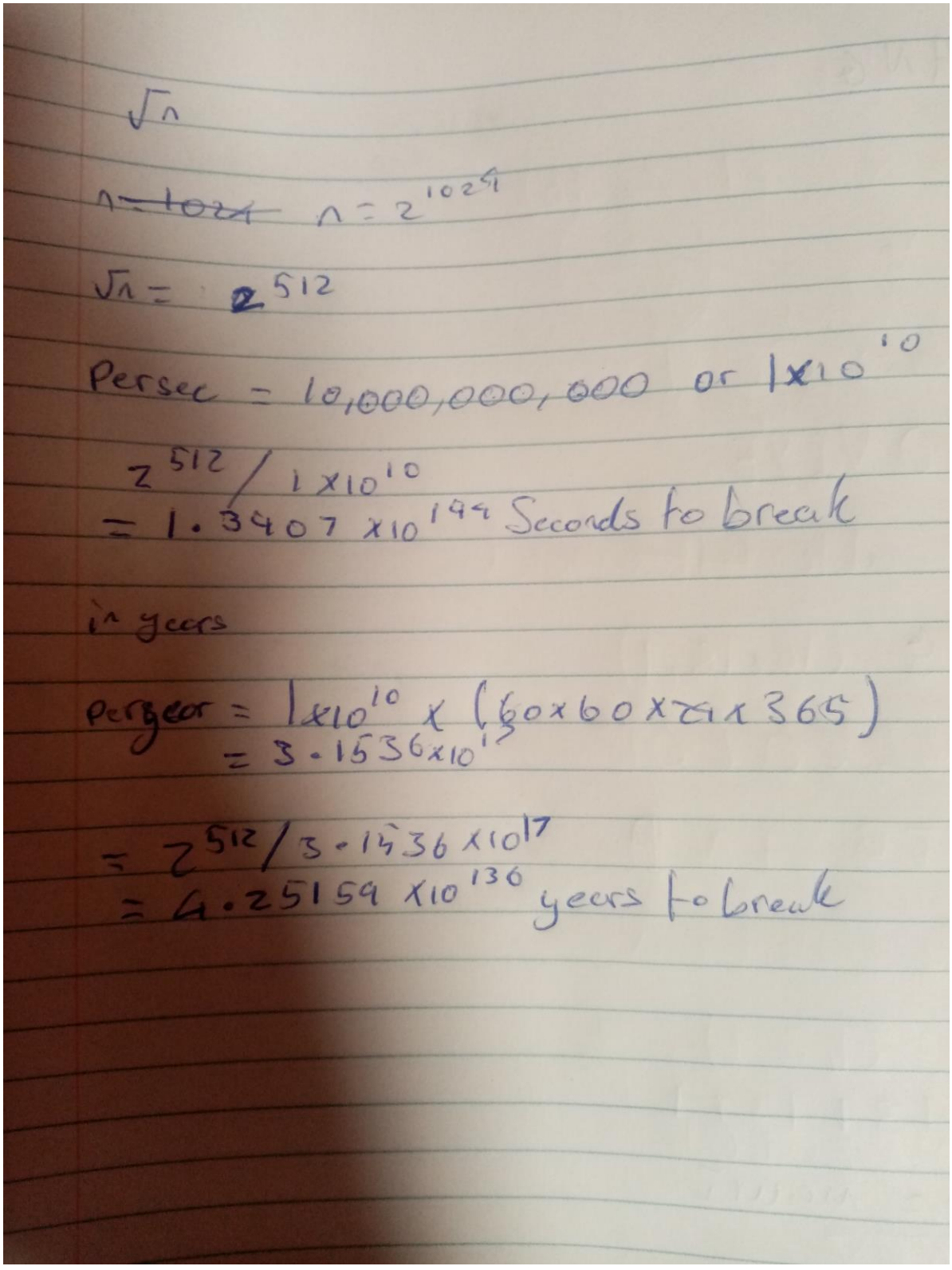
E is most likely to occur within the top 2 most occurring letters

In this case it is B, E,X,U

b->e is 3, e->e is 0, x->e is 19, u-> e is 16

I tried all of the shift values and “16” gave me the plain text, even though this was double encrypted the difference between the first and second encryption is nothing it was able to be broken with one shift value, therefore encrypting using a Caesar cipher more than once does not make is harder to brake.

20.


