14/Jan/20

Hash Functions

Key is the major factor in cryptography.

Public Key Cryptosystem

(A) PA PrA

(B) PB PrB — he can unlock it because he knows

(C)

Types of cryptography

Wallet associated to a user

Whenever you are about to receive it is going to generate ~~key~~ pub & priv. key

Whenever it has enough bitcoins it makes the block and put it above the chain.

Mining: Anyone can become.

Permission less Blockchain

17/Jan/20

RSA : deals with the integers of size 2000 bits

simple cryptosystem

1) P, Q → large primes {choose}
   2 nos

2) N = P * Q → (log P or log Q) {compute}

3) $\phi(N) = (P-1) * (Q-1)$ {compute}
   Euler's Function

$$\phi(n) = \left| \{ a \mid a \in Z^+, 1 \le a < n, a \nmid n \} \right|$$

4) choose e. such that $\gcd(e, \phi(N)) = 1$

5) compute d s.t.

$ed = 1 \bmod (\phi(N))$

↳ $\{ed = q\,\phi(N) + 1\}$

$a \equiv b \bmod d$.

$d \mid (a-b)$

$\exists\, q$ s.t

$d = q(a-b)$

$Z_N^2 \{0,1,2$

P

Z_N

Key Param

$K = (P$
get

Public

Privat

encryptio

$e_K =$ P

$x \in P$

decrypti

$d_K(y) =$

$\therefore$

$d_K(y)$ 2

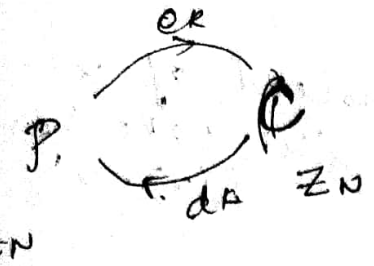
=)

q

y

y

$Z_N = \{0, 1, 2 \cdots N-1\}$



Key Parameters

$K = (p, q, e, d, \phi(N), N)$ — from them get it off

Public info : (e, N) — Encryption will use this

Private part : (d)

encryption :-

$e_K : P \rightarrow C$

$x \in P \quad e_K(x) \equiv x^e \bmod N$

$\equiv y$

decryption :-

$d_K(y) : y^d \bmod N \equiv x$

$d_K(y) : y^d \bmod N$

$\Rightarrow x^{ed} \bmod N$

$\Rightarrow x^{k(\phi(N))+1} \bmod N$

$\Rightarrow (x^{\phi(N)})^k \cdot x \bmod N$

$\Rightarrow 1 \cdot x \bmod N$

$\Rightarrow x \bmod N$

compute exponential sq. & mult. algo.

extended Euclids algorithm ↑ To compute d.

Euler's quotient. $(x^{\phi(N)}) \equiv 1$ always

Signature → msg, sig

(x, s)

$s = x^d \bmod N$

instead of x
i'll take hash
$h(x) = x'$

If $s^e \stackrel{?}{=} x \bmod N$
s is valid

Now send this

$s = (x')^d \bmod N$

if $s^e = x' \bmod N$.
s is valid.

Problem ?

(2) $\log_e^b = a$

$\log_2^4$

group $(S, *)$
↳ closed
Associative
id
inverse.

1) $(S, *, *)$ in Ab. group with id "0"

2) $(S - \{0\}, *)$ in Ab. group

3) $a * (b + c) = a * b + a * c$

$\forall\ a, b, c \in S$

Field

complex no, Rational no.

finite field

$Z_p, +_p, *_p$ are defined as prime p

$a +_p b = (a+b) \bmod p$

$a *_p b = (a*b) \bmod p$

$Z_2 = \{0, 1\}$.

$+_2$	0	1
0	0	1
1	1	0

corresponds to XOR operation

$*_2$	1
1	1

Ex:-

$(Z_3, +_3, *_3)$ is field with 3 elements.

$|Z_p^*| = p-1$ coz p is prime.

$a \in Z_p^*$ $a^{(p-1)} \equiv 1 \bmod p$ → Cardinality of the group

Legranges theorem

$Z_p^* = \langle g \rangle$ there will be relevent g in the group such that all power of 'g' will give you Z_p^*.

$$g^{p-1} \equiv 1$$

Discrete log problem.

$Z_7^* = \langle q \rangle$?

