# 1.

## a)

p(C=3) = p(P=b)p(K=k3) + p(P=c)p(K=k5) + p(P=d)p(K=k1) + p(P=e)p(K=k2) + p(P=e)p(K=k4)

= 0.2 \* (p(P=b) + p(P=c) + p(P=d) + p(P=e) + p(P=e))

= 0.2 \* (0.15 + 0.21 + 0.06 + 0.27 + 0.27)

= 0.192

p(C=5) = p(P=a)p(K=k2) + p(P=b)p(K=k5) + p(P=c)p(K=k4) + p(P=d)p(K=k3) + p(P=e)p(K=k1)

= 0.2 \* (p(P=a) + p(P=b) + p(P=c) + p(P=d) + p(P=e))

= 0.2 \* (0.31 + 0.15 + 0.21 + 0.06 + 0.27)

= 0.2

## b)

p(C=3 | P=a) = 0

Since a can never encrypt to 3, 3 is just not in the column of a

p(C=3 | P=c) = 0.2 (since it’s produced once by k5; then sum key probabilities)

p(P=c | C=3) = p(C=3 | P=c) p(P=c) / p(C=3)

= 0.2 \* 0.21 / 0.192

= 0.21875

## c) i.

p(C=3|P=a) =/= p(C=3)

or

p(P=c|C=3) =/= p(P=c)

## c) ii.

Swap a,k4 with e,k4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e |
| k1 | 4 | 2 | 1 | 3 | 5 |
| k2 | 5 | 1 | 4 | 2 | 3 |
| k3 | 1 | 3 | 2 | 5 | 4 |
| k4 | 2 -> 3 | 4 | 5 | 1 | 3 -> 2 |
| k5 | 2 | 5 | 3 | 4 | 1 |

That way all rows and columns have no overlapping assignments, so encryption is a bijection, and it’s equivalent to a OTP.

## d)

(One of many choices)

I’d use public key cryptography e.g. RSA with elliptic curves, to firstly transmit a session key.

Then use this session key for symmetric encryption e.g. AES, for the communication.

Symmetric encryption is much more efficient, simpler to implement, less constraints so it is preferred for communication. However, it has the key sharing problem; public key crypto solves this.

RSA, with elliptic curves reduces the inefficiency/overhead pure RSA requires.

# 2.

## a)

I’m assuming p is public.

(S1) Alice can compute this as clearly she has xA, m, and p. Computing modular exponentiation is fine, by doing repeated squaring and mod’ing at each iteration, this can be done efficiently.

(S2) Bob clearly has a, xB and p. Again mod exp is fine.

(S3) Alice clearly has xA, p. Computing a modular inverse is no problem, it can be done with the extended euclidean algorithm, since:

xA^-1 xA + k (p-1) = 1 (mod p-1)

<=> xA^-1 xA = 1 (mod p-1)

And the algorithm will be able to find the bezout coefficients, xA^-1 and k, thus providing the inverse. So this is efficient.

Then again the modular exp is fine.

## b) i.

a’ ^ {xB-1} = m (mod p)

## b) ii.

(S1) m ^ {xA} (mod p)

(S2) m ^ {xA xB} (mod p)

(S3) m ^ {xA xB xA-1} = m ^ {xB} (mod p)

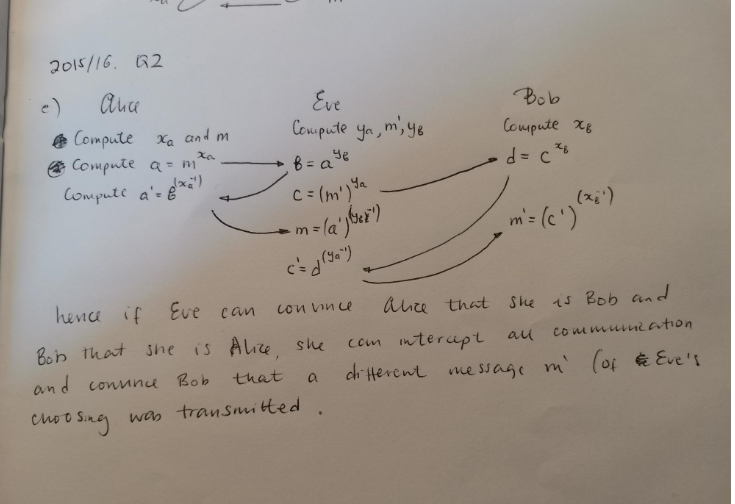
(S4) m ^ {xB xB-1} = m (mod p)