√n
n = 1024 n = 2^1024
√n = 2^512
Persec = 10,000,000,000 or 1x10^10
2^512 / 1x10^10
= 1.3407 x10^144 Seconds to break
in years
peryear = 1x10^10 x (60x60x24x365)
= 3.1536x10^17
= 2^512 / 3.1436 x10^17
= 4.25159 x10^136 years to break

21.

PKEWHNG PKNG UWHK PKEWHNG RUNU
-PKEWANG PKAM

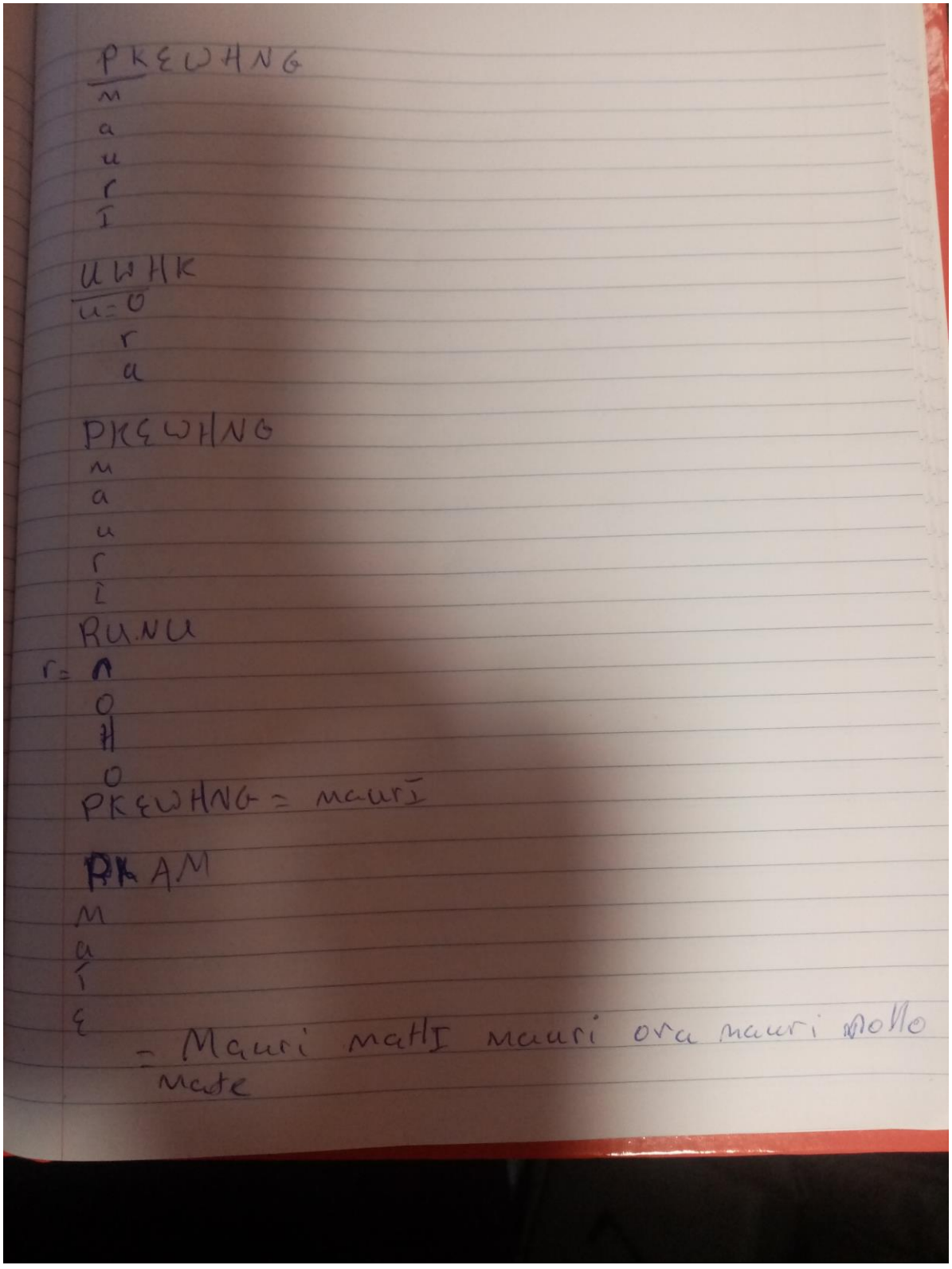
AM = IE
Shift = [illegible]

a = T
E = E

PKEWHNG
P = w
K = O
E = M
wh = I
ng = T
P = M
K = a
wh = r
ng = I

P = M
K = a
E = a
wh = r
ng = I

PKNG
m
a
H
AI
6



Shift value = 4

22.