$Z_7 = \{0, 1, 2, 3, 4, 5, 6\}$

$Z_7^* = 6$

$6 = (x)^g$ $\quad a^{p-1}$

$5^1 = 5$

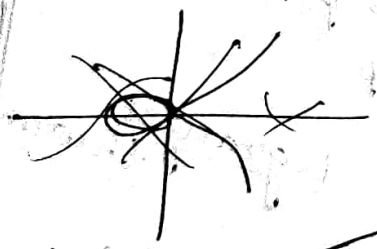
$5^2 = \frac{25}{7} = 4$

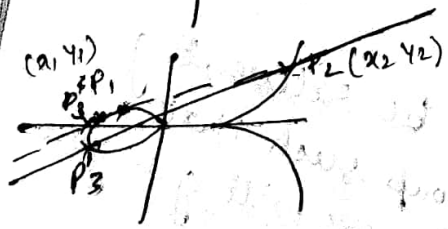
$5^3 = \frac{125}{7} = 6.$

$5^4 = \frac{625}{7} = 2$

$5^5 = 3.$

In '5' all the elemen are coming that are inside the set Z_7. So, that's the generator.





Whereever it will be cutting take the mirror image &

How to add these 2 points

$P_1 + P_2 = P_3$

$P_1 + P_4$

21/Jan/20

ElGamal Crypto System

y - public
x - private

$y = g^x \bmod p$

Based on difficulty of discrete learning algorithm

Alice

1. Generate a random prime p
2. choose an arbitrary integer 'a', which is a primitive root modulo p.
3. choose a random no. x from internal (1, p) which is co-prime to (p-1).
4. Calculate $y = a^x \bmod p$

The public key is a triple (a, p, y), the private key = x.

Alice

Encryption

Let M

① select a random secret no. 'k', which is coprime to (p-1).

② Calculate

$$V = a^k \bmod p$$

$$\sigma = M \cdot y^k \bmod p$$

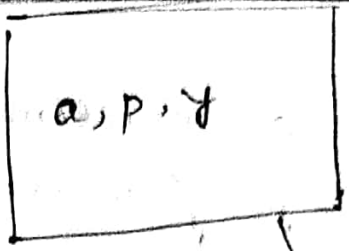
Ciphertext → (V, σ)

Decryption

we need to know here the private key. 'x', and then :-

$$M = (\sigma \times V^{p-1-x}) \bmod p$$

$$= V^{-x} \sigma \bmod p$$



(B) sender

(A) receiver

$M \in \{0, 1, \ldots, p-1\}$

$y = a^x \bmod p$

x – random value

$1 < x < (p-2)$

preliminary calculation

K-random $1 \le K \le (p-2)$

$\gamma = a^K \bmod p$

$\sigma = M \cdot y^K \bmod p$

$[\gamma, \sigma]$

$M = \gamma^{-x} \cdot \sigma \bmod p$

Elgamal signature scheme

$g \rightarrow$ generator

Key Generation

* choose randomly a secret key x with $1 < x < p-1$
* compute $y = g^x \bmod p$.
* Public key (p, g, y)
* Secret key (x)

Signature Generation Scheme

To sign a msg M, the signer performs the following steps

* choose a random 'k'
* compute $r = g^k \bmod p$
* compute $s = (H(m) - x \cdot r) k^{-1}$

The pair (r, s) is the signature of m.

*You can use any hash func" here.

Verification

A signature (r, s) of a m is verified as follows

* $0 < r < p$ and $0 < s \leq (p-1)$
* $g^{H(m)} \equiv y^r r^s \bmod p$.

$P-(x_1, y_1)$ $\quad x_1 \neq x_2$

$Q-(x_2, y_2)$

(1) $P+Q = (x_1, y_1) + (x_2, y_2)$

$= (x_3, y_3)$

$x_3 = \lambda^2 - x_1 - x_2$

$y_3 = \lambda(x_1 - x_3) - y_1$ and

$\lambda = \frac{(y_2 - y_1)}{x_2 - x_1}$

Case-2

$x_1 = x_2$, $y \neq y_2$

$(x, y) = (x, y) + (x, -y) = \Theta$.

