# 1)

## a)

Each Hi is 128 bits (block size). All together H0 H1 ... Hk  is therefore k\*128 (Not sure which its asking for).

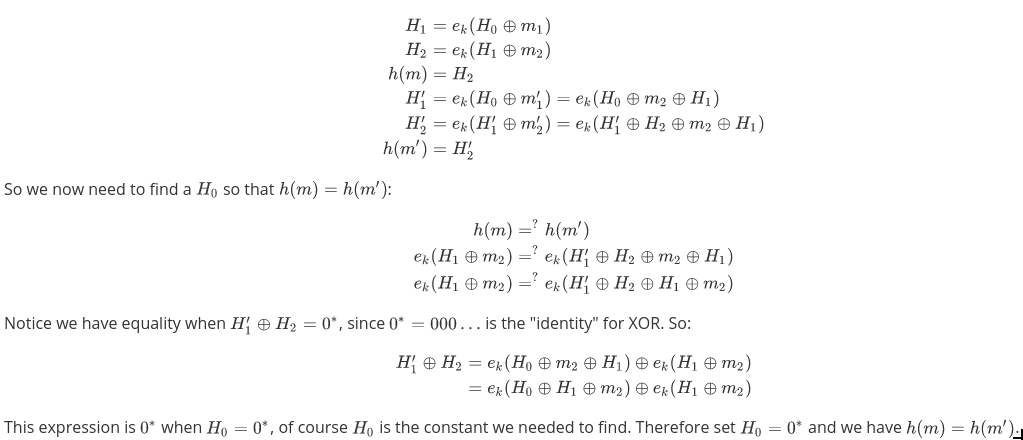
## b)

CBC - cipher block chaining

The ciphertext of CBC uses all the chained ouputs, not just the last.

i.e. h(m) = H0H1H2 ... Hk

## c) i.



Latex for (c)i. in case of edits

\begin{align\*}

H\_{1} &= e\_{k}(H\_{0} \oplus m\_{1}) \\

H\_{2} &= e\_{k}(H\_{1} \oplus m\_{2}) \\

h(m) &= H\_{2} \\

H\_{1}' &= e\_{k}(H\_{0} \oplus m\_{1}') = e\_{k}(H\_{0} \oplus m\_{2} \oplus H\_{1}) \\

H\_{2}' &= e\_{k}(H\_{1}' \oplus m\_{2}') = e\_{k}(H\_{1}' \oplus H\_{2} \oplus m\_{2} \oplus H\_{1}) \\

h(m') &= H\_{2}' \\

\end{align\*}

So we now need to find a H\_{0} so that h(m) = h(m'):

\begin{align\*}

h(m) &=^{?} h(m') \\

e\_{k}(H\_{1} \oplus m\_{2})

&=^{?}

e\_{k}(H\_{1}' \oplus H\_{2} \oplus m\_{2} \oplus H\_{1}) \\

e\_{k}(H\_{1} \oplus m\_{2})

&=^{?}

e\_{k}(H\_{1}' \oplus H\_{2} \oplus H\_{1} \oplus m\_{2})

\end{align\*}

Notice we have equality when H\_{1}' \oplus H\_{2} = 0^{\*}, since 0^{\*} = 000 \ldots is the "identity" for XOR. So:

\begin{align\*}

H\_{1}' \oplus H\_{2}

&= e\_{k}(H\_{0} \oplus m\_{2} \oplus H\_{1}) \oplus e\_{k}(H\_{1} \oplus m\_{2}) \\

&= e\_{k}(H\_{0} \oplus H\_{1} \oplus m\_{2}) \oplus e\_{k}(H\_{1} \oplus m\_{2})

\end{align\*}

This expression is 0^{\*} when H\_{0} = 0^{\*}, of course H\_{0} is the constant we needed to find. Therefore set H\_{0} = 0^{\*} and we have h(m) = h(m').

## c) ii.

m’ is calculated from the XOR with the output of the AES cipher, that is, H1 H2 . The whole idea of a cipher is that the ciphertext has no relation to the plaintext (or as much as this is possible). Therefore H1 H2 should have no relation to m1, m2, so m should have no relation to m’, so we can say it is very likely m, m’ are different, even if H0 is fixed as 0s.

For the security of h, this means we have two inputs for the same output. Ambiguous.

## c) iii.

Not sure how were supposed to change h whilst retaining eq (3), which already defines h???

But assuming we can change (3), maybe make it like CBC i.e.

h(m) = H0H1H2 ... Hk

Which does prevent this one collision.

## d)

Assuming input text is of **arbitrary length**.

Reversible:

Pad with zeros, and

Add block specifying length of unpadded message

Non-reversible:

Pad with zeros.

Assuming input text is of a **fixed length** (if not, split into blocks).

Reversible:

Pad with zeros (upto fixed length).

Non-reversible:

Pad with random values (e.g. pseudorandom generated data, with seed based on time/date)

# 2.

## a)

The key pair (N, e) is the public key.

The hash function h is shared.

So Alice knows all these three - that’s all that she requires.

## b)

r = cd mod N

K = h(r)

Checks (I assume only on c):

0<= c < N c != 0, 1 (maybe?)

## c)

r is randomly generated by Alice, so at this point only she knows it.

She communicates only c to Bob; this is the risk, as it could be intercepted by a MITM.

However, using RSA assumptions, this is safe. r can be found from c, but only if you have d.

So Bob can calculate r using d. They now both now r, and can calculate K.

K can now be used for symmetric encryption. With properties of h, this is not ambiguous.

Not sure how preimage resistance comes into it, but I guess if K were discovered, that does not provide r, so a new hash could be calculated.

## d)

First recall, if a hash functions output set is too small, preimage can become trivial by brute force.

If it’s not in [0, N), it better be bigger; too small output set will be a problem.

If it’s in [0, N) but with leading zeros there might be a problem; depends on how many zeros. If there’s a few, could be ok. Too many, and we have an issue.

# 3.

## a)

Not examined anymore right?????

# 4.

## a) i.

(A∧D)∨(A∧B∧C∧E)∨(B∧D∧E)

Given s as secret.

s = s1⊕ s2 = s3 ⊕ s4 ⊕ s5 ⊕ s6 = s7 ⊕ s8 ⊕ s9

where we can calculate s2 = s1 ⊕ s, s6 = s3 ⊕ s4 ⊕ s5 ⊕ s, s9 = s7 ⊕ s8 ⊕ s

all others randomly generated.

## a) ii.

sA = s1 ‖ s3

sB = s4 ‖ s7

sC = s5

sD = s2 ‖ s8

sE = s6 ‖ s9

## a) iii.

This set of parties B, C, D is not in the monotone access structure Γ.

This means together they do not have enough shares to combine them to find the secret.

## a) iv.

s2 is just one share of one of the three sets of parties that can recover the secret. Thus 2 are still left:

(A∧B∧C∧E)∨(B∧D∧E)

Combining either sets allows finding out the secret:

s = s3 ⊕ s4 ⊕ s5 ⊕ s6 = s7 ⊕ s8 ⊕ s9

However, s = s1⊕ s2 is now not possible since s2 is lost.

## b) i.

f1 = 25 f2 = 59 (randomly generated by me)

## b) ii.

f = 104 + 25 X + 59 X2

## b) iii.

X = {1,2,3,4,5,6,7}

f(X) = {81, 69, 68, 78, 99, 24, 67}

## b) iv.

e.g. ry[3] = -4 / (3-4) \* -5 / (3-5)

ry = {10, -15, 6}

s = (10 \* 68 -15 \* 78 + 6 \* 99) mod 107 = 104