Total keys = 40161 * 40161

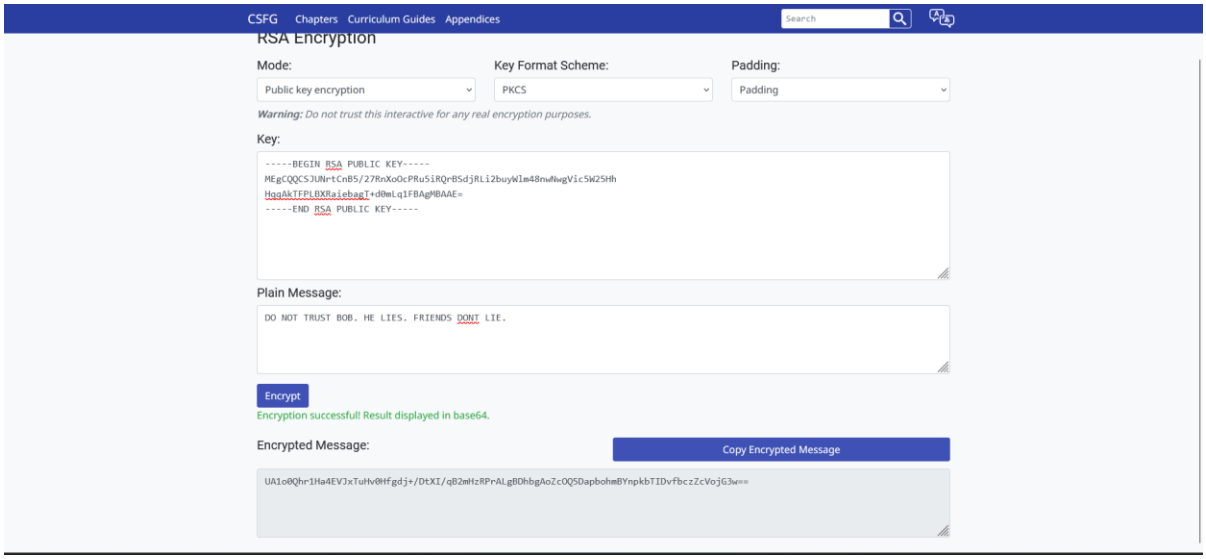
=1612905921

Tries per second = 1,000,000

Time to take in seconds = 1612905921/1000000

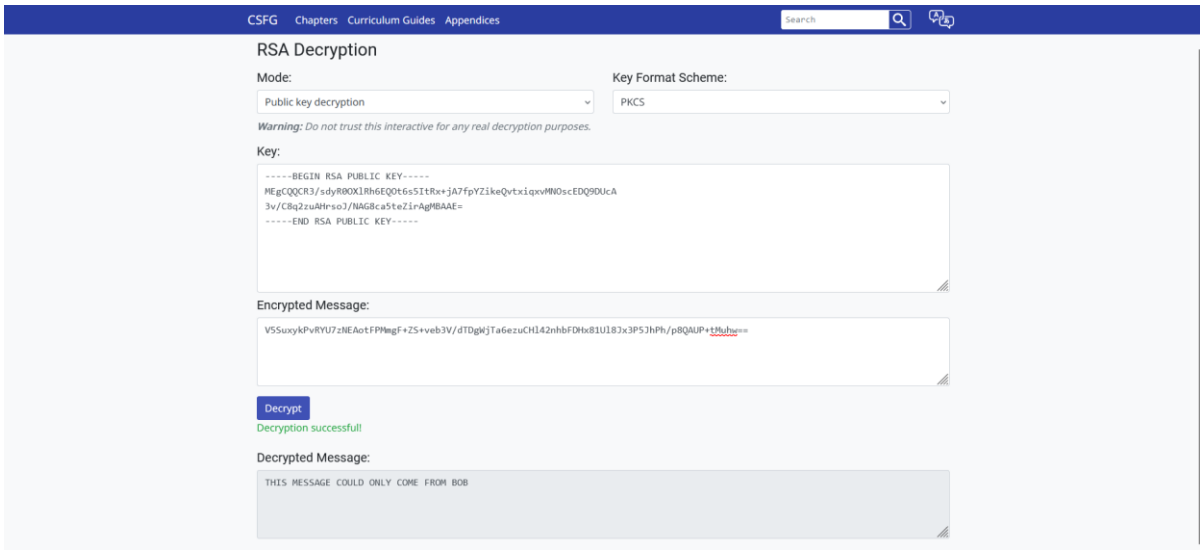
=1612.90 seconds

23.



