10/22

Shamir Secret Sharing

$$f(x) = S_1 + a_1 x + \ldots + a_{t-1} x^{t-1}$$

(t-degree polynomial)

secret $S_1 = f(0)$

Let $g(x) = S_2 + b_1 x + \ldots + b_{t-1} x^{t-1}$

For secret $S = S_1 + S_2$, the shares are given by

$$h(x) = f(x) + g(x)$$
$$= (S_1 + S_2) + \boxed{c_1} x + \ldots + \boxed{c_{t-1}} x^{t-1}$$

Homomorphism (addition)

If we know the shares of S_1, S_2 then
shares of S = shares of S_1 +
shares of S_2

Adding the shares $\Longleftrightarrow$ secrets are added

Homomorphism (multiplicative)

If all shares of S are multiplied by n, then secret is S is also multiplied by n

$$f(x_1) \longrightarrow 5f(x_1)$$
$$\Rightarrow 5f(x_1) = 5(S_1 + a_1 x_1 + \ldots + a_{t-1} x_1^{t-1})$$

entire poly. is multiplied by 5

∴ Shamir secret sharing idea is homomorphi

Class P :

~~(eg)~~ decision problems → answer Yes or No
decision problems which can be solved in poly. time by a deterministic algorithm are class P problems.

Instance of a problem :

Converting a problem to a string in a language given by $\{0,1\}^*$.

Class P is a language consisting of all problems that can be solved by a deterministic Turing Machine

Class NP :

↓

Non-deterministic polynomial time

decision problems that can be solved ~~in polynomial time~~ by a non-deterministic polynomial time algorithm

NP Algorithm's steps

Assumption ① NP algorithm has a magical power to guess the solution.
→ they get the certificate

(a) The solution can be verified in poly. time

NP problems are decision problems that can be verified in polynomial time by a deterministic algorithm
(step 1 gives a solution which has to be verified in polynomial time).
(assume soln is given to us and verify it in poly time).

NP → language which can be accepted by a non-deterministic Turing Machine

Interactive Proofs (IP)

Proof : A correct and valid argument which establishes the truth of a statement. ↓
consists of premises (assumptions) followed by 1 or more conclusions

IP → involves 2 people
→ prover
→ verifier
tocol is
rover tries to prove a statement to the verifier

ractive Proof Properties :

Completeness
oundness

Completeness :

Consider both P & V are honest and that P knows that the statement is true

⇒ P should be able to prove the statement to the verifier with overwhelming probability (> 1/2 atleast 2/3 but need not be 1)

Soundness :

Consider that P is not honest but verifier is honest ⇒ P does not know if stmt is true or not.

⇒ Dishonest prover P' cannot convince the verifier regarding the truth of a statement (P cannot prove a stmt).

(OR)

A language L has IP system if some probabilistic polynomial time algorithm V (randomized) exists such that for some algorithm P and for every P' (⇒ dishon P) and for ~~every~~ a string x, following properties hold

i) ~~Com~~

i) Completeness :

If $x \in L$ then $(P, V)(x) = 1$.

(P should be able to prove x to V with overwhelming prob.)

ii) Soundness :

If $x \notin L$ then $(P, V)(x) = 0$

$IP = \{ L$ which has interactive proof $\}$.

$NP = \{ x : V(x) = 1 \}$.

⇓

set of strings that can be verified in polynomial time

$NP \subseteq IP$ (can extended NPs capabilities and make it interactive using an alg algo P).

27/10/22

Zero Knowledge Proof (ZKP)

(1) Prover is trying to convince ~~to~~ the verifier that a statement is true without revealing anything except that the statement is true.
→ a secret is need to achieve this

(2) Prover knows some secret and is trying to convince the verifier that he knows the secret without revealing the secret

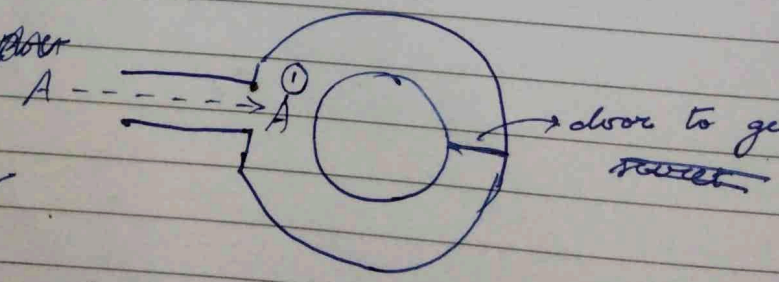
This is also a form of IP. So it has the following properties:

1. Soundness
2. Completeness
3. ZK property

⇓

Prover does not reveal anything about the secret

Alibaba & Cave



Prover : Alice
Verifier : Bob
Alice knows the secret to open the door

I : Bob tells to go
Alice gets into the cave.
II : Now Bob gives Alice a challenge to ~~be noticed~~ to solve

⇒ the challenge here is random
We assume that the verifier Bob, is honest (simply tosses a coin and tells her which direction - left/right to go based on the toss's ~~coin~~ outcome).

