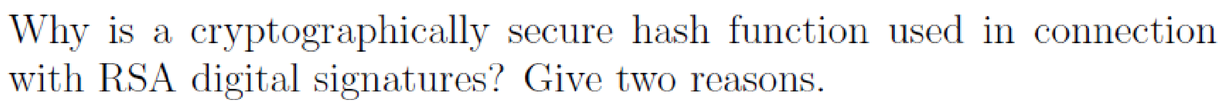
Uge 50

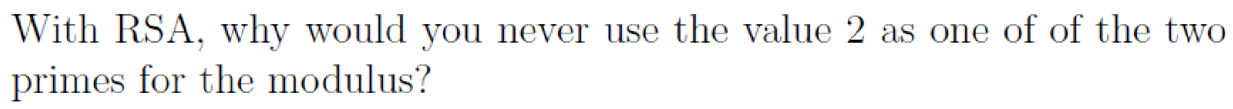
Ex 1


Why is a cryptographically secure hash function used in connection with RSA digital signatures? Give two reasons.

- Fixed message size
- Because we do not want to be able to easily find two messages hashing to the same value.

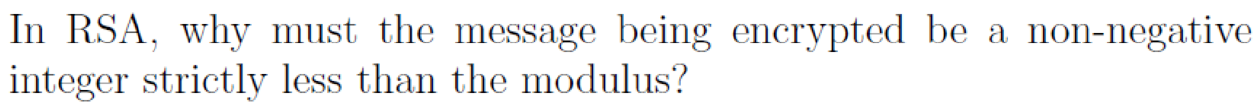
"cost of finding messages with same hash should be very high".

Ex 2


With RSA, why would you never use the value 2 as one of of the two primes for the modulus?

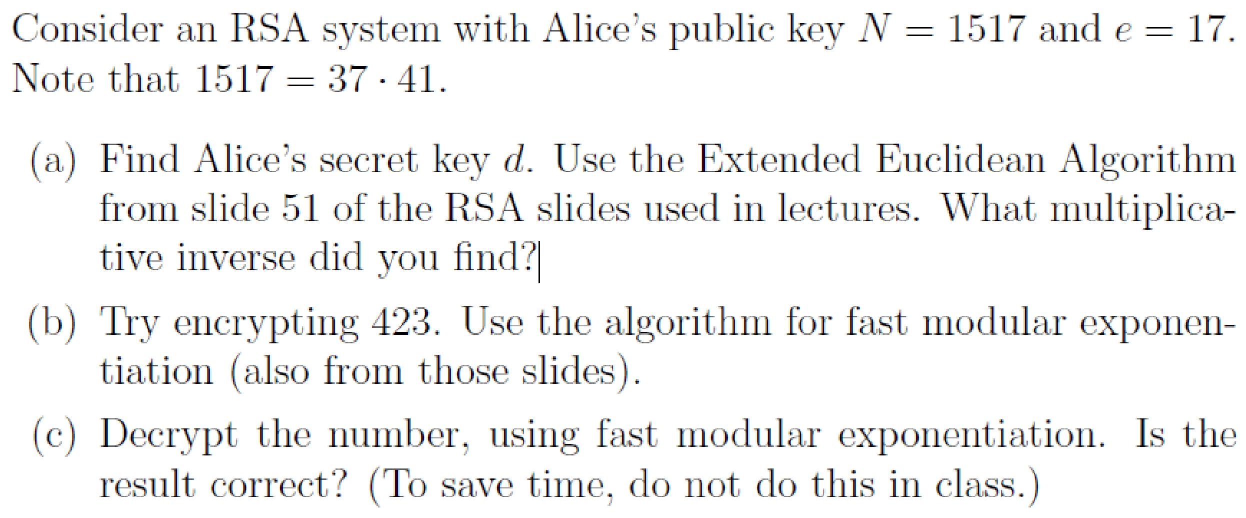
- Too small a number, easy to factor N, meaning easy to get the private key.
- Finding an *N* which is even would imply that one of the primes is a 2.

