1. answerShannons’s notion of perfect secrecy
2. Compute p(C=1) and p(C=5)

p(C=1) = 0.2 \* (0.17 + 0.22 + 0.21 + 0.19 + 0.21) = 0.2

p(C=5) = 0.2 \* 0.22 + 0.4 \* 0.21 + 0.2 \* 0.19 + 0.2 \* 0.21 = 0.208

1. Compute p(C=5 | P = c) and p(P=c | C = 5)

p(C=5 | P = c) = 0.2 + 0.2 = 0.4

p(P=c | C = 5) = p(P=c) \* p(C=5 | P = c) / p(C=5) (By Bayes)

= 0.21 \* 0.4 / 0.208 = 0.4038

1. Shannon’s theory
2. Why not perfectly secure according to Shannon?

* Only 1st precondition met (each k in K has equal probability of 1/|K|)
* The second precondition does not hold, i.e., there are not unique keys k for all c in C and all m in P
  + For example plaintext ‘a’ can be encrypted with two different keys to the same cipher text 3 (keys 2 and 4)

1. Corrected table - example

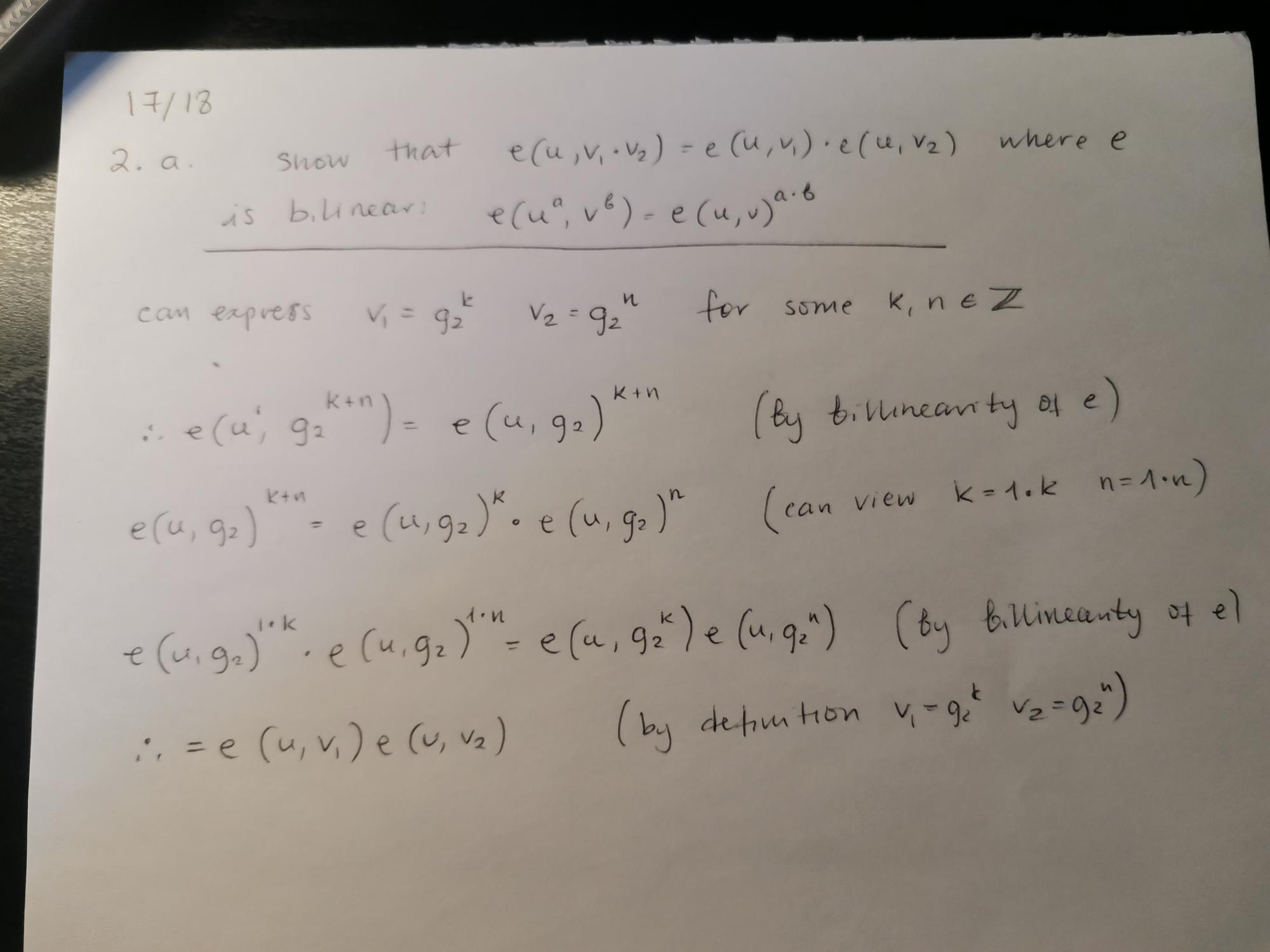
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **a** | **b** | **c** | **d** | **e** |
| **K1** | 2 | 4 | 1 | 3 | 5 |
| **K2** | 3-> 5 | 5-> 3 | 2 | 1 | 4 |
| **K3** | 1 | 4 ->5 | 5 -> 4 | 2 | 3 |
| **K4** | 3 | 2 | 5 | 4 | 1 |
| **K5** | 4 | 1 | 3 | 5 | 2 |

