SAT-2

① * **Hash function:**

→ Hash function is a mathematical function which transform the input (or "message") into a fixed-sized output string of bytes, typically a value called as hash value.

→ A general hash function has the following 3 properties

1) **Input flexibility:** Input can be of any size and any string.

2) **Fixed-output size:** Output has a pre-determined fix length. For example, a hash function gives a 256-bit output hash value.

3) **Computational efficiency:** The hash function should find the hash value is a reasonable amount of tim i.e., it should be computationally efficient.

* **Cryptographic hash function:**

→ To make the hash value cryptographically secur three additional hash properties have to be followed

1) **Collision resistant:**

→ A hash function is said to be collision resistant it is infeasible to find two distinct inputs x and such that $x \neq y$, but $H(x) = H(y)$.

→ Since the ~~of~~ input is infinite and the output finite, it is ensured that multiple inputs have th same output.

→ For example, for a 256-bit output string, ther are 2^{256} possible outputs. Hence, if we take $\underline{2^{256}+1}$ input strings, it is guaranteed to have a pair of inputs mapping to the same output.

→ It will require a computation of 2^{256} but accordi to birthday paradox, only the square root of computations, approximately 2^{130} are required and it is guaranteed to have a collision with 99.8% probability.

2) Hiding:

→ A hash function is hiding for $\underline{y = H(x)}$, if it infeasible to find the input x.

→ It is required that the x is spread out, that it is not easily predictable.

→ If x is not spread out, choose a random r with high min-entropy distribution such that $\underline{H(r \| x) = y}$, makes it difficult to predict x.

→ Min-entropy refers to the difficulty in predict the value & hence makes the hash function secu Example has $1/2^{256}$ probability.

3) Puzzle friendliness:

→ A hash function is said to be puzzle frien for a n-bit output y, and a random variables k with high min-entropy such that $\underline{H(k \| x) = y}$ it infeasible to determine the input x in sign less than $\underline{2^n}$ times.

→ For a search puzzle, there are multiple possible values of x, which requires exhaustive search.

→ Here, the puzzle ID id is chosen as the rand min-entropy value and concatenated with x.
$H(id \| x) = y$ is the hash function value &

→ If Y is the output set the target set of outputs which is small, it determining x difficult, if Y is large as muc n, determining x becomes trivial.

Example:

SHA-256 combines all these properties to ma hash function secure, provide data integrity & r

② . • Hash pointers:

→ Hash pointers combine 2 things: a pointer previous block and a cryptographic hash of it contents.

→ Hash pointer can be used ~~at a~~ instead regular pointer to detect tamper to data an data integrity.

→ There are 2 data structures that can be impl using hash pointers:

1) Blockchain: (Linked list with hash pointers)

→ Instead of using a regular pointer, a hash is used in a blockchain.

→ For a search puzzle, there are multiple possible values of x, which requires exhaustive search.

→ Here, the puzzle ID id is chosen as the random min-entropy value and concatenated with x

$H(id||x) = y$ is the hash function value & Y is the target set of outputs

→ If Y is the output set which is small, it makes determining x difficult, if Y is large as much as n, determining x becomes trivial.

Example:

SHA-256 combines all these properties to make th hash function secure, provide data integrity & reliabili

②. * Hash pointers:

→ Hash pointers combine 2 things: a pointer to the previous block and a cryptographic hash of its contents.

→ Hash pointer can be used ~~as a~~ instead of a regular pointer to detect tamper to data and mai data integrity.

→ There are 2 data structures that can be implemente using hash pointers:

1) Blockchain : (Linked list with hash pointers)

→ Instead of using a regular pointer, a hash poi is used in a blockchain.

→ It consists of 2 informations:

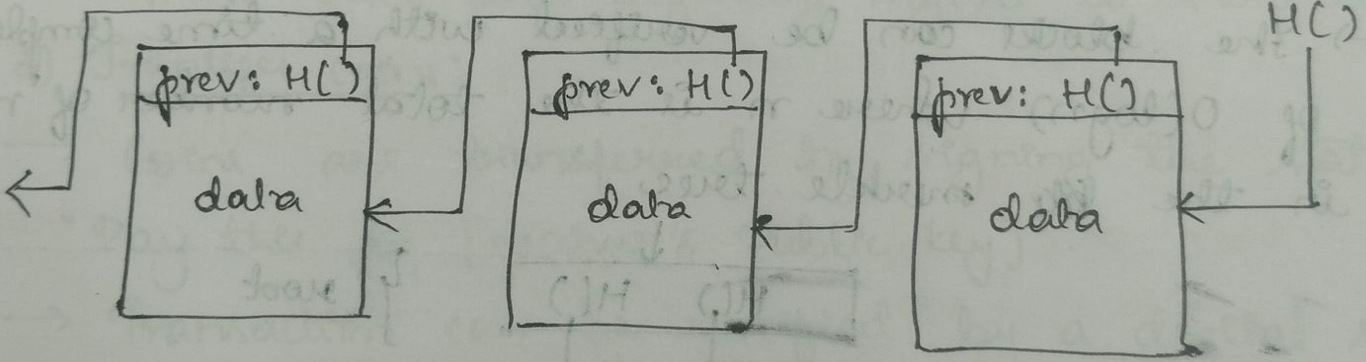
1) Pointer to previous block.
2) Cryptographic Hash of its contents.