28/Jan/2

Block Structure

Block Header

Merkle Tree

Bitcoin Mining

Transaction Script

Empty stack → True

① Push 2 ↑

② Push 3

ADD 5 EQUAL

③ ADD (pop) 2 & 3 push 5

5 EQUAL

④ 5 EQUAL ↑ Push

⑤ EQUAL Pop (5 & 5)

True

Locking script form

You store publickey in hashed form

↑

[duplication][compute Hash value] <cafe Pub. KeyHash>

OP_EQUALVERIFY, OP_CHECKSIG.
Verification

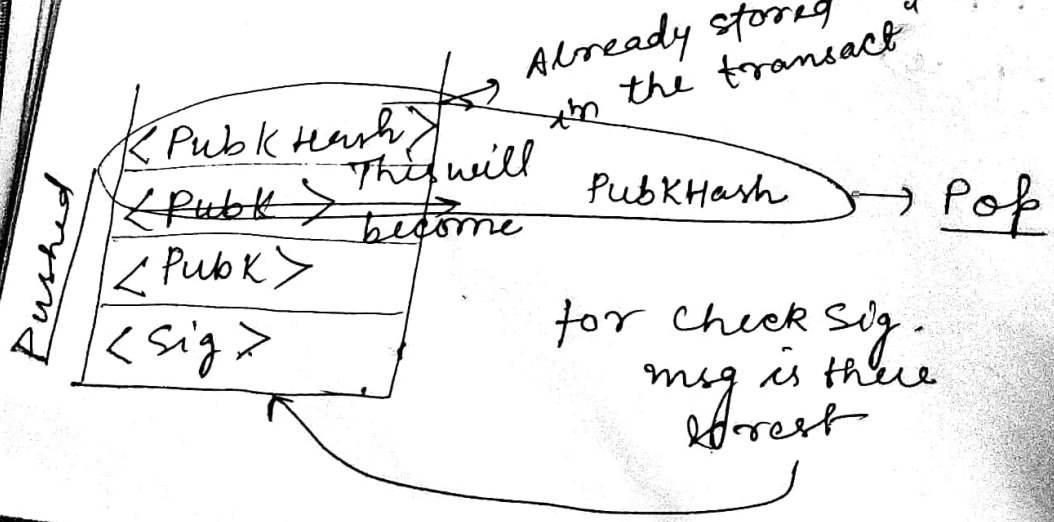
Verifier Part need only → PubKey + Msg + signature.

Unlocking Script

<Cafe signature> <Cafe Publickey>

Evaluating Combined Script

<sig> <PubK> DUP HASH160 <PubKHash>
EQUALVERIFY CHECKSIG.



Double Spending

on digital transaction I can send one 100 among 2 people.

So, how can we avoid that?

solution:

Every miner will maintain a list of unspent transaction o/p.

UTXO once you make the transaction automatically it is deleted from it. So that's how we take care of double spending.

Every miner has a responsibility of making new coins.

~~Reward~~ tran '0' → reward for miner. (which gives) — created after each tran.

1 satoshi = 10^{-8} BC.

0.01000 BC → $0.01 \times 10^8 = 10^6$ satoshis

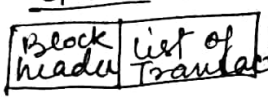
If the flow is more, the reward will come down.

Consensus

(Kind of Agreement)

31/Jan/2020

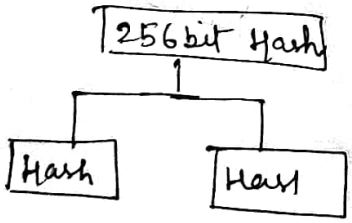
Block



connected via ↓ Backed linked list
hash value — SHA256

Hash is | 2cf - - - - - - 9824 |
↙
digest

Merkle root: To aggregate the transactions
→ It comprises of hash values.



Transactions can be of any size (may be with GBs) but finally they are converted into 256 bit.

Address of the block: Hash of the header block.

→ Our tr
header

Nonce:
by b

Block is
block
to mak

What is

H (pr

→ Supose w
make
with l
So this

→ Difficult

Mining

- Tampe

→ Our transactions are also part of the header

Nonce: Miner will calculate the nonce by bruteforce method.

Block is invalid because appending the block you need to change the nonce also to make it valid.

What is there in prev of Genesis block? → NULL

H (prev-Hash || Tx || (nonce)) < difficulty

Tx: "list of transc"

difficulty: this is also hash value.

→ Suppose we have to make a hash value with leading four bit as 0. So this is the difficulty.

→ Difficulty ↑ses as blocks ↑ses.

Mining process → is finding the nonce & we have to spend too much computational power

Tampering last block is a bit easy task. But not in the middle

As the blocks ↑ces, security also ↑ces

Everyone can become a miner, it was ~~those~~
but nowdays we cannot becomes because ~~not~~ thy are not only using GPUs but ASICs which has very highs computational

---

SHA-3 → 512 bits

∞ ip domain

2^{512} output domain

output changes about 50%.