1. Why AES running in CBC mode of operation and using 256-bit keys is not perfectly secure but has much smaller size of K than a one-time pad would have in comparison

* With CBC mode, the randomisation is introduced with an initialisation vector (IV), which is used for encryption.
* Plaintext is XORed with the IV and the output is then encrypted with the key k. This is then fed into the encryption of the next block, creating a chain.
* In this mode, we don’t need to have a unique key for each c and m, IV creates the randomisation and repeated text gets mapped to different encrypted data.
* A 256 bit key is expanded into 15 round keys of size 128 bits

1. Public Key Cryptography over Bilinear pairings

a)



b)

i)

First a quick proof: If e is degenerate and bilinear, then e(. , .) = 1 for all inputs. Any arbitrary element of G1 can be written as g1a (cyclic group), likewise for any element of G2, g2b. Thus e acting on these arbitrary elements:

e(g1a , g2b ) = e(g1 , g2)ab = 1ab = 1

So e(. , .) = 1 for all inputs.

Now for the problem:

A digital signature scheme is insecure if we can tamper with a message, and it still will be accepted. So consider we sent m, which becomes m’ after tampering (then hashing they become t, t’).

To verify it, we check if:

e(g1, σ) =?= e(v, t’)

Now if e is degenerate & bilinear, e(. , .) = 1 for all inputs - as we’ve proven. So

e(g1, σ) = 1 = e(v, t’)

Thus the scheme will accept; as they are the same. But it’s accepting after tampering => insecure.

ii)

The scheme accepts if:

e(g1, σ) =?= e(v, t) <=> e(g1, tx) =?= e(g1x, t)

Now since e is bilinear

e(g1, tx) = e(g1, t)x = e(g1x, t)

So if we use the original m, and therefore t, we have equality and the scheme accepts.

c)

First consider taking an element of G1, g, to a power x. This is gx. But since the group operation is addition, gx = g + g + g + ... = xg.

Now to show bilinearity, we need that:

e(ua, vb) = e(u, v)ab with u ∊ G1, v ∊ G2

Now ua is within G1, so as we said, it is actually just au . Thus:

e(ua, vb) = e(au, vb) = (vb)au = vbau = (vu)ab = e(u, v)ab

d)

Either use 1 or p-1 is a generator of G1. Any primitive root is a generator of G2, let this be v. Now:

e(1, v) = v or e(p-1, v) = vp-1= 1 (by FLT)

So if we pick g1 as p-1, we have degeneracy; must avoid. But if we use g1 = 1, we’re ok; security is good in this case.

Security requires discrete logarithm is computationally infeasible. This requires really big p.

If this is the case, looks secure.

Modular exponentiation is easy to compute, so we have good availability.

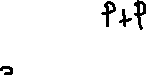
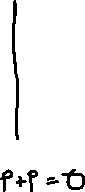
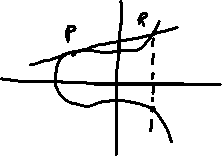
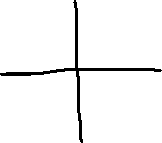
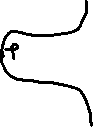
1. Cryptocurrency bitcoin
2. Elliptic curve E : Y^2 = X^3 + 7 and finite field F13
3. Is P = (11,5) in E(F13)?

Y^2 = 5 ^ 2 = 12 mod 13

X^3 + 7 = 5 + 7 mod 13 = 12 mod 13

LHS == RHS à point is on the elliptic curve

1. Draw a rough outline of the graph of Elliptic Curve E in the plane R^2 and illustrate in the definition of P + P



1. Second part
2. Briefly give an overview of how cryptographic primitives (well-established, low-level [cryptographic](https://en.wikipedia.org/wiki/Cryptography) algorithms, e.g., hash-functions) are used to generate Bitcoin addresses

Bitcoin address for private key k is calculated with a function that includes two hashes. First the public key is hashed with SHA-256 function that has 64 rounds of single steps and an output length of 256 bits. Then that hash is hashed with RIPDEM-160 function that has 5 rounds of 16 steps and output of 160 bits. Using two hash functions provides more protection if the one is compromised, the other one will still protected

1. Consider the programming language used in Bitcoin for transaction scripts. State whether the language is Turing complete, and briefly state the advantages inherent in that completeness or incompleteness

Making Bitcoin Turing complete would have meant having to provide for looping statements and even a possibility of infinite loops. For not allowing loops, a language makes it completely deterministic and you can know for sure when and how a program will end. You can’t end up in infinite loop, if you don’t have loops at all, and there is no need to worry about program getting stuck or crashing. As bitcoin is based on the logic that every node in the network needs to execute every script to ensure validity, it needs to be light-weight.