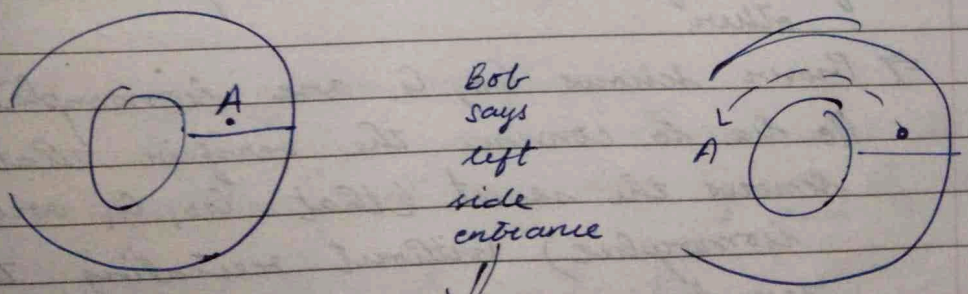
Assume Bob got a head and directs Alice to go left.
⇒ he can observe Alice going in that direction.

If Alice really knows the secret, she will open the door and come out through the other way
If she comes out in the same

① Bob tells Alice to go in but he does not know which direction she takes
Alice chooses 1 direction & goes to the door

② After Alice gets to the door, Bob goes in front of the cave and randomly asks Alice to come out through one of the ways

Alice can get out with $P_E = 1/2$ even if she does not know the secret



$p = 1/2$ (random choice)

Alice can get out via the same route she went without passing through the door

So, Bob and Alice repeat this experiment t times

$\Rightarrow$ ~~P(convince~~

P(prover convincing verifier without knowing the secret) $= \left(\frac{1}{2}\right)^t$

$\Rightarrow$ for this probability to be negligible $t \geq 100$ (roughly)

But if Alice knows the challenge from Bob (the direction he asks her to come out from) then Alice can always convince the verifier just by choosing the same direction Bob would choose without even knowing the secret.

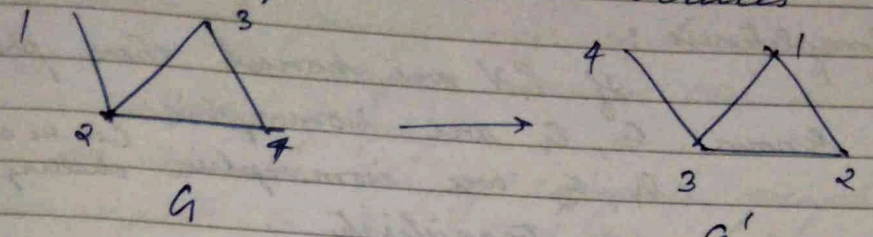
(1) Isomorphism :

If A mapping $I : G_0 \rightarrow G_1$ which is
1. one-to-one mapping between V_0 to V_1
2. one-to-one mapping between E_0 to E_1
such that incidence relationship is preserved then G_0, G_1 are isomorphic

Given 2 graphs G_0, G_1, it is hard to find if G_0, G_1 are isomorphic to each other.

$\Rightarrow$ Prover knows G_0, G_1 are isomorphic He has to convince the verifier that he knows the secret (that G_0, G_1 are isomorphic) without revealing the secret.

Given graph G_1, finding a graph G' that is isomorphic to G_1 is easy
→ just permute the vertices



G → G'

⇓

easy to find

checking if G_1, G' are isomorphic (in general for any 2 graphs) is difficult).