→ Each block in the blockchain contains data and a hash pointer.

→ If an adversary tampers the data in block k, it will not match with the hash in block k+1, indicating tampering.

→ This tampering will be detected at the head of the list following the chained connections.

→ The head of the list points to the most recent block.



* Blockchain structure

2) Merkle tree: (Binary tree with hash pointers)

→ Here the regular pointers are replaced with hash pointers.

→ The data block forms the leaves.

→ Each parent consists of 2 hash pointers to its child nodes.

→ This pattern continues until a single root node is reached.

→ **Proof of membership:**

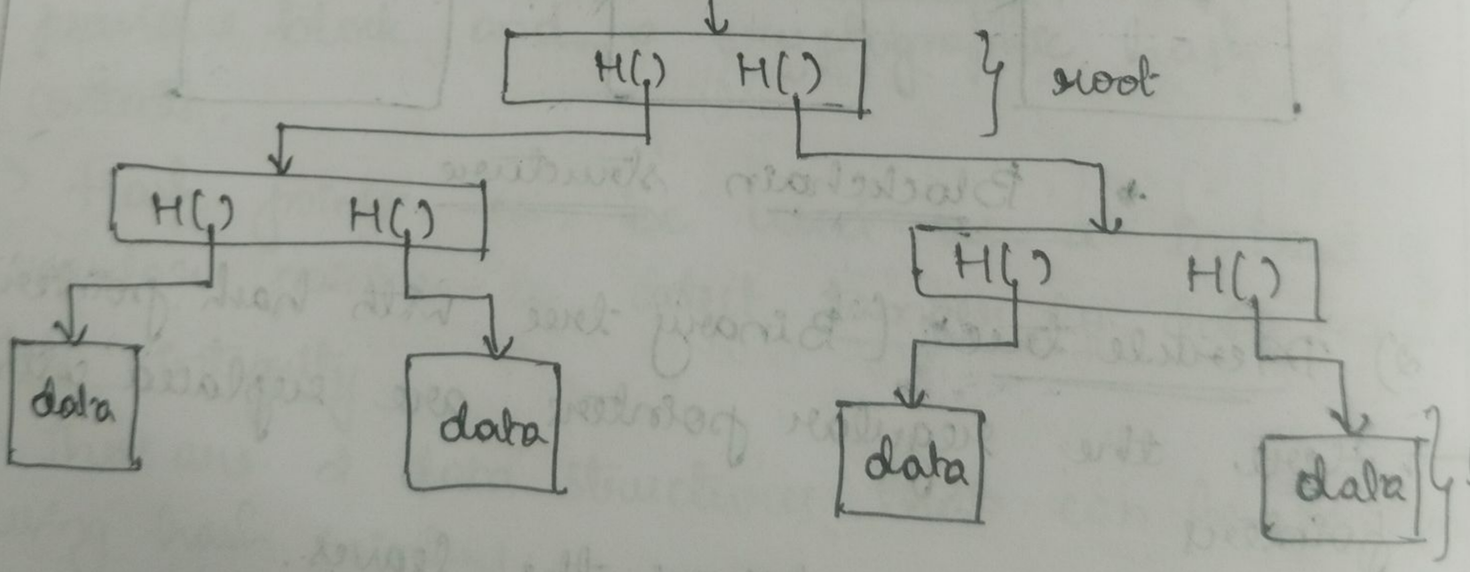
A block can be verified if it is a member of the merkle tree by following the hashes from that block to the root.

→ **Proof of non-membership:**

Using a sorted merkle tree, we can verify if a block is a part of the tree by verifying the bloc before and after that block.

If both the blocks are continuous, the queried block is not present.

→ The blocks can be verified with a time complexi of O(logn) where n is the total number of no in the merkle tree.