## c)

~~A MITM could steal anything transmitted, whilst knowing anything public.~~

~~a, b, a’ can all be stolen, equal to m ^ {xA}, m ^ {xA xB}, m ^ {xB} resp.~~

~~To find out m, which is the secret a MITM would want to know, they would need to compute the discrete logarithm of any one of those three values. Since that’s a widely assumed computationally infeasible task, this protocol is considered safe from MITM attacks (or unsafe, but only computationally, idk what they’re going on for)~~



## d)

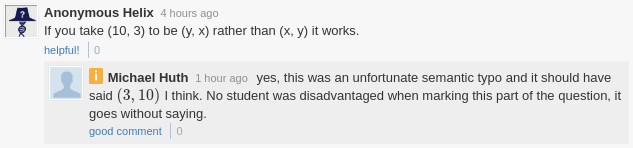
We’re doing exponentiation in the multiplicative group Zp \ {0}, sized p - 1 = 2 q by lagrange’s theorem, the subgroups of a group are its divisors. So the divisors here are 1, 2, q, so there’s subgroups of those sizes too.

We want to avoid the subgroups of size 1, 2 otherwise the arithmetic would be really easy for an attacker. The plaintexts which generate these subgroups are 1 and -1 == p-1 . They can be avoided by just removing them from the plaintext set, as P’ specifies.

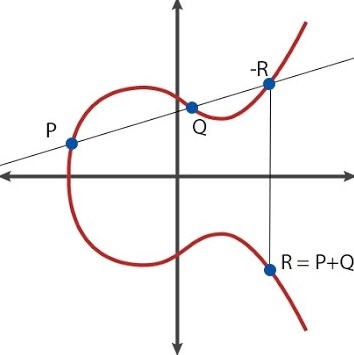
# 3.

## a) i.

Uhmmmmm????? It’s not?????



## a) ii.



## b) i.

K = k ★G

address: RIPEMD-160(SHA-256(K))

## b) ii.

Uhm well there’s two so that’s confusing

If they mean both, yeah, it means you can find private key given address - steal the cash  
If they mean one, well the other will maintain overall pre-image resistance so things are still safe for now.

## b) iii.

Private key is a random 256 bit number. You can just write a random number down and that’s a valid private key.

## c) i.

It’s still just an arbitrary hash, finding the preimage of it is still the same task.

## c) ii.

This can make certain addresses (e.g. for donations) harder to attack via attacks that replace the receivers address from the victim’s clipboard (I know it's handwavy). When the user copies an address that starts with 1donation... if the attacker replaces the charity address to a malicious address controlled by them it's going to be obvious to the victim. The attacker would have to generate a vanity address as well which is not easy.

# 4.

## a) i.

m(Γ) = {{B, C}, {C, D}, {A, B, D}}

Correct since all other sets are supersets of these

## a) ii.

(B∧C)∨(C∧D)∨(A∧B∧D)

Given s as secret.

s = s1⊕ s2 = s3 ⊕ s4 = s5 ⊕ s6 ⊕ s7

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

all others randomly generated.

sA = s5

sB = s1 ‖ s6

sC = s2 ‖ s3

sD = s4 ‖ s7

## a) iii.

They don’t have enough shares to rebuild s. They have s1, s5, s6, no combinment of which is s

## b) i.

Minimally non-qualifying:

{A, B}, {A, C}, {A, D}, {B, D}

compliments:

{C, D}, {B, D}, {B, C}, {A, C}

s = s1 ⊕ s2 ⊕ s3 ⊕ s4

sA = s4

sB = s2 ‖ s3

sC = s1 ‖ s3 ‖ s4

sD = s1 ‖ s2

## b) ii.

Yes, we only have {A, B, C}, {A, B, D}, {A, C, D}, {A, B, C, D}, which is unique.

Proof:

We consider all the possible subsets of P, and show there are 4, so the set of subsets is unique.

All sets must contain A (C1), so of the form {A, ...}. Now either a subset contains D (2) or it doesn’t (1):

(1) Assuming it doesn’t contain D; this means (S3) applies, so it must contain both B and C. Therefore the only option is {A, B, C}.

(2) Assuming it contains D; so we have the form {A, D, ...}. Now a subset must at least contain B or C (S2). So we have B, C, or both B and C added. This is three subsets: {A, D, B}, {A, D, C}, {A, D, B, C}.

So in total we have four options which are the subsets as outlined. Since it is only four, we have uniqueness.