Protocol: Say G_0, G_1 are isomorphic & prover wants to prove he know it

1. Compute G_2 which is isomorphic to G_1 using $f: G_1 \to G_2$

2. Prover sends G_2 to verifier (Only G_2 shared and not f).
(If prover is malicious, he can send something else instead of G_2 also)

⇓

Verifier cannot check if G_1, G_2 are isomorphic ⇒ hard problem
(so he relies on the prover here which can be exploited)

3. Challenge from verifier:
Prover has to show isomorphism between G_i and G_2 where

$i \in 2$ $i \in 2$ ($\Rightarrow i \in \{0,1\}$)
i is chosen randomly

③ Prover has to prove isomorphism btw G_0, G_2 or and G_1, G_2 depending on the value of i

✓ Completeness :

If P, V are honest then prover knows G_0, G_1 are isomorphic
G_1, G_2 are isomorphic (so he can solve the challenge if i=1)
By transitivity
G_0, G_2 are also isomorphic

Honest Prover should be able to convince the verifier

prover can't solve the challenge if i=0

⇒ He should be able to solve the challenge

Isomorphism problem :

Given graphs G_0, G_1
Secret : G_0, G_1 are isomorphic
Prover has to prove he knows the secret using zero knowledge proof

28/10/22

Prover can convince the verifier ⇒ in polynomial time

✓ Soundness :

Prover P is dishonest.

P knows $\exists I : G_0$ $I : G_1 \to G_2$ but not $I : G_0 \to G_1$

If challenger verifier picks i=1
⇒ prover can show $I : G_1 \to G_2$

but if i=0
prover has to prove $I : G_0 \to G_2$ but he does not know $I : G_0 \to G_1$ itself

⇏

each step in the

Suppose prover knows that verifier is going to ~~ask~~ give for $i=0$

⇒ dishonest prover can send any graph as G_2 (and the verifier assumes this to be the actual G_2 graph).

so he can simply compute graph G_2 isomorphic to G_0 (not G_1) and send it to the verifier so that he can prove the ~~challenge~~ $I: G_0 \to G_2$ (which is the challenge given by the verifier).

P(dishonest prover convinces verifier) $= 1/2$

⇒ Repeat this process t times

(if ~~very~~ prover cannot prove the 1st time itself, we stop

Repeat only if prover solves the challenge)

⇒ P(dishonest P convinces V) $= \left(\frac{1}{2}\right)^t$

$t = 100$

⇒ Prob. is negligible

✓ Zero knowledge property: Prover does not reveal anything about the secret.

Real protocol

$P \xrightarrow{x_1} V$

$V \xrightarrow{x_2} P$

$P \xrightarrow{x_3} V$

each step in the process ↑ transcript

simulation (without an actual prover)

$\xrightarrow{x_1} V$

$V \xrightarrow{x_2}$

$\xrightarrow{x_3} V$

→ in poly time

If such a simulation is possible then no secret is revealed (which is why we were able to simulate the protocol without a real prover).

Simulator has to create real transcript in polynomial time (expected) without participation of prover

Steps :

Simulator chooses i randomly, $i \in \{0, 1\}$.

If $i = 0$

find $I : G_0 \to G_2$ isomorphic graph G_2 from G_0

else

find isomorphic graph G_2 from G_1

(1) send G_2 to V. $(P \xrightarrow{G_2} V)$

(2) Verifier chooses $i' \in \{0, 1\}$ randomly

Verifier sends i' back $(V \xrightarrow{i'} P)$.

↓ no real prover here

if $i == i'$

⇒ simulator can solve the challenge

⇒ real protocol is simulated

else

go to step 1.

~~Or~~ Within an expectation value of 2 trials, simulator can ~~or~~ create the real transcripts

bcos $P(\text{success}) = 1/2$

Time complexity of simulator = 2^t. (t is #transcripts)

(2) Hamiltonian Cycle Problem :

Given graph G, finding a hamiltonian ~~of~~ cycle is a hard problem

(complete graph - easy
any general graphy - hard).

Given graph G
secret : ~~G~~ hamiltonian cycle in G.

11/11/22

Camlin Page
Date / /

Prover has to prove his knowledge of x such that $y = g^x$ (g is generator of Schnorr group)
(y is public)

Honest Verifier (assumption).

① Prover ~~comp~~ chooses r and computes $t = g^r \bmod p$

② ~~The~~ $V \xrightarrow{c} P$

③ $P \xrightarrow{s = r + cx} V$

Verifier checks if $g^s = ty^c$

V has to choose c (of some 100 bit size) randomly (if V is honest).

Dishonest Verifier

⇒ V can send c by computing it using some algorithm and not choose it randomly

↓

a function $f(t, g, \ldots)$

i.e $c = f(t, g, \ldots)$.

~~The~~ simulator first chooses c, then s ~~x~~. and finally t.