* Merkle tree structure

③. * Cryptocurrency types:

→ There are 2 types of cryptocurrencies:

1) GoofyCoin
2) Scrooge Coin

* Goofy coin:

→ It is the simplest cryptocurrency which consist of 2 types of transactions:

1) Create coins:

→ Coins are created by signing the statement createCoin [uniqueCoinID].

→ These coins are digitally signed to show its valid

2) Transfer coins:

→ Coins are transferred by signing the stateme "Pay this to [receiver's Public key]".

→ Transaction can be verified by a digital signat

→ Double spending attack:

During the transfer, it is possible to transfer th same coin to more than one recipient.

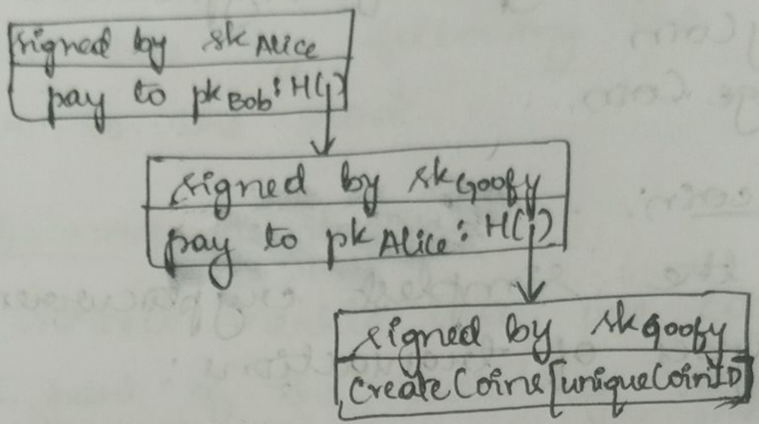
For example:

→ If Alice has 25 coins initially.

→ Then, Alice may sign 2 statements transferri the same coin to Bob and Chuck.

→ Now both Bob and Chuck have a valid cla to this coin.

→ This creates distrust and undermines the curren
system.



* Example for Double-spending in Goofy coins.

* Scrooge Coin:

→ It is an implementation of Goofy coin with a prevention mechanism for double spending attacks

→ These are the features of Scroogecoin that overco the drawback of goofycoin:

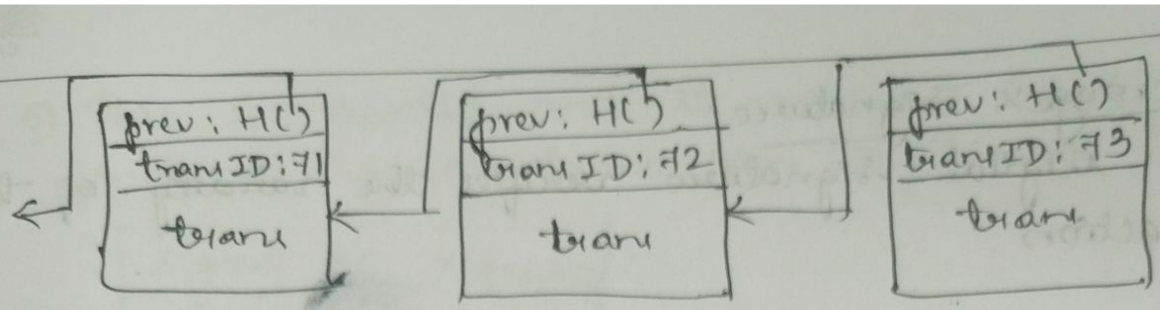
1) Append-only ledger:

→ There is a public ledger which has all transactio records.

→ It is append-only i.e., no modifications are allowe which makes the system transparent & secure

2) Blockchain structure:

→ It has the blockchain structure where each block points to the previous block with a hash pointe



* ScroogeCoin structure

3) Transaction type:

→ Create coins: It can create coins and assign initial owners.

transID: 73	type: CreateCoins		
Coins created			
num	value	recepient	
0	3.1	0x...	→ 73[0]
1	3.6	0x...	→ 73[1]
2	7.1	0x...	→ 73[2]

→ pay coins: It pays the amount by creating new co to the receiver and destroying spent coins.

transID: 7D	type: payCoins		
consumed coinIDs: 69[0], 80[4], 90[1]			
Coins created			
num	value	recepient	
0	3.1	0x...	→ 73[0]
1	3.6	0x...	→ 73[1]
2	7.1	0x...	→ 73[2]
Digital signatures.			

4) Validation rules:

→ Coins are created before being consumed.

→ Coins are not spent in previous transactions

→ The value of coins created is equal to those coins spent.

→ All participants (owners of the coin) sign the

5) Scrooge's signature:
The digital signature verifies the validity of the transaction.