By encrypting carols reply with Alice's public key it means that only Alice's private key can decrypt the message and since the private key is only with the owner and is not shared to anyone ous Alice is the only one that can read carols message.

24.



To make sure that it was bob sending the message he would've encrypted the message via his private key meaning that the only key that can successfully decrypt the message would be his public

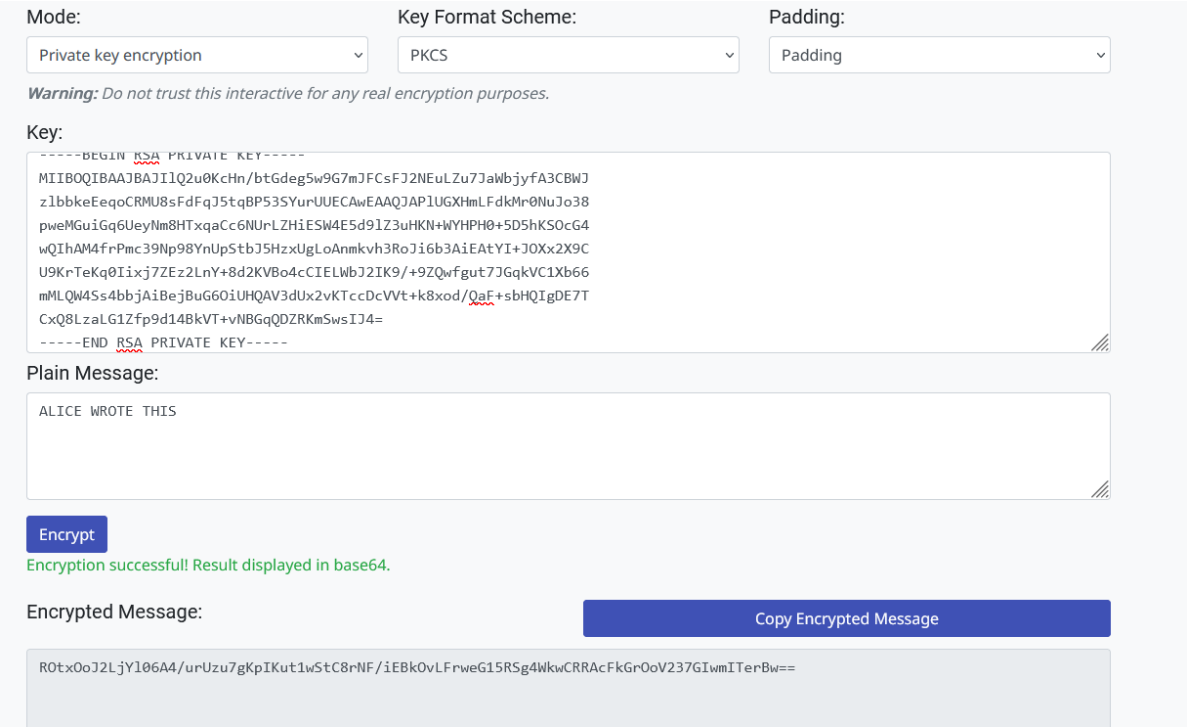
key which would prove that it is bob because he is the only one with his private key and since his public key did decrypt, this message was from bob.

25.