(has to a simulate a virtual transcript without any real prover ~~and~~ such that the probability of each virtual transcript is same as that of the real transcript)

⇒ Cannot be done ∴ when V is dishonest

Zero knowledge property may not be satisfied with dishonest verifier

because $c = f(b, g, \ldots)$ but t is computed using c by simulator
⇒ Dishonest V protocol may not be secure
(Maybe secure but proof not known)

We can force the verifier to choose a c randomly to prevent any dishonesty in verifier
⇒ Verifier is made to commit to a value c even before the above protocol begins

1. Verifier commits c
2. Prover chooses x and sends $t = g^x \bmod p$
3. Verifier reveals the value he committed
4. $P \xrightarrow[s = x + cx]{} V$

Interactive proofs → ① requires all the parties involved (P and V) to be online always
② Also there maybe many V and prover has to prove to each V individually

Non-interactive ZKP :

⇒ Verifier can verify after a day later also

Prover chooses x, computes t
⇒ V not needed here
But to compute s, he needs c from verifier
(⇒ the interactive step)

Instead, prover himself has to choose c for non-interactive proof

But prover can be dishonest here (can convince V by choosing a value of c even without knowing x)

⇒ We use hash (↓ random o/p for a given i/p).

~~Even if prover does not choose c randomly, H(~~

Prover chooses c = H(t, g, y). (t should be known to both P and V)

- If prover predicts c, he can ~~choose~~ convince the verifier (public params).

(1) Even if he randomly chooses c he cannot compute s = r + cx such that $g^s = r\, t\, y^c$ because of DLP

~~(x cannot be known because if you choose c, you cannot find t~~

even if he chooses t randomly

(2) Even if prover randomly chooses s and c, and computes t such that $g^s = t\, y^c$ but verifier will check that c = H(t, g, y) which will not be satisfied

⇒ Dishonest prover cannot cheat.

Steps

1. Prover chooses r, computes $t = g^r \bmod p$
2. Prover computes $c = H(t, g, y)$
3. Prover computes $s = r + cx$
   P sends r, t, c, s to V
4. Verifier checks if $c == H(t, g, y)$. and $t = g^s y^{-c}$.

can be done any time

(*) IKP was introduced in 1985 by Shafi Goldwasser, Silvio Micali and Charles Rackoff (Interactive).

Fiat & Shamire came up with Non-interactive proof.

(1) If P knows two secrets x_1, x_2 such that $y_1 = g^{x_1}$ and $y_2 = g^{x_2}$ and has to prove both to verifier

→ Repeat the entire protocol again for x_2 after proving knowledge of x_1
→ Instead of repeating all steps, we can use the same random c sent by V for both secrets.

14/11/2

(1) P computes $t_1 = g^{r_1} \bmod p$, $t_2 = g^{r_2} \bmod p$

$P \xrightarrow{t_1, t_2} V$

(2) $P \xleftarrow{c} V$

Camlin Page
Date / /

(3) P $\xrightarrow{s_1 = r_1 + cx_1,\ s_2 = r_2 + cx_2}$ V

checks if $t_1 = g^{s_1} y_1^{-c}$ and $t_2 = g^{s_2} y_2^{-c}$

(2) IIIly prover might know x such that $y_1 = g^x$ and $y_2 = h^x$

($g, h \rightarrow$ 2 generators of group G).
↓
can be more than 1

(Repeat above process with $t_2 = h^{r_2}$ mod p and V checks if

$t_1 = g^{s_1} y_1^{-c}$

$t_2 = h^{s_2} y_2^{-c}$

(3) Prover may want to prove that he knows x_1 or x_2

⇒ sufficient to prove that he knows only x_1 (or only x_2)

But this ~~might~~ reveals ~~some info about x_2~~ That he knows x_1 ~~only~~

14/11/22

* ZKP for knowing x_1 or x_2

Instead of proving his knowledge of only 1 values (which reveals info) he shows that he knows both

(Here P wants to convince the verifier that that he knows atleast one of the secrets.