Ex 3


In RSA, why must the message being encrypted be a non-negative integer strictly less than the modulus?

- Modulus changes the sign to be positive, and if it was not smaller, it would wrap around, meaning we do not what we actually encrypted. The result when decrypting would be between 0 and N, where as the encrypted message is larger than N.
- Remember that we can "move the mod into the power" (fast modular exponentiation).

Ex 4


Consider an RSA system with Alice's public key $N = 1517$ and $e = 17$. Note that $1517 = 37 \cdot 41$.

(a) Find Alice's secret key d. Use the Extended Euclidean Algorithm from slide 51 of the RSA slides used in lectures. What multiplicative inverse did you find?

(b) Try encrypting 423. Use the algorithm for fast modular exponentiation (also from those slides).

(c) Decrypt the number, using fast modular exponentiation. Is the result correct? (To save time, do not do this in class.)

a. EEA:

to find out the private key we need to find d.

We know: $e_a \cdot d_a \equiv 1 \left(\text{mod}(p_a - 1)\,(q_a - 1)\right)$

We know: $17 \cdot d_a \equiv 1 \left(\text{mod}(37 - 1)\,(41 - 1)\right)$

We know: $17 \cdot d_a \equiv 1 \bmod 1440$

we have to find the multiplicative inverse of 17 mod 1440

We have e_a, p_a, q_a, and need to find the multiplicative inverse, meaning the number which multiplied by e_a is congruent to $1 \text{ mod}\left(\,(p_a - 1)\,(q_a - 1)\right)$.

we find gcd(17,1440) to find the s and t.

$\boldsymbol{n}$	$\boldsymbol{d_n}$	$\boldsymbol{q_n}$	$\boldsymbol{s_n}$	$\boldsymbol{t_n}$
0	1440	-	0	1
1	17	-	1	0
2	12	84	-84	1
3	5	1	85	-1
4	2	2	-254	3
5	1	2	593	-7
6	0	2	-1440	17

$$d_n = d_{n-2} \bmod d_{n-1}$$

$$q_n = \left\lfloor \frac{d_{n-2}}{d_{n-1}} \right\rfloor$$

$$s_n = s_{n-2} - q_n \cdot s_{n-1}$$

$$t_n = t_{n-2} - q_n \cdot t_{n-1}$$

$$s = s_{n-1}$$

$$t = t_{n-1}$$

$$\gcd(a, b) = d_{n-1}$$

we get:

$$s = 593$$

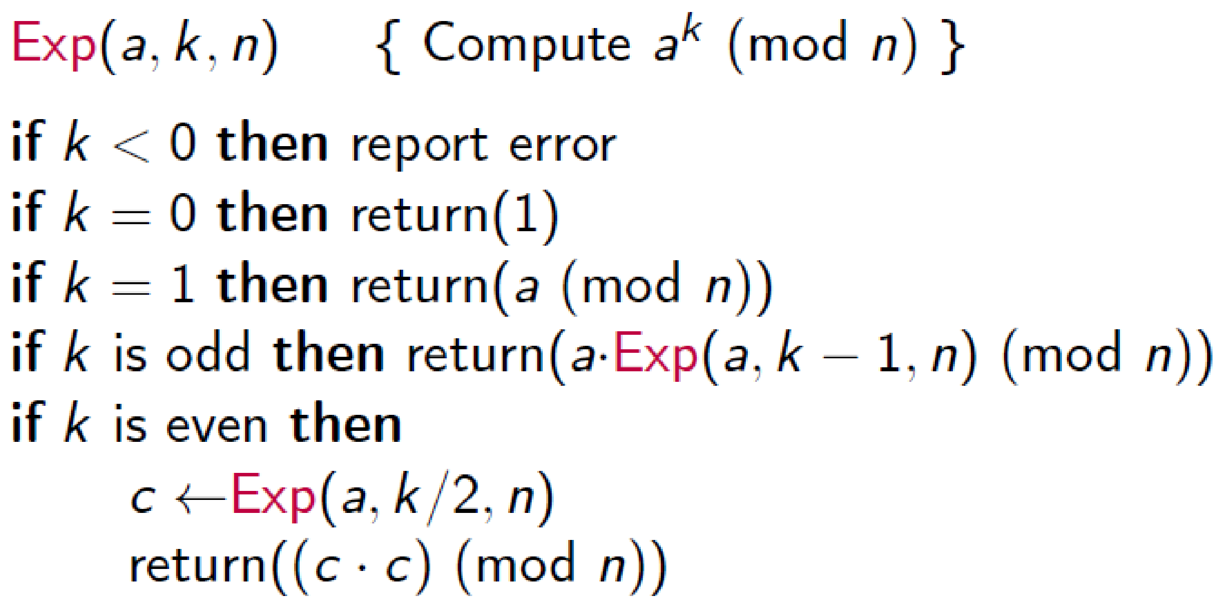
$$t = -7$$

This means that the multiplicative inverse is 593

b. encrypt 423 using fast modular exponentiation:

$$m^{e_A} (\text{mod } N_A) = 423^{17} \text{ mod } 1517$$

we use the following algorithm to encrypt:



```
Exp($a, k, n$)      { Compute $a^k \pmod{n}$ }

if $k < 0$ then report error
if $k = 0$ then return(1)
if $k = 1$ then return($a \pmod{n}$)
if $k$ is odd then return($a \cdot$Exp($a, k-1, n$) $\pmod{n}$)
if $k$ is even then
    $c \leftarrow$Exp($a, k/2, n$)
    return(($c \cdot c$) $\pmod{n}$)
```

we need to calculate exp(423, 17, 1517)

We do so by writing the algorithm in python:

---calculating 423^17 mod 1517

k odd -> return: 423 * exp(423, 17 - 1, 1517) mod 1517

---calculating 423^16 mod 1517

k even -> return: exp(423, 16/2, 1517)^2 mod 1517

---calculating 423^8 mod 1517

k even -> return: exp(423, 8/2, 1517)^2 mod 1517

---calculating 423^4 mod 1517

k even -> return: exp(423, 4/2, 1517)^2 mod 1517

---calculating 423^2 mod 1517

k even -> return: exp(423, 2/2, 1517)^2 mod 1517

---calculating 423^1 mod 1517

k = 1 -> return: 423 mod 1517

Following the result back:

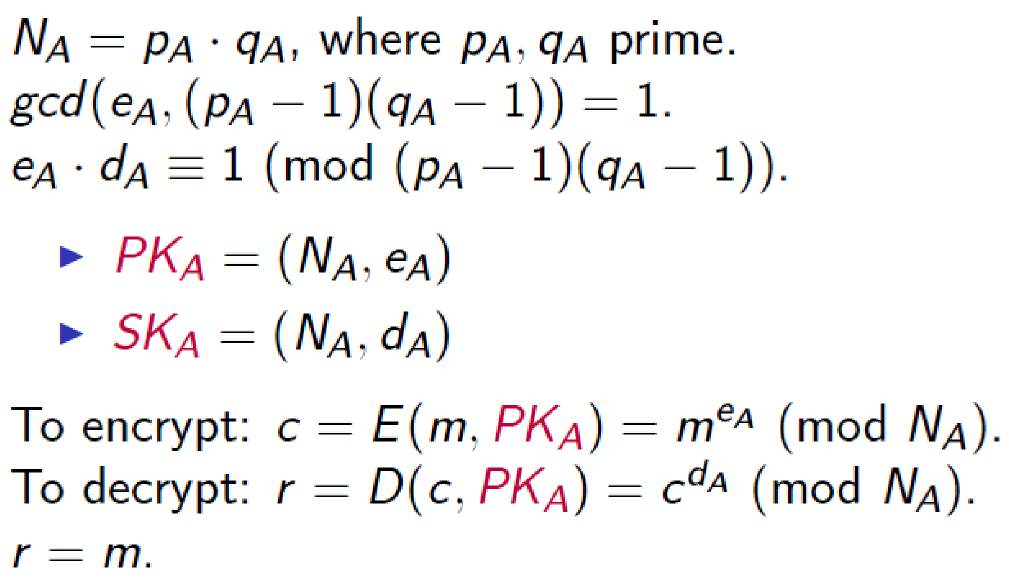
result 423^2 mod 1517

result 1440^2 mod 1517

result 1378^2 mod 1517

result 1117^2 mod 1517

result 715 * 423 = 562



$N_A = p_A \cdot q_A$, where p_A, q_A prime.
$gcd(e_A, (p_A - 1)(q_A - 1)) = 1$.
$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}$.

