Accumulator

Given set S, prove that a particular element $e \in S$.

⇒ Primitives for membership or non-membership

$G = \{a : x^2 \equiv a \bmod p\}$ → cyclic group.

$|G| = \frac{p-1}{2}$

choose primes p, p', q, q' such that

$$p = 2p'+1, \quad q = 2q'+1, \quad N = pq$$

$G = \{a : x^2 \equiv a \bmod N\}$, G is a cyclic group

Hard problem:

$$x^2 \equiv a \bmod N$$

It is hard to check if a is QR mod N or not without knowing the factors of N

⇒ It is hard to find x without knowing the factors of N.

$G = \{a : x^2 \equiv a \bmod N\}$

⇒ cyclic group

⇒ $|G| = \left(\frac{p-1}{2}\right)\left(\frac{q-1}{2}\right)$.

N is public, p, q are private

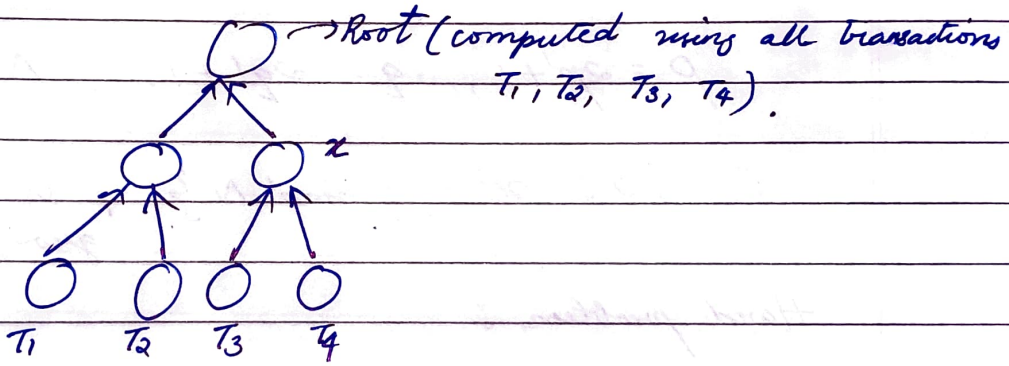
so G is a cyclic group of unknown order

Steps :

1. Find accumulated value
2. Witness
3. Verify

Merkle root can also be used to test membership
↓
used in blockchain

Root value is the accumulated value (combined value of all elements)



For T_2,

Witnesses are T_1, x (used to verify membership of T_2).

To be provided to verifier ⇒ Use T_1, x, Z (accumulated value = root here) to verify

Z = root is public. So verifier computes the root using (T_1, T_2) and x ~~then~~ ~~th~~ and compares it with Z.

Complexity = O(log n)
↳ height of tree

Can also be done in constant time using some other algo

Date
Page

Constant time Algo to verify membership

Public parameter : Cyclic group G with generator g

Let $y_1, y_2, \ldots, y_n \in G$.

Hash : $H : y_i \longrightarrow p_i$ (maps every element y_i to a prime number p_i).

Accumulated value : $A = g^{p_1 \times p_2 \times \ldots \times p_n} = g^{p_1 p_2 \ldots p_n}$

Witness (y_i) :

Compute $p_i = H(y_i)$.

$$w_i = (A)^{1/p_i} = g^{p_1 p_2 \ldots p_{i-1} p_{i+1} \ldots p_n}$$

verify (w_i, p_i, A) :

if $(w_i)^{p_i} == A$

accept

else

reject

$\Rightarrow O(1)$ complexity

Algo to verify non-membership

Let $S^* = p_1 p_2 \ldots p_n$

Given element y_j

Compute $p_j = H(y_j)$.

If $y_j \notin S$, p_j is not a factor of S^*

$\Rightarrow$ $GCD(s^*, p_j) = 1$. (p_j is prime & not a factor of s^*)

$\Rightarrow$ $\exists$ a, b such that

$as^* + bp_j = 1$.

