Name- KUMAR NIKHIL
Roll no: 1806032
Program Code: UG_CS
Branch: CSE
Course title: Blockchain Technology
Course Code: CS6475
Exam Date: 19th May 2021

ANSWERS

Question-1

H → Hash function (hiding + puzzle friendly)

Puzzle Friendliness:

A Hash function is said to be puzzle friendly if for every possible n-bit output value y, if k is chosen from a distribution with high entropy.

Given: x and a highly-unlikely-and randomly-chosen y

It is difficult to find z such that $h(y|z) = x$ (but it should exist).

Now in the given case:

$G(z) = H(z)^{0} \,||\, z_{last}$

~~but~~ but it's already given H(z) is puzzle friendly, so for a given x and y it is difficult to find z in $H(y|z) = x$, so if we can't find z from H then it is difficult to find z_{last} as well and hence from G(z) we can't find z easily for the given x and y and it is also puzzle friendly.

Name- KUMAR NIKHIL
Roll no- 1806032
Program Code- UG-CS
Branch- CS1
Course Title- Blockchain Technology
Course Code- CS6475
Exam date - 19th May, 2021

Hiding: A hash function is hiding if secret value r is chosen from a probability distribution that has high entropy, then given $H(r||x)$ it is infeasable to find x.

Given: $H(r|x)$
Secret: x and a highly-unlikely and randomly chosen r

Hard to find y such that $H(y) = H(r|x)$
The hiding property should work for all plaintext spaces, even if your ~~att~~ plaintext space is $\{0,1\}$.
G_1 is not hiding for that plaintext space. That means if $x=0$ or 1 then for
$G_1(z) = H(z) || z_{last}$
$z_{last} = z$, so for given $G_1(z)$ last bit i.e z and hiding property doesn't hold for one bit numbers.

SHA-256 (Secure Hash Algorithm)

It is one of the cryptographic hash function which has digest length of 256 bits. It's a keyless hash function. It was developed by National Institute of Standards & Technology.
Five requirements for SHA256 :-
1) It is one way - cannot restore data from hash value
2) It is deterministic
3) Fast computation
4) The Avalanche Effect
5) Must withstand collisions

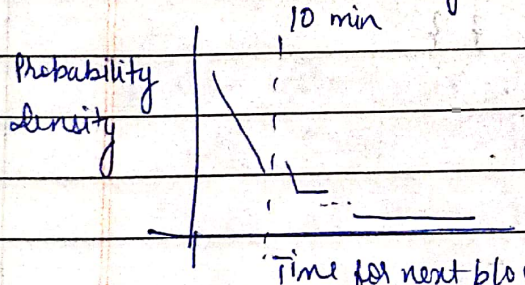
Name- Kumar Nikhil
Roll no- 1806032
Program code- UG-CS
Branch- CS
Course Title- Blockchain Technology Course Code- CS6475
Exam Date- 19th May'21

Question-2

a) No, the transactions that users create all require a digital signatures. Creating a valid digital signature requires the private key for the public key. Because that public key is fixed, the specific private key that the user has is required in order to modify the transaction.

An ISP doesn't have that users private keys. So they will be unable to produce a conflicting transaction as they will be unable to create a valid transaction. They cannot produce a valid signature, so a double spend cannot be made.
The only thing ISP can do is censor a users transactions.

If a block is found now then the next block will be found soon and there is a small probability that it will take a long time to find the next block.



Mean time to find a block = 10 minutes / fraction of hash power.

∴ Probability to find the next block in next 10 min = More than 50%

Name- KUMAR NIKHIL
Roll no- 1806082.
Course code- CS6475

b) It is a poisson process

Confidence process = 99% = 0.99
lambda = 6

Therefore $P(X=x) = \frac{(e^{1(-0.99)}) * (0.99^{\wedge}6)}{6!}$

evaluating we get,
$x = 0.485\%$

Question-3

E-VOTING

a) // structure, variable and ~~variable~~ contract declaration

contract Election {
// Model a candidate struct Candidate {
uint id; string name;
uint voteCount; }
constructor() public { addCandidate("candidate 1");
addCandidate("candidate 2"); } }

contract {

pragma solidity >=0.7.0 <0.9.0;
contract Ballot {
struct Voter {

Name – KUMAR NIKHIL
Roll no – 1806032
Course Code – CS6475

```
uint weight;
bool voted;
address delegate;
    uint vote; }
Struct Proposal {
    bytes32 name;
    uint voteCount; }
address public chairperson;
mapping (address => voter) public voters;
Proposal [] public proposals;
constructor (bytes32[] memory proposalNames) {
    chairperson = msg.sender;
    voters[chairperson].weight = 1;
}
function giveRightToVote (address voter) public {
    require ( msg.sender == chairperson);
require ( !voters[voter].voted);
require (voters[voter].weight == 0);
voters[voter].weight = 1; }
function delegate (address to) public {
    voter storage sender = voters[msg.sender];
}
function vote (uint proposal) public {
    voter storage sender = voters[msg.sender];
require (sender.weight != 0, "has no right to vote");
sender.voted = true;
sende.vote = proposal;
}
```