- $PK_A = (N_A, e_A)$
- $SK_A = (N_A, d_A)$

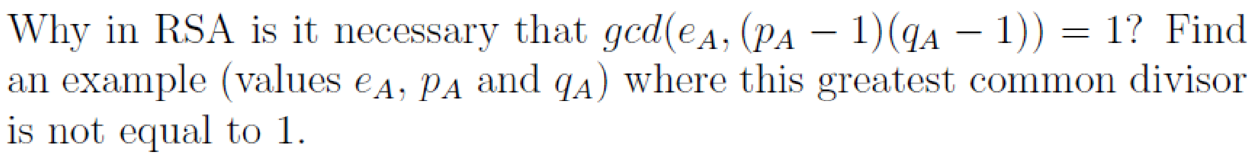
To encrypt: $c = E(m, PK_A) = m^{e_A} \pmod{N_A}$.
To decrypt: $r = D(c, PK_A) = c^{d_A} \pmod{N_A}$.
$r = m$.

c. decrypting using fast modular exponentation:

meaning calculating $c^{d_A} \pmod{N_A} = 562^{593} \pmod{1517}$

res = 423 yeaaaah.

Ex 5


Why in RSA is it necessary that $gcd(e_A, (p_A - 1)(q_A - 1)) = 1$? Find an example (values e_A, p_A and q_A) where this greatest common divisor is not equal to 1.

if the gcd is not 1, multiple messages can end up encrypting the same value.

$$e = 3, \qquad p = 13, \qquad q = 5$$

then $N = 65$ and $gcd(3, (13 - 1) \cdot (5 - 1)) = 3$

$$(13 - 1) \cdot (5 - 1) = 48$$

if you encrypt 2 you get:

$$2^3 \; mod \; 65 \; = \; 8$$

But if you encrypt 57 you get:

$$57^3 \; mod \; 65 \; = \; 8$$

Other problem, you cannot find d:

$$Slet\ definitioner:$$

$$e \cdot d = 1 + k \cdot (p - 1)(q - 1)$$

$$3 \cdot d = 1$$

$$3d - k \cdot 48 = 1$$

$$3(d - k \cdot 16) = 1$$

not possible!

generally:

$$X = (p - 1)(q - 1)$$

$$e \cdot d = 1 + k \cdot X$$

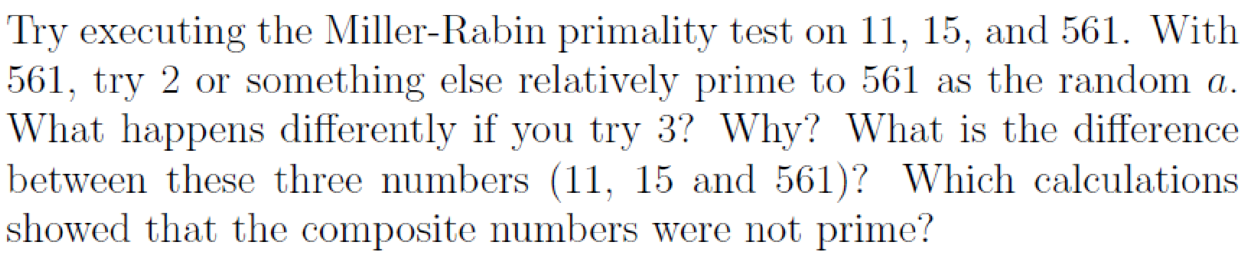
$$e \cdot d - k \cdot X = 1$$

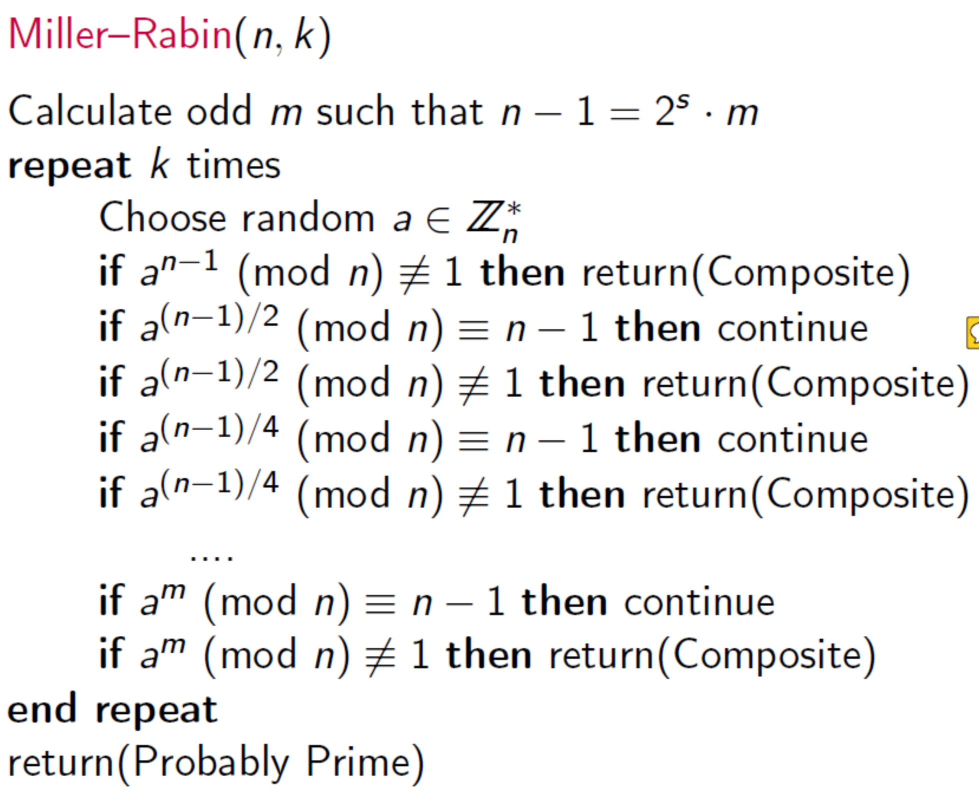
due to gcd not being 1 we have some c:

$$c\left(\frac{e}{c}d - k \cdot \frac{X}{c}\right) = 1$$

which can never be 1.

Ex 6


Try executing the Miller-Rabin primality test on 11, 15, and 561. With 561, try 2 or something else relatively prime to 561 as the random a. What happens differently if you try 3? Why? What is the difference between these three numbers (11, 15 and 561)? Which calculations showed that the composite numbers were not prime?


```
Miller–Rabin($n, k$)

Calculate odd $m$ such that $n - 1 = 2^s \cdot m$
repeat $k$ times
    Choose random $a \in \mathbb{Z}_n^*$
    if $a^{n-1} \pmod{n} \not\equiv 1$ then return(Composite)
    if $a^{(n-1)/2} \pmod{n} \equiv n - 1$ then continue
    if $a^{(n-1)/2} \pmod{n} \not\equiv 1$ then return(Composite)
    if $a^{(n-1)/4} \pmod{n} \equiv n - 1$ then continue
    if $a^{(n-1)/4} \pmod{n} \not\equiv 1$ then return(Composite)
        ....
    if $a^m \pmod{n} \equiv n - 1$ then continue
    if $a^m \pmod{n} \not\equiv 1$ then return(Composite)
end repeat
return(Probably Prime)
```

kører med $r = 2$

$$n = 11$$

Vælger et tilfældigt tal mellem 1 og n.