$(g)^{as^* + bp_j} = g^1$

$$\boxed{(g^{s^*})^a (g^b)^{p_j} = g.}$$

(can find a, b using Extended Euclidean, g, s^*, p_j are known)

Non-membership (y_j):

Compute $p_j = H(y_j)$

Compute a & b

If $(g^{s^*})^a (g^b)^{p_j} = g$

non-member

✓ Above algo is dynamic

If y_j leaves

$\Rightarrow$ $A_{new} = (A)^{1/p_j}$

Remove p_j from witness also

$\Rightarrow$ $w_i = (w_i)^{1/p_j}$

But in Merkle root, the entire tree has to be reconstructed

If number of elements $\neq$ ~~next power of~~ multiple of 2

$\Rightarrow$ we also need dummy leaves

Date
Page

Commitment :

It is a cryptographic primitive that allows one to commit to a chosen value while keeping it hidden to others with the ability to ~~release~~ reveal the committed value later on.

2 properties

Hide : Value is hidden (no one can see the value until I reveal it myself) → from others (adversary)

Adv → Receiver

Binding : I (adversary) should not be able to change the value I comitted to

Adv → sender himself

⇒ used in blockchain also
(can commit my solution to the challenge, put it out on the blockchain and reveal it later).

13/10/22

commit(m) → C (Hide).
decommit(C) → m

⇓

no decryption involved. The owner just reveals the value he committed to
(used to prevent front attacks in blockchain).

Ways to implement committment :

1. using hash fn :

commit(m) → H(m).

Owner should not be able to find m' such that H(m) = H(m')
Otherwise ⇒ He can modify the value before revealing and hence binding property will not be satisfied.

But this is a hard problem with hash functions (due to secondary image resistance property)
⇒ so it can be used for committment (binding also guaranteed)

Prob - computationally intensive

2. Pederson Committment :

like DLP

PB : Schnorr cyclic group with generator g

DLP { value (number) h such that no one knows t which satisfies $h = g^t$.

⇓

DLP required for security of this scheme

If sender uses a large t, and computes h as g^t ⇒ it is still not very secure so h must be randomly chosen

Commit(m) :

$C = g^m h^r$ $\quad$ r - random number

By DLP, it is not possible to compute m

Decommit(C) :

if ($g^m h^r == C$) $\quad$ (sender reveals m, r).
accept
(otherwise sender has modified m).

~~(Say sender can compute m', r' such that $C = g^{m'} h^{r'}$, then we show that one can also break some hard problem).~~

If sender can compute (m', r') such that $C = g^{m'} h^{r'}$ then one can solve the DLP problem

$$C = g^m h^r = g^{m'} h^{r'}$$

$$g^{m-m'} = h^{r'-r}$$

$$h = g^{\frac{m-m'}{r-r'}}$$

~~Given h, g, m, m', r, r' and $h = g^t$,~~
we can compute t as

$$t = \frac{m-m'}{r-r'}$$ $\quad$ (m', r' are revealed by sender. How can we know m, r??).

$\Rightarrow$ can solve DLP ??

So sender cannot modify m

*. Pederson scheme is homomorphic

$$C(m_1+m_2) = g^{m_1+m_2} h^{r}$$
$$C(m_1)\,C(m_2) = g^{m_1} h^{r_1} g^{m_2} h^{r_2}$$
$$= g^{m_1+m_2} \cdot h^{r_1+r_2}$$

If we choose r as $r_1 + r_2$, it then $C(m_1+m_2) = C(m_1) \cdot C(m_2)$
⇒ homomorphic ??

Pederson Committment with additive group

PB : i) elliptic cyclic group with generator G
ii) H such that no one knows t which satisfies $H = tG$.

Commit (m) :

$C = mG + rH$ r - random

Decommit (m) :

if ($mG + Hr == C$)
accept.

homomorphism :

$$C(m_1+m_2) = (m_1+m_2)G + rH$$
$$C(m_1) + C(m_2) = m_1G + r_1H + m_2G + r_2H$$
$$= (m_1+m_2)G + (r_1+r_2)H$$

Fixing r as (r_1+r_2),
we can find committment for m_1+m_2
⇒ homomorphic

Shamir's secret sharing

Say there are 11 deans. A room can be opened only if 6 or more deans come together (a dean can have 2 keys also)

1. How many locks are needed
2. How many keys should each dean have

We can have a lock for every 5 deans but give the keys only to the remaining 6 deans
⇒ no 5 deans will be able to open all locks
(atleast one will be lock cannot be unlocked).