Name- KUMAR NIKHIL
Roll no- 1806032
Course Code- CS6475

```
function winnerName() public view
    returns (bytes32 winnerName_) {
    winnerName_ = proposals[winningProposal()].name;
    }
}
```

b) The main reason for expensive Bitcoin miner fees is supply and demand. The Bitcoin size is 1 MB which means miners can only confirm 1 MB worth of transactions for each block (one every 10 minutes) As a result miner fees sky-rocketed.

- Transaction creation and signing are the most expensive in the Bitcoin mining process and Bitcoin transaction fees are directly proportional to size of bytes of your transaction.
- Similarly sending bitcoin transaction is a lot like sending mail through postal service: If you send a large package, shipment will take longer and be more expensive.
- Bitcoins fee are ~~similar~~ sensitive to size of transaction.
- 'Merkle Tree' are used in Bitcoin to summarize all transactions in a block, producing an overall digital fingerprint for verification of transaction. This data structure speed up verification.

STEPS:
STEP1: Transaction creation and signing
STEP2: Broadcasting
STEP3: Propagation and verification
STEP4: Validation

Name- KUMAR NIKHIL
Roll no - 180632
Course Code - CS6475

Question-4

(i) Digital Signature Attacks :-

→ There are three types of digital signature attacks:-

(a) Chosen Message Attacks :-

It tricks the genuine users to digitally sign a message that user does not normaly intend to sign.

(b) Known-Message Attack :-

The attacker obtains some message that users sends and a key to create a new fault message and forge of user

(c) Key-only attack :-

In this it is assumed that user ~~name~~ make some info public and attacker misuse the public information.

Cryptosystems Attack :-

↙ (Cryptonalytic attacks) attacks the mathematical weakness in the algorithm.

↘ (Implementation attack) attacks the specific implementation of the cipher

→ Cipher text only attack :-

The attacker knows only the ciphertext to be decoded.

Name- KUMAR NIKHIL
Roll no- 1806032
Course Code- CS6475

→ Known - plaintext attack:-

The attacker has a collection of plaintext ciphertext pairs and is trying to find the key to decrypt some other ciphertext.

→ Chosen Plaintext Attack:-

Attacker chooses the plaintext to be encrypted and read the ciphertext.

→ Chosen ciphertext attack:-

Attacker can select any ciphertext and study the plaintext by decrypting them.

→ Chosen text attack:-

The attacker has the ability required in the previous two attacks.

(ii) RSA Scheme

Given, $p = 809$, $q = 751$

n is $p * q = 809 \times 751 = 607559$

$\phi(n) = (p-1)*(q-1) = 808 * 750 = 606000$

As private key $d = 23$

public key e will be:

$d * e = 1 \mod \phi(n)$

$23 * e = 1 \mod 606006$

$\therefore \boxed{e = 158087}$

Name- KUMAR NIKHIL
Roll no- 1806032
Course Code - CS6475

Message will be signed with private key of sender and verified by public key of sender.

(a) Signing of $M1 = 100$ as

$$S_1 = M_1^d \bmod n = 100^{23} \bmod 607559$$
$$= 223388$$

Verification of $S_1 = 223388$ as:-

$$M_1 = S_1^e \bmod n = 223388^{158087} \bmod 607559$$
$$= 100$$

(b) Signing of $M_2 = 50$ as:

$$S_2 = M_2^d \bmod n = 50^{23} \bmod 607559$$
$$= 5627$$

Verification of $S_2 = 5627$ as:-

$$M_2 = S_2^e \bmod n = 5627^{158087} \bmod 607559$$
$$= 50$$

(c) $M = M1 * M2 = 5000$

$$S = S_1 * S_2 = (223388 * 5627) \bmod 607559$$
$$\boxed{S = 572264}$$

$$S = M^d \bmod 607559$$
$$= 5000^{23} \bmod 607559$$
$$\boxed{S = 572264}$$

Hence Proved

END