→ Some drawbacks of Goofycoin are;

1) Distrust: Since coins can be easily duplicated, it is not a secure currency.
2) Devalues the currency: Since double spending is possible, the currency loses its value.

→ All the features mentioned above overcome the drawbacks of goofycoin.

④. * Double Spending attack;

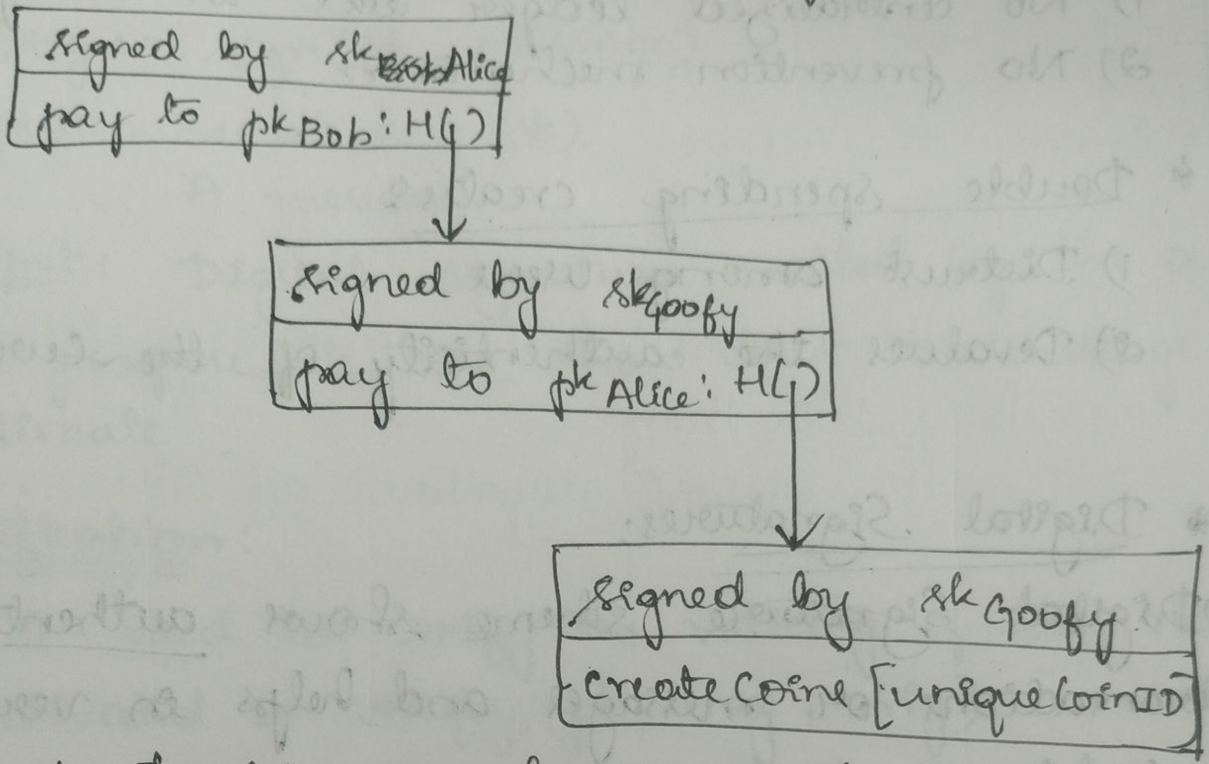
→ This attack is visible in goofy coin.

→ It occurs when the same coin is transferred to multiple receipients.

→ For example:

1) Alice has created a coin.
2) Transfers the coin to Bob by signing the transaction.
3) Now, without notifying anyone, Alic signs another transaction transferring the coin to Chuck.
4) This enables both Both Bob and Chuck to claim that coin

5) This is invalid and it makes it difficult to find the valid transaction / legitimate transaction.



* Double spending attack in Goofy coin

→ Here coins are created by Goofy.

→ The coins creation requires the signature of Goofy using Secret key.

→ This is paid to Alice with a valid digital signature of Goofy.

→ Now Alice pays this coin to Bob by signin the statement "Pay this to Bob".

→ Afterwards, Alice also pays it to Chuck

→ Now both have valid claims to this c

→ This is called Double - spending attack

* Double spending occurs due to:

1) No centralized ledger like in ScroogeCoin.
2) No prevention mechanism

* Double spending creates:

1) Distrust among users.
2) Devalues the authenticity of the currency.

⑤ * Digital Signatures:

→ Digital Signature scheme shows authenticity of the transactions or messages and helps in verifying the validity of the message.

→ There are 3 key algorithms:

1) Key generation:

→ It is used to generate a pair of keys:
(sk, pk) := generate(key size)

→ Input: key size is given as input.