(1) Prover chooses r_1

Computes $u_1 = g^{r_1}$

chooses s_2, c_2 and computes $u_2 = g^{s_2} y_2^{-c_2}$

But c_2 now is not given by the verifier

1. $P \xrightarrow{u_1, u_2} V$
2. $P \xleftarrow{c} V$
3. P computes $c_1 = c \oplus c_2$

$$s_1 = r_1 - c_1 x_1$$
$$s_2 = r_2 - c_2 x_2$$

$$P \xrightarrow{s_1, s_2, c_1, c_2} V$$

4. ~~For~~ V checks

$$\text{if } (u_1 == g^{s_1} y_1^{-c_1} \text{ and } u_2 == g^{s_2} y_2^{-c_2}) \text{ accepts}$$

The challenge c sent by verifier is considered to be XOR of challenges for both ~~*~~ secrets

This method just reveals that he knows only 1 secret but not which one

(If he ~~does~~ knows only x_1 and not x_2 he can still convince the verifier knows x_2 and not $x_1 \Rightarrow$ just swap them).

*. Prover wants to convince the verifier that he knows x, r such that

$$C = g^x h^r$$

This is the public value that ~~has~~ is ~~been committed to~~ (Pedersen commitment).

1. Prover chooses x_1, r_1.
Computes $u_1 = g^{x_1} h^{r_1}$

$P \xrightarrow{u_1} V$

2. $V \xrightarrow{c_1} P$

3. Prover computes

$s_1 = x_1 + c_1 x$ ⇒ for proving knowledge of x

$s_2 = r_1 + c_1 r$ ⇒ for proving knowledge of r

$g^{s_1} = g^{x_1} (g^x)^{c_1}$

~~$g^{s_2} = g^{r_1} (g^r)^c$~~

$h^{s_2} = h^{r_1} (h^r)^{c_1}$

$g^{s_1} h^{s_2} = g^{x_1} h^{r_1} (g^x h^r)^{c_1}$

$= u_1 (\quad)^{c_1}$

$P \xrightarrow{s_1, s_2, c_1, c_2} V$

4. V will check

if $(g^{s_1} h^{s_2} == u_1 C^{c_1})$

accept

or

if $(u_1 == g^{s_1} h^{s_2} C^{-c_1})$

accept

*. ↑ with AND for 2 commitments

(P knows (x_1, r_1) of C_1 and (x_2, r_2) of C_2)

Repeat all the above steps for both C_1 & C_2 but use same challenge c)

18/11/22

*. Prover knows that secrets (x, r) in 2 commitments C_1, C_2 are same

$$C_1 = g_1^x h_1^r \qquad C_2 = g_2^x h_2^r$$

⇒ P uses same values and computes u_1, u_2 (bcos x, r are same for both C_1, C_2).

1. P chooses ~~values~~ r_1, r_2 and computes $u_1 = g_1^{r_1} h_1^{r_2}$

$$u_2 = g_2^{r_1} h_2^{r_2}$$

$$P \xrightarrow{u_1, u_2} V$$

2 $$V \xrightarrow{c} P$$

3. $$s_1 = r_1 + cx$$
$$s_2 = r_2 + cr$$

$$P \xrightarrow{s_1, s_2}$$

$$g_1^{s_1} = g_1^{r_1} (g_1^x)^c$$

~~$g^{s_2} = g^{r_2} (g^r)^c$~~

$$g_2^{s_1} = g_2^{r_1} (g_2^x)^c$$

$$h_1^{s_2} = h_1^{r_2} (h_1^r)^c$$
$$h_2^{s_2} = h_2^{r_2} (h_2^r)^c$$

$$g_1^{s_1} h_1^{s_2} = g_1^{r_1} h_1^{r_2} (g_1^x h_1^r)^c$$
$$= u_1 (C_1)^c$$
$$g_2^{s_1} h_2^{s_2} = g_2^{r_1} h_2^{r_2} (g_2^x h_2^r)^c$$
$$= u_2 (C_2)^c$$

4. V checks of

if ($u_1 = = g_1^{s_1} h_1^{s_2} c_1^{-c}$ and
$u_2 = = g_2^{s_1} h_2^{s_2} c_2^{-c}$)
accept

18/11/22

21/11/22

ZK - SNARKS

⇒ Non-algebraic functions involved

(Eg): Secret is Prover knows x such that

$H(x) = y$

can be any complex function

Step 1 : Computation of arithmatic circuit (+, −, ×) (division can be implemented using +, −, ×)

Step 2 : Convert arithmatic circuit to Quadratic Arithmatic Programming (QAP) → polynomial

(Proving $H(x) = y$ is the same as showing $p(x)$ is divisible by $T(x)$)

target poly corresponding to y

Given $T(x)$, it is hard to find $p(x)$ satisfying divisibility criteria (so x is not revealed).