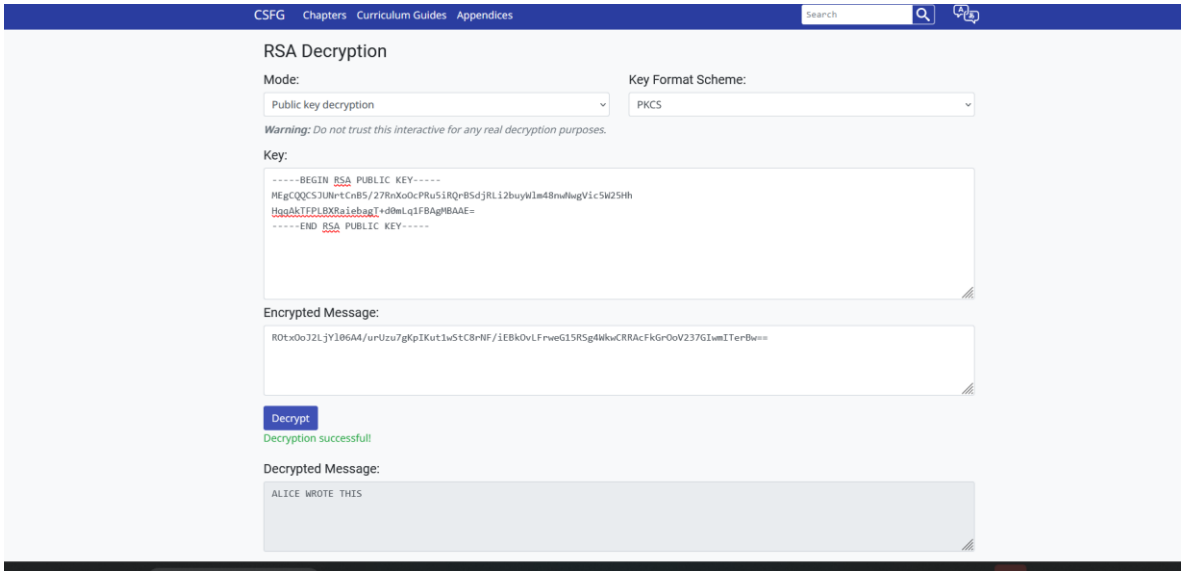
Alice would need to encrypt the message using her private key she would then need to share the corresponding public key to the other user, they would the decrypt the message using the provided public key. Alice would then need to calc the SHA2 hash of the original plaintext and provide the hash to the user who would also need to calculate the hash of their received decrypted message. The user would need then to compare the two hashes if the are the same this proves the integrity of the file because if it was tampered with in transit the hash would be totally different and since they decrypted the message using Alice's public key it means that it had to be encrypted using her private key. Both hashes match proving the integrity and validity of the file and that the message was decrypted using Alice's public key proves that Alice was the one whom sent this.

`Alices plaintext = ALICE WROTE THIS`

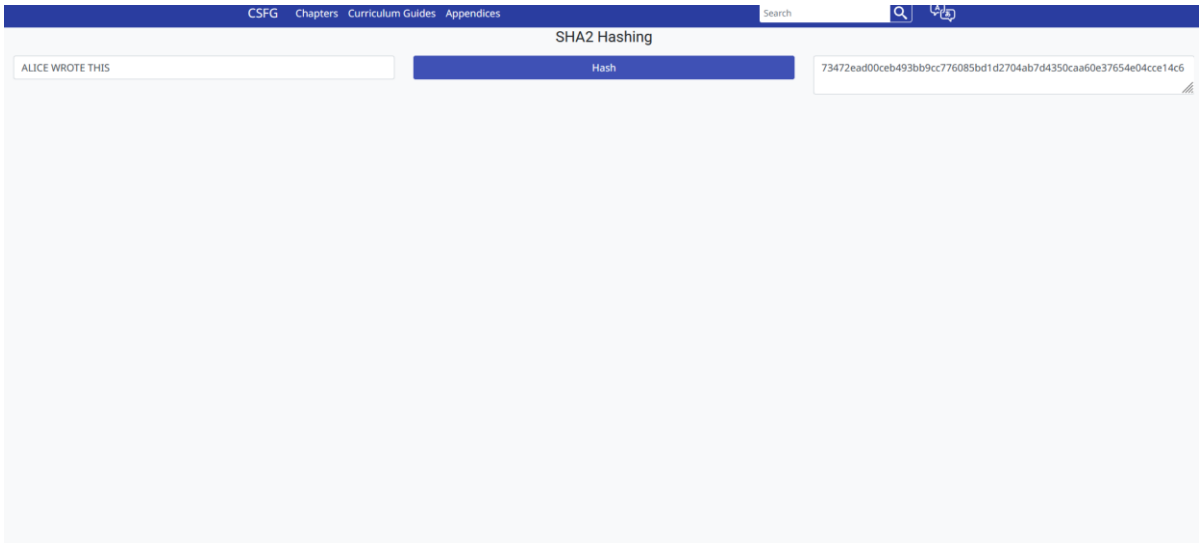
Encrypted using alices private key cipher text