→ Output: Secret key (sk): private & used for signing.
Public key (pk): publicly available & used for verification

→ It is necessary to process further steps.

2) Signing:

→ It is used to authenticate or sign the message.

sig := sign(sk, message)

→ Input: Secret key (sk)
A message.

→ Output: Digital signature created by the owner.

→ The purpose is the show that the message is legitimate.

3) Verification:

→ It is used to verify the validity of the message

isValid := verify.(pk, msg, sig)

→ Input: public key (pk)
A message (msg).
Digital signature (sig)

→ Output: isValid is a boolean value. If its tr
shows that the message is valid otherwise it
invalid.

→ The purpose is the verify the security and m
data integrity & reliability.

* Properties of Digital signatures:

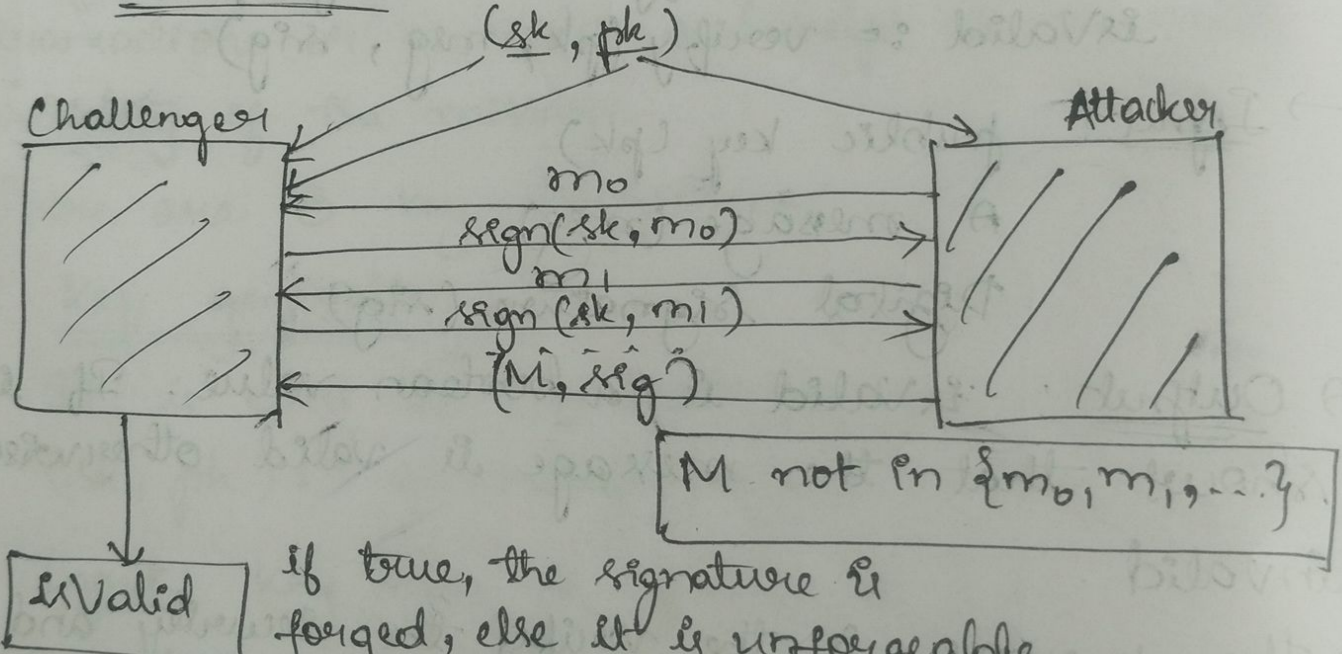
1) Validity: It is used to determine if the m
is valid and from a legitimate sender.

verify(pk, msg, sign(sk, msg)) = true.

2) **Unforgeability**: If the verification returns true, the message is unforgeable or it is forged.
It says, that it is impossible to determine the message without the digital signature & secret key.
A message can be signed only if the secret key is known.

It is not possible by knowing either:
- → The public key
- → Or the digital signatures of other messages.

* **Illustration**:



if true, the signature is forged, else it is unforgeable.

Setup:

* **Digital Signature Scheme**

A challenger creates 2 keys: secret (sk) and public (pk)
The attacker only has access to public key (pk)

Challenge:

The attacker sends a new message M alo with its digital signature created by the atta

Outcome:

→ If the verification of this signature returns true, it implies that the signature can be forged otherwise the signature is unforgeable.

→ It is intended that the Challenger signs the message sent by the Attacker to validate it.

Example: ECDSA (Elliptic Curve Digital Signature Algorithm).