QAP:

QAP over field F contains three sets of polynomials

~~$A = \{V_k(x)\}$~~ $1 \leq k$

L (left) ← $A = \{A_k(x)\}$ $0 \leq k \leq m$

R (right) ← $B = \{B_k(x)\}$ "

output ← $C = \{C_k(x)\}$ "

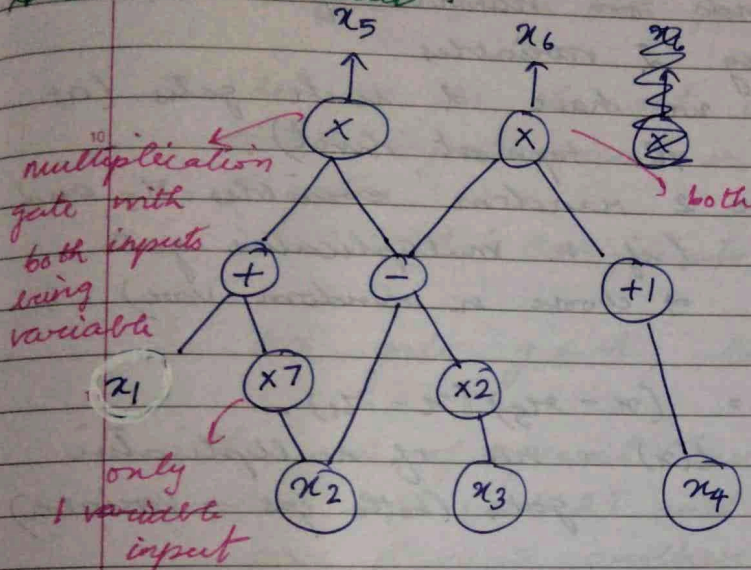
target polynomial is $t(x)$

(Eg): $C: F_{11}^2 \times F_{11}^2 \to F_{11}^2$

$C(\underbrace{x_1, x_2, x_3}_{ip}, \underbrace{x_4, x_5, x_6}_{op}) = \{(x_1 + 7x_2)(x_2 - 2x_3), (x_2 - 2x_3)(x_4 + 1)\}$.

?

Arithmatic circuit:



→ only 2 gates × gates with 2 variable ips

If $N = 6$, then assignment = $(\underbrace{x_1, x_2, x_3, x_4}_{\text{input}}, \underbrace{x_5, x_6}_{\text{output}})$

↓ secret

Algorithm

Fields assignment $(a_1, a_2, \ldots, a_{N-1}, a_N) \in F^N$ is valid assignment of f's input and output if and only if there exists $(a_{N+1}, a_{N+2}, \ldots, a_m)$ such that $t(x)$ divides $p(x)$

(Now secret is $p(x)$ and not the assignment → enough to show $t(x)$ divides $p(x)$).

$A(x)$

$$p(x) = \underbrace{\left(A_0(x) + \sum_{k=1}^{n} a_k A_k(x)\right)}_{A(x)} \left(B_0(x) + \sum_{k=1}^{m} a_k B_k(x)\right) - \left(C_0(x) + \sum_{k=1}^{m} a_k C_k(x)\right).$$

$p(x) = A(x) \cdot B(x) - C(x)$.

Target poly: $t(x)$ depends on number of multiplications involving 2 variables

for eg, we have 2 such gates (at the topmost level).

⇒ choose 2 random variables r_5 and r_6 (if n multiplication gates ⇒ choose n random var.)

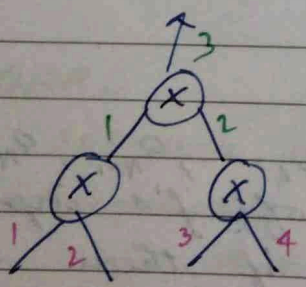
$$t(x) = (x - r_5)(x - r_6).$$

degree of $t(x)$ = no. of multiplication gates (both ips → variable)

m = Number of inputs to X gates + no. of outputs of X gates

Here $m = 4 + 2 = 6$

(2 ips for each gate)



⇒ $N = 4 + (1) = 5$ (only final output)

⇒ $m = 4 + 3 = 7$

(3 o/p from each gate)

In general $m \geq N$

↓ includes intermediate ops from X gates as well

↳ runs only final outputs

degree of A_k, B_k, C_k = no. of X gates $- 1$

Notations :

1. Let M be the set of multiplicative (X) gates (with both ips → variable)

2. Let W be the set of special wires i.e either input wires or output wires of X gates

(X : multiplicative gates).