∴ Number of locks = $^{11}C_5$. (all possible 5-dean groups)

Number of keys per dean = $^{10}C_5$
(key for every group a dean is not a part of).

Given a secret s.
I want to compute n shares of the secret s such that atleast t shareholders must come together to

Given a secret s.
Dealer computes n shares from s such that if one gets t or more shares, he can compute secret s.

Idea behind dears & locks puzzle

⇒ If ~~you select~~ 5 or fewer dears try to unlock of the room
then there is 1 groups involving all 5 of them and nobody would have the key ~~corresponding~~ for the lock corresponding to their group
If there is 1 more person, then she can open that ~~lock~~ only left-out lock

more groups if < 5

14/10/22

Dealer computes n-shares of secret s and give it to n-persons
If t-persons come together, they can compute secret S.

⇒
Dealer creates a polynomial of degree $t-1$
All coefficients are random but constant term is secret S

$$p(x) = \underbrace{S}_{\text{secret}} + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1}$$

$a_1, a_2, \ldots, a_{t-1} \rightarrow$ random nos.

Random

Choose x_1 and compute $p(x_1)$
Choose x_2 & compute $p(x_2)$
⋮

$p(x_1) \rightarrow$ shares of 1st person
$p(x_2) \rightarrow$ shares of 2nd person
⋮
$p(x_n) \rightarrow$ shares of nth person

Now there are t variables in $P(x)$

$\downarrow$

$s, a_1, a_2, \ldots, a_{t-1}$

so if t people come together, we have t equations and t unknowns

$\rightarrow$ can find all coefficients and secret s.

(So ~~each~~ person i knows both x_i and $P(x_i)$?).

by solving t eqns. ↑

t people $\Rightarrow$ can compute entire poly

But they only need S.

$\Rightarrow$ can be done using lagrange algo

Lagrange Interpolation

$$P(x) = \underbrace{b_1 \frac{(x-x_2)(x-x_3)\ldots(x-x_t)}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_t)}}_{\text{poly of degree } t-1} + b_2 \frac{(x-x_1)(x-x_3)\ldots}{\underbrace{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_t)}_{\text{constant}}}$$

$$+ \frac{b_2(x-x_1)(x-x_3)\ldots(x-x_t)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_t)} + \ldots +$$

$$b_t \frac{(x-x_1)(x-x_2)\ldots(x-x_{t-1})}{(x_t-x_1)(x_t-x_2)\ldots(x_t-x_{t-1})}.$$

S = constant term in $p(x)$.

$$= b_1 \cdot \frac{x_2 x_3 \ldots x_t}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_t)} + \ldots + b_t \frac{x_1 x_2 \ldots x_{t-1}}{(x_t-x_1)\ldots(x_t-x_{t-1})}$$

$P(x_1) = b_1$

$P(x_2) = b_2$

$\vdots$

~~$P(x_n) = b_n$~~

$P(x_t) = b_t$

$$P(x) = P(x_1)[\ ] + P(x_2)[\ ] + \dots + P(x_t)[\ ].$$

constant term = $P(0)$.

$S = P(0)$.

⇒ We did not have to solve t eqns - much easier to do this

1. $S = 9,\ p = 41,\ t = 3,\ n = 7$

n can be anything in general

↓ #shares

$p(x) = S + a_1 x + a_2 x^2$ (degree $t-1 = 2$)

↓ ↓ random

Let $p(x) = 9 + 2x + 31x^2$

~~Say $x_1 = 1,\ x_2 = 2,\ x_3 = 3, \dots$~~

Assume that $x_i = i : 1 \le i \le 7$

$p(x_1) = (9 + 2 + 31) \bmod 41$

$= 42 \bmod 41 = 1.$

41
7
123

$p(x_2) = $ ~~$(9 + 18 + 124) \bmod 41$~~ $(9 + 4 + 124) \bmod 41$

$= $ ~~$151 \bmod 41$~~ $137 \bmod 41$

$= 14$

41
6
246

$p(x_3) = (9 + 6 + 279) \bmod 41$

$= (15 + 33) \bmod 41$

$= 48 \bmod 41 = 7.$

$(1, P_1) = (1, 1)$

$(2, P_2) = (2, 14)$

$(3, P_3) = (3, 7)$

$(4, P_4) = (4, 21)$

$(5, P_5) = (5, 15)$

$(6, P_6) = (6, 30)$

$(7, P_7) = (7, 25)$

Say we choose 3 shares → 1, 6, 7.