1. State an issue of private keys for Bitcoin addresses that creates friction between usability and security. Then describe a measure that mitigates that friction, still provides good usability and won’t compromise security
2. Explain why Bitcoin does not achieve final consensus but only temporary or eventual consensus

Transactions on public blockchains always have a probability of being reversed. It decays over time but is never zero. This means the finality is probabilistic. Each transaction needs to be confirmed by other nodes, and the nodes need to have a consensus on the valid chain of transactions. If the consensus would change after a transaction has occurred and recorded, the transaction can be reversed.

However, the probability of a transaction being reversed decreases when other blocks are added to the chain. Once a transaction has been included in one block it will continue to be buried under new blocks that are also confirmed by the network. This will consolidate the consensus as all the blocks would need to be reversed to reverse the transaction mentioned.

1. Assume that scalable quantum computers won’t be available before 2100. Briefly sketch a possible future of Bitcoin in 2019, justifying your predictions with scientific, economic or other evidence or contextual information

* Quantum computers could break the encryptions and help to calculate private keys by brute-force attacks
* It is projected to take more than 100 years before the bitcoin network mined its last token (max of bitcoins is 21 million)
  + Mining new bitcoins will become harder and harder and hence slower
* Miners will shift from mining new bitcoins to participate and validate new transactions, because each transaction has a fee attached to it
* This will resemble more a closed economy where transaction fees are like taxes
  + Could stabilise the market and make bitcoin a more stable transaction currency (steps have been taken towards this by large companies accepting crypto as a payments method – e.g., Paypal, Tesla)
* One implication of bitcoin is its impact on environment. Bitcoin infrastructure and transactions require a lot of computing power and energy. Currently, the energy usage is more than for Argentina, so there could be potential limitations due to the adverse environmental impacts.

## Question 4

Secret sharing schemes

S = student

I = Instructor

A = Administrator

E = external examiner

1. Comprehension of Monotone Access Structure
2. For the elements {A,E} and {A,S} of m(G), explain why they were chosen to be in G, and minimal at that
   * I assume they were chosen to make sure, that neither the external examiner nor the student should be able to access the exam papers independently. Student has an incentive to change the marking to a better one, and thus should not be granted those access rights (and would need a system administration to safeguard the grades). For external examiner on the other hand, I assume it was designed so that they should only evaluate the assessment done by the examiner and appeal for changes in the grade if they find traces of unfairness or inconsistencies (as opposed to independently changing the grades).
3. Critique two of the choices for elements in m(G) and suggest alternative minimal elements
   * The given m(G) suggest that E and I together can access the exam papers. I would assume they should be kept independent from each other to avoid any persuasion / changing of grades based on opinions of instructor or examiner. If the two disagree with grades it should be given to an admin / college body to determine the right grade.
   * So I would remove {I,E} from the set or change it to {I,E,A}
4. Consider another requirement: “An external examiner alone should be able to get access, but an external examiner together with a student should not get such access”

State whether this can be modelled by a monotone access structure?

* If E is a minimal qualifying set, as the statement suggests, all the sets containing E are also qualifying sets (i.e. it is enough that the group has share se to gain access). Hence, it is not possible to have {E,S} as non-qualifying set in a monotone access structure (could work in non-monotone access structure).

1. Replicated Secret Sharing: m(G) = {{A,E}, {A,I}, {A,S}, {E,I}}
2. Compute all maximal non-qualifying sets O of G, and use these to derive the shares for each P in {A, E, I, S}
   * Max non-qualifying sets:
     + O1 = {A}
     + O2 = {E,S}
     + O3 = {I,S}
   * Complements
     + B1 = {E,I,S}
     + B2 = {A,I}
     + B3 = {A,E}
   * Shares
     + Sa = s2 || s3
     + Se = s1 || s3
     + Si = s1 || s2
     + Ss = s1
3. If S and I collude
   * They know s1 and s1 || s2
   * They know how the secret is constructed s = (s1 XOR s2) XOR s3
     + Even though the attacker knows the first part (s1 XOR s2), if we assume s3 was generated truly random, the secret is still perfectly secure.
     + (Solving for s is like solving for a one-time pad)
4. Threshold Secret Sharing

Suppose that P has 100 elements and G consists of all subsets of P that have more than 50 elements. State which secret sharing scheme you would use in order to avoid very long shares.

I would use Shamir Secret sharing with a threshold of 50 elements. In this scheme the secret sharing relies on polynomial interpolation, where the secret is encoded into a polynomial and each party is given a data point in the polynomial. Each share is then just a point in polynomial making, by which we can avoid very long shares. The secret can be recovered by at least 51 shares, which fits to this case where G consists of all subsets of P that have more than 50 elements.