3. For gate $g \in M$, let $I_{g,L} \subset W$ be the set of special wires entering g from the left (set A) and which does not pass through another X gate before g

(Similarly for $I_{g,R}$).

↓

$I_{g,R} \subset W$ → set of special wires entering g from the Right and does not pass through another X gate before g

Left input of X gate → corresponds to polynomials in set A

Right input of X gate → corresponds to poly in set B.

$\overrightarrow{A} = (A_0, A_1, A_2, \ldots, A_m)$.

(Each g gate in W gives a value for A_i → these values are given using which we construct A_i).

For our example, $A_i(r_5)$ and $A_i(r_6)$ are given

→ we construct A_i with degree $2-1=1$ (in general → no. of X gates − 1).

$$A_i(r_g) = \begin{cases} C_{g,L,i} & \text{if } i \in I_{g,L} \\ 0 & \text{otherwise} \end{cases}$$

($C_{g,L}$ is coefficient).

$x_1 + 7x_2$

$C_{g,L}$ is 1 $C_{g,L}$ is 7.

$C_{g,L,i}$ denotes the scalars with which i^{th} special wire ~~and~~ enters gate g from left.

Illy $B_i(r_g) = \begin{cases} C_{g,R,i} & \text{if } i \in I_{g,R} \\ 0 & \text{otherwise} \end{cases}$

Output polynomial : $P(\vec{c})$

$$C_i(r_g) = \begin{cases} 1 & \text{if } i = g \\ 0 & \text{otherwise} \end{cases}$$

(wire i is output of gate g)

* If $(a_0, a_1, \ldots, a_N)$ is assignment then we have to prove that $p(x)$ is divisible by $t(x)$

because we multiply coefficient with assign a_i only for input wires i of gate g → it gives the input to gate g

$$A(r_g) = A_0(r_g) + \sum_{k=1}^{m} a_k A_k(r_g)$$

↳ assignment (input to that gate)

= Input to gate g from Left.

because we multiply coefficient $c_{g,k,i}$ with assignment a_i only for input wires i of gate g ⇒ it gives the inputs to gate g

$$B(r_g) = B_0(r_g) + \sum_{k=1}^{m} a_k B_k(r_g)$$

= Input to gate g from Right

$$C(r_g) = a_g$$

= Output of gate g

$$P(r_g) = A(r_g) B(r_g) - C(r_g).$$

$P(r_5) = 0$, $P(r_6) = 0$ for our example

⇒ $(x - r_5)$ is factor of $P(x)$

$(x - r_6)$ is factor of $P(x)$

⇒ $\underbrace{(x - r_5)(x - r_6)}_{t(x)}$ is factor of $P(x)$

i.e $t(x)$ is a factor of $P(x)$.

25/11/22

m : No. of input and AND gates
M : set of AND gates for $g \in M$. A ~~set~~

We have to find $\vec{A}$, $\vec{B}$ and $\vec{C}$
$(A_0, A_1, \ldots, A_m, B_0, B_1, \ldots, B_m, C_0, C_1, \ldots C_m)$

We know the values $A_0(r_5)$ and $A_0(r_6)$
Using Shamir's ^v formula, we can find A_0
long range polynomial

$$A_0(x) = A_0(r_5)\frac{(x - r_6)}{(r_5 - r_6)} + A_0(r_6).\frac{(x - r_5)}{(r_6 - r_5)}$$

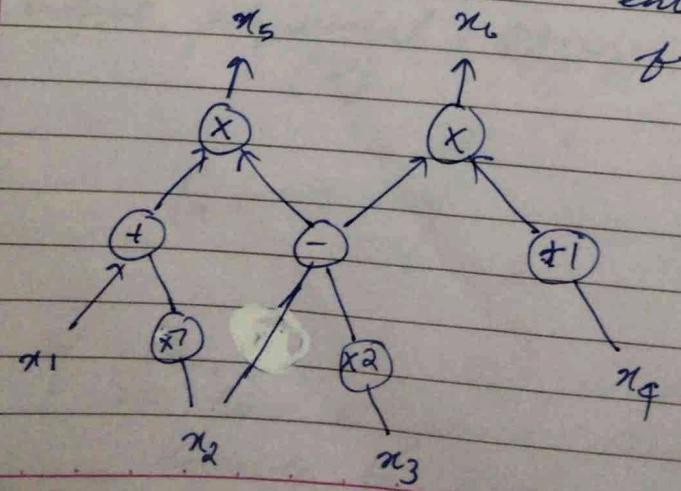
Degree of $A_0(x) = m - 1$
$= 2 - 1 = 1$

No. of random variables chosen = no. of o/ps from AND gates
$= m$

$A_0 \Rightarrow$ wire 0 i.e constant terms entering
We ~~also~~ have to include the constant inputs to the gates $g \in M$,

~~$A(x) = A_0(x) + \sum$~~ $A_0(r_5) = A_0(r_6) = 0$

$\Rightarrow A_0(x) = 0$ (no constants entering any gate from left)



For r_5 gate, input from left is x_1 (bcos it passes only through + gate and not any × gate before r_5).