~~$p(x) = b_1 (x - x_1)(x - x_6)$~~

$$p(x) = b_1 \frac{(x - x_6)(x - x_7)}{(x_1 - x_6)(x_1 - x_7)} + b_2 \frac{(x - x_1)(x - x_7)}{(x_6 - x_1)(x_6 - x_7)} + b_3 \frac{(x - x_1)(x - x_6)}{(x_7 - x_1)(x_7 - x_6)}$$

$$= p(x_1) \frac{(x-6)(x-7)}{(-5)(-6)} + p(x_6) \frac{(x-1)(x-7)}{5 \cdot -1} + p(x_7) \frac{(x-1)(x-6)}{6 \cdot 1}$$

$$s = p(0) = \left(1 \cdot \frac{-6 \cdot -7}{30 \cdot 5} + 30 \cdot \frac{-1 \cdot -7}{-5} + 25 \cdot \frac{-1 \cdot -6}{6} \right) \bmod 41$$

$$= \left(\frac{7}{5} - 42 + 25 \right) \bmod 41$$

5

$\left(\frac{9}{5} - 17\right) \bmod 41$

$= (7.5^{-1} \bmod 41 + 24) \bmod 41$

41		1	0
5	8	0	1
1		1	-8
0			

$-8 \bmod 41 = 33.$

$= (7.33 + 24) \bmod 41$

$= (231 + 24) \bmod 41$

$= 255 \bmod 41 \quad (255 - 246).$

$= 9$

Theorem:

Shamir Secret sharing is perfectly secure

⇒ With $t-1$ shares, you cannot get any information about s.

(eg: predicting i^{th} bit of s as 0 can be done only with prob = 1/2).

Proof:

To prove: $Pr[S = s \mid P_{i \in C}^{\text{shares}}, C : t-1 \text{ shares}] = Pr[S = s].$

$$Pr[S = s \mid P_{i \in C}^{\text{shares}}] = \frac{Pr[(S = s) \wedge (P_{i \in C}^{\text{shares}})]}{Pr[P_{i \in C}^{\text{shares}}]}$$

we don't know the secret. to consider every possible value of secret to find $P(\text{shares})$

$$= \Pr[(S=s) \cap (\text{shares} = P_{i \in C})]$$

$$= \Pr[(\text{shares} = P_{i \in C}) \mid S = s] \cdot P(S = s)$$

$$\sum_{s_j \in Z_p} \Pr[(\text{shares} = P_{i \in C}) \mid S = s_j] \cdot P(S = s_j)$$

$Z_p = \{0, 1, \ldots, p-1\}$

Assuming all shares are drawn from a uniform distribution

$$\Rightarrow \Pr[\underbrace{1^{st} \text{ share}}_{\text{does not depend on secret } s.} = 5] = 1/p.$$

$$\Rightarrow \Pr[\underbrace{(\text{shares} = P_{i \in C})}_{t-1 \text{ shares}} \mid \underbrace{S = s}_{\text{does not influence shares}}] = \frac{1}{p^{t-1}}$$

same as $\Pr[\text{shares} = P_{i \in C}]$.

$$= \frac{\frac{1}{p^{t-1}} \cdot P(S = s)}{\sum_{s_j \in Z_p} \frac{1}{p^{t-1}} \cdot P(S = s_j)}$$

$$= \frac{\frac{1}{p^{t-1}} \cdot P(S = s)}{\frac{1}{p^{t-1}} \underbrace{\sum_{s_j \in Z_p} P(S = s_j)}_{= 1 \text{ (summing over all possible values of } S).}}$$

$$\Pr[S = s \mid (\text{shares} = P_{i \in C})] = \Pr[S = s].$$