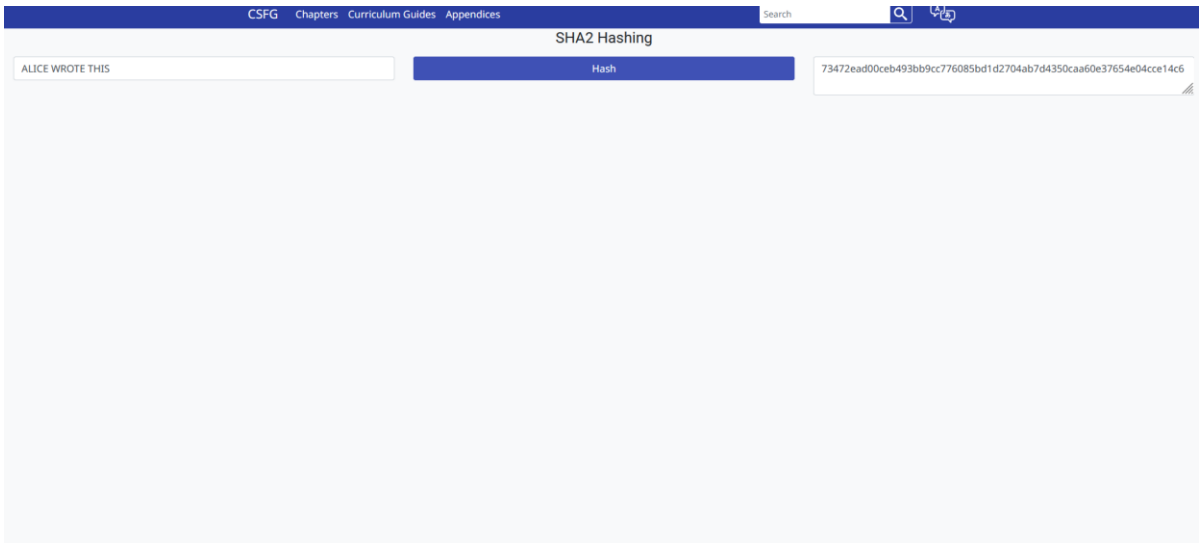
User decrypting using the public key



Alice calculating hash



Other user taking received decrypted message and finding SHA2 hash



Alice would then send hash to user and the user would compare the two

They find the hash from Alice matches their hash so that proves the integrity of the fill and since they used Alice's public key it means the message had to be encrypted with Alice's private key proving that it was Alice who sent this message.

26.

Recently a threat group known as TA558 have been targeting travel and hospitality organisations with malicious malware attacks. TA558 have been attacking and delivering malware to the organisations via social engineering attacks. Where they would send misleading emails to employees of companies via fake concerns of hotel reservations that come with an attached file that has a URL, RAR, or ISO attachment that once clicked would immediate start downloading and installing the malware on to the device containing various malware such as Async RAT or Revenge RAT. Stated by Pawan Kumar N malware such as Async RAT (Remote Access Trojan) allows the attacker to effectively take full control of the infected computer and would allow to take the system hostage for a ransom. TA558 is highly believed to be financially motivated and targeting and already stressed and weak system is trying to make money off it. This is crippling for organisations because this would stop them from doing any business not being able to get to customers due them not being able to access their system while held ransom crippling them. On CERT NZ there are ways to prevent attacks like this. The organisation could implement Security awareness building teaching their employees about potential ways they could be tricked to install malware via misleading emails, this would stop the hacker's ability to trick employees into downloading the ransomware because they are aware of these attacks. Another way is to implement application control which only allows for certain applications to be installed on the system, which would prevent the malware to even be installed allowing the organisation to be less permeable to these attacks.

The story -> https://threatpost.com/reservation-links-prey-on-travelers/180462/

CERT NZ -> https://www.cert.govt.nz/it-specialists/critical-controls/10-critical-controls/

Pawan Kumar N -> https://blog.qualys.com/vulnerabilities-threat-research/2022/08/16/asyncrat-c2-framework-overview-technical-analysis-and-detection

CHALLENGE

27.

28.

29.