Dvi Das Sir

7/2/08

Bitco

1) Sm
2) De
3) Tr

<script

↳ gt

#(why

Lock

(Locking

Vital

201

He

7/2/08

Bitcoin features :-

1) Immutability
2) Decentralization
3) Tracability

<script> is involved in writting Bitcoin programming.

↳ It can hold only 256 opcode.

OP_ADD

OP_SUB

OP_EQUAL-VERRIFY

<script>

Why Bitcoin is made turing incomplete?

Lock → send → unlock
(Locking script) (Unlocking script)

Vitalik

2014

He released a white ppr in the name Etherium

We can't write locking & unlocking script in other application.

Bitcoin	Ethereum
Nakamoto	Vitalik
2007	2014
Script-256	Solidity

Solidity
↳ This is used to write smart contracts
↳ set of instructions
↳ special kind of programs.
↳ we are not gonna explicitly run this program ~~only~~ its triggered.

You can ~~de~~implement our own idea in the form of smart contract and can deploy in Ethereum. which can't be done in Bitcoin.

Smart Contraction has :-
↳ Functions
↳ State Variable.

x = 5
x = 10
] Ethereum maintains both the states using time stamp concept.

function) which are associated with objects are methods.

⇒ We can assume our Ethereum BlockChain as Server &

MetaMask → To create a link b/n our Ethereum & Browser.

3 Types of n/w

Mainnet Test Private

Block Height → no. of blocks.

Smart Contract &

Pragma Solidity >0.4.0; → which compile we are going to use to comp Smart Contract

Contract SampleData

{

uint storeData

function set(uint x)

{

storeData = x

}

function get()

public View Returns (uint)

{

return storeData

}

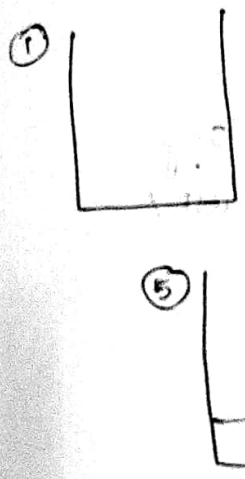
}.

To compile Ethereum we need EVM.

14/2/20

OP_2 OP_3 OP_

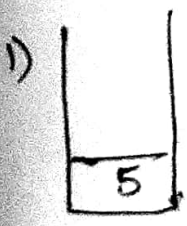
stack pointer



Result : false

Ex-2

OP_5 OP_



(3) OP_EQUA

popped

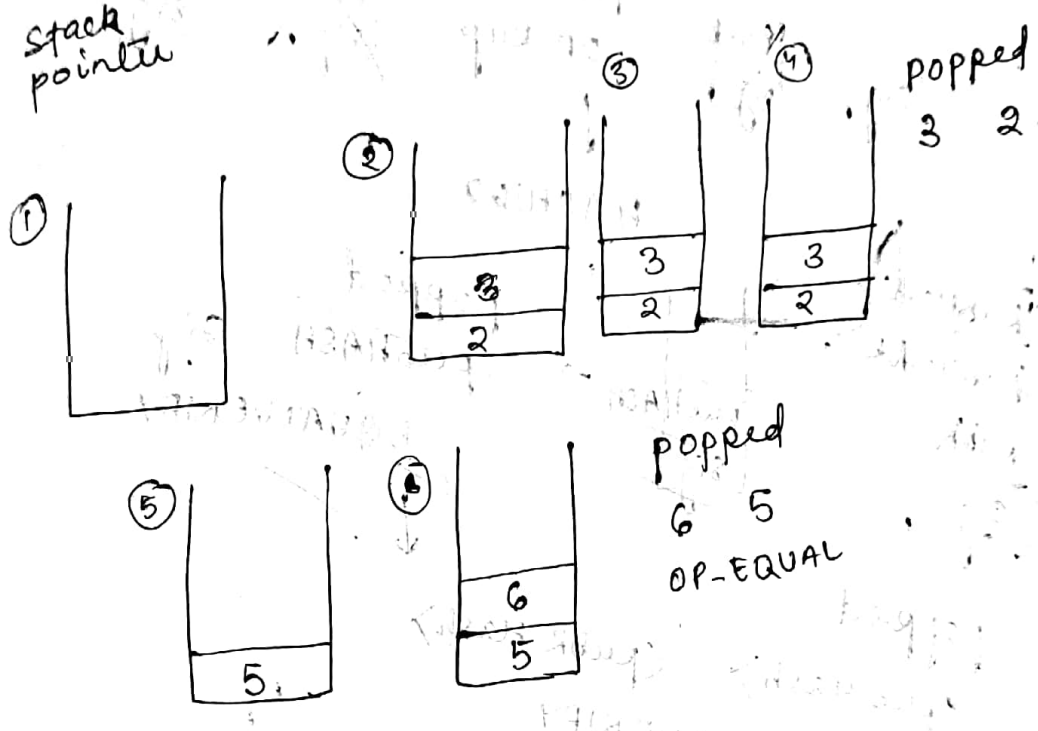
5

Result :