Vi vælger 8.

$$8^{11-1}\,mod\;11 = 1$$

Den fejlede ikke første test:

$$8^{\frac{11-1}{2}}mod\;11 = 10$$

vi fortsætter, men vi har gentaget r gange, så vi breaker. Og ved at det højest sandsynligt er et primtal.

Lad os teste $n = 15$:

Tilfældigt tal =7

$$7^{15-1}\ mod\ 15 = 4$$

Ved med sikkerhed at dette ikke er et primtal. Return 100% sikkert ikke primtal, da alle primtal vil ovenstående give 1.

Lad os teste $n = 561$:

$$tilfældigt\ tal\ a = 27$$

$$27^{560}\ mod\ 561 = 375$$

Dette er ikke et primtal. (obs er et Carmichael nummer).

Hvis vi gør det med det tilfældige tal 2:

$$2^{560}\ mod\ 561 = 1$$

$$2^{280}\ mod\ 561 = 1$$

$$2^{140}\ mod\ 561 = 67$$

Dvs. dette er sammensat.

hvis vi gør det med det tilfældige tal 3:

$$3^{560} \bmod 561 = 375$$

dvs. dette er sammensat.

11 er primtal, 15 er sammensat og 561 er Carmichael

if 2 is used then after 3 trials 561 is found to be not prime. 2 is coprime with 561.

if 3 is used then 561 is found to be not prime on the first test. 3 is not coprime with 561

Ex 7

Find four different square roots of 1 modulo 143 (numbers which multiplied by themselves modulo 143 give 1). Note that all of these numbers should be at least 0 and less than 143.

Use python to do this.

```python
print([i for i in range(0, 143) if i ** 2 % 143 == 1])
```

gives [1, 12, 131, 142]

Ex 8

Add two of these different square roots which are not negatives of each other modulo 143 (two where adding them together does not give 143). Find the greatest common divisor of this result and 143. Subtract these same two different square roots and find the greatest common divisor of this result and 143. (Think about why you get these results.)

$$1 + 12 = 13$$

$$gcd(13, 143) = 13$$

$$12 - 1 = 11$$

$$gcd(11, 143) = 11$$

we try 131 and 1:

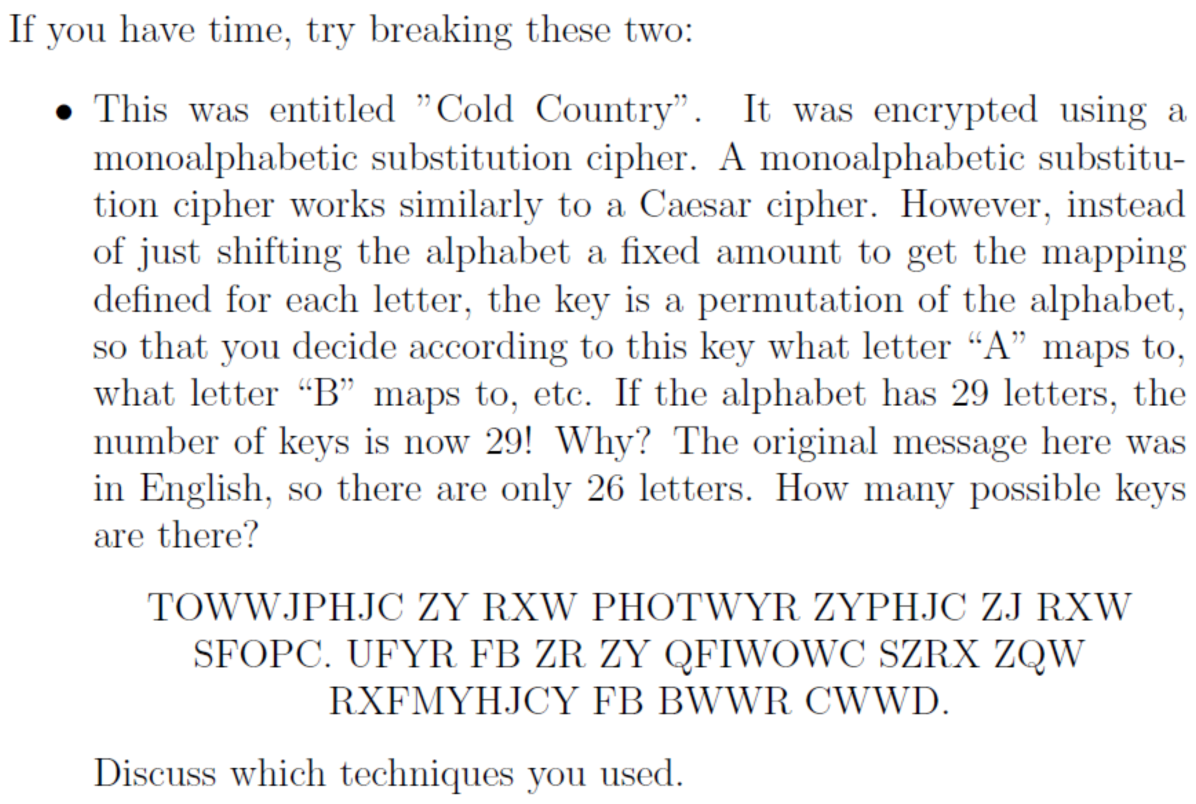
$$1 + 131 = 132$$

$$\mathrm{gcd}(132, 143) = 13$$

$$131 - 1 = 130$$

$$\mathrm{gcd}(130,143) = 11$$

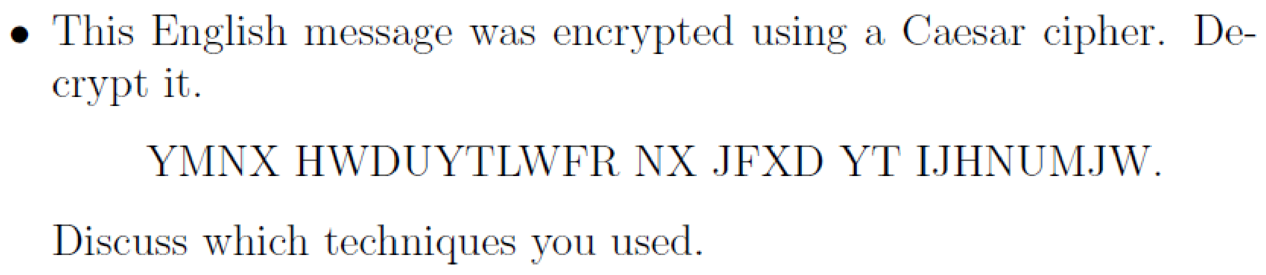
Ex 9


If you have time, try breaking these two:

- This was entitled ”Cold Country”. It was encrypted using a monoalphabetic substitution cipher. A monoalphabetic substitution cipher works similarly to a Caesar cipher. However, instead of just shifting the alphabet a fixed amount to get the mapping defined for each letter, the key is a permutation of the alphabet, so that you decide according to this key what letter “A” maps to, what letter “B” maps to, etc. If the alphabet has 29 letters, the number of keys is now 29! Why? The original message here was in English, so there are only 26 letters. How many possible keys are there?

  TOWWJPHJC ZY RXW PHOTWYR ZYPHJC ZJ RXW SFOPC. UFYR FB ZR ZY QFIWOWC SZRX ZQW RXFMYHJCY FB BWWR CWWD.

  Discuss which techniques you used.



- This English message was encrypted using a Caesar cipher. Decrypt it.

  YMNX HWDUYTLWFR NX JFXD YT IJHNUMJW.

  Discuss which techniques you used.

- Greenland is the largest island in the world. most of it is covered with ice thousands of feet deep.
- This cryptogram is easy to decipher

Methods: looked at small words, guessed using the hint (in the first).

The problem is the many possible assignments of keys. 26 factorial. So one could have a dictionary with the english language to check every decoding and then output the results if all words are found in the dictionary (or it is names). The many permutations can be reduced by finding the most recurring letters in the text and then guess permutations where these assign to frequent letters in the english language first (statistical analysis)