14/2/20

OP_2 OP_3 OP_ADD OP_6 OP_EQUAL

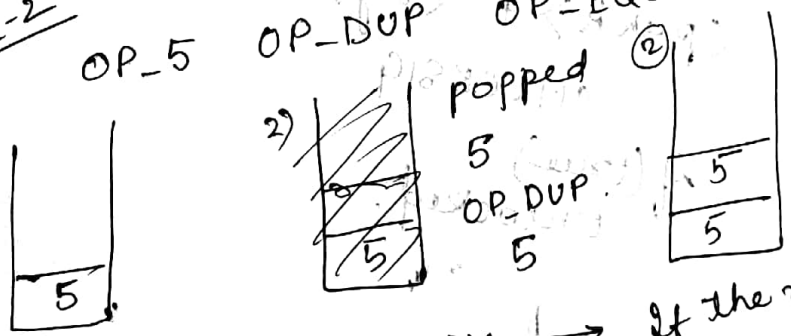
stack pointer



Result : False.

Ex-2

OP_5 OP_DUP OP_EQUALVERIFY



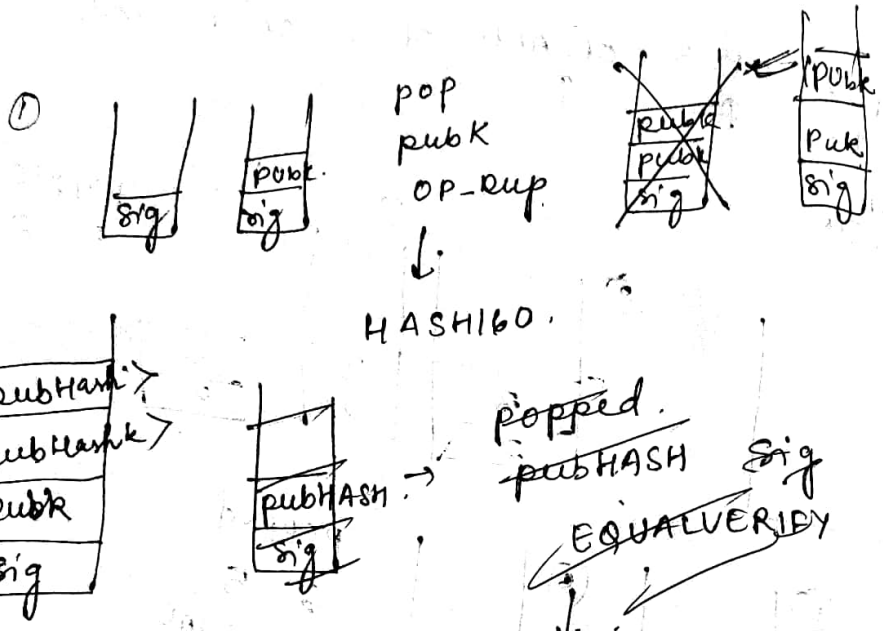
(3) OP_EQUALVERIFY → If the result is false then it stops or else it will go on.

popped
5 5

Result : True.

These are just values

<sig> <pubK> DUP HASH160 <pubKHash> EQUALVERIFY CHECKSIG



popped.

<pub Hash> <pubk Hash>

EQUALVERIFY

if(true)

popped.

<pukk> + <sig>

CHECKSIG

if(true)

unlocked.

solidity read

Gas Limit : 100.185

Gas Used by Tra

Gas Price = 0.

Assignmnt

Student Att

Contract

→ Details of st

→ Attendence

TP = GUT X

QP for 1 uni

solidity read the docs.

Gas Limit : 100.185
Gas Used By Txn : 100.185
Gas Price : 0.050 00...1 Eth

Assignment

Student Attendence Management Contract

→ Details of students
→ Attendence values

TP = QUT X QP

QP for 1 unit.