$\therefore A_1(r_5) = 1$ (coefficient of x_1).

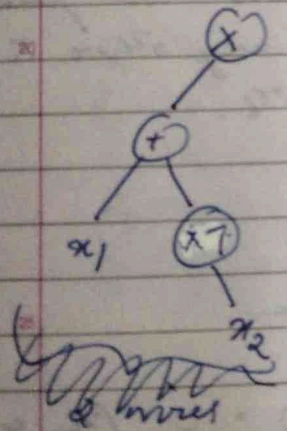
$A_1(r_6) = 0$

$$A_1(x) = A_1(r_5)\frac{(x - r_6)}{r_5 - r_6}$$

$$= \frac{x - r_6}{r_5 - r_6}$$

$A_{Q_i}(r_g) = C_{g,L,i}$ if $i \in I_{g,L}$

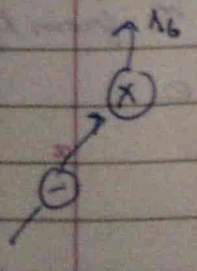
$A_2(r_g) = C_{g,L,2}$ ⇒ if 2nd wire from left enters × gate or not.



input x_2

⇒ 2nd wire from x_2 does enter r_5 gate

$\therefore A_2(r_5) = 7$ ~~(coffiaan~~

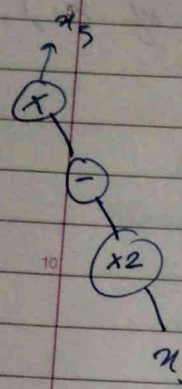


x_2 also enters r_6 from left

⇒ $A_2(r_6) = 1$.

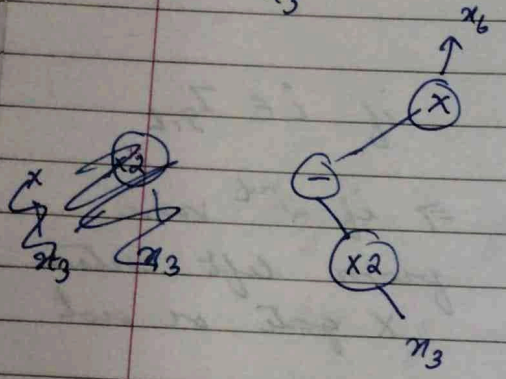
$$A_2(x) = 7\frac{(x - r_6)}{(r_5 - r_6)} + 1\frac{(x - r_5)}{(r_6 - r_5)}$$

A_3 → consider only wire 3 (x_3) to see if it reaches gates of EM from left. If yes, its coefficient to is $A_3(x_5)$



⇒ x_3 (wire 3) enters gate x_5 from right

⇒ $A_3(x_5) = 0$



x_3 enters from left of x_6

⇒ $A_3(x_6) = -2$

Also x_3 does not pass through any other x gate before x_6.

Similarly for A_4, A_5, A_6

$A_4(x_5) =$

$A_4(x_5) = 0$

$A_5(x_5) = 0$

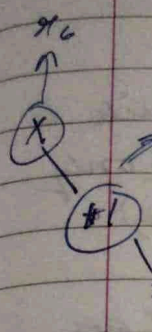
$A_6(x_5) = 0$

$A_4(x_6) =$

$A_4(x_6) = 0$ (x_4 enters x_6 from right)

$A_5(x_6) = 0$

$A_6(x_6) = 0$



$B_0(x_5) = 0$ → means constant

no constant (x_4 does not enters x_5 from R)

$B_0(x_6) = 1$

because ~~x_4 even though x_4 does not enter x_6,~~ we have constant +1 entering from right of x_6).

~~$B_4(x_6)$~~ ← $B_1(x_5) = 0$, $B_1(x_6) = 0$. → means x_1

$B_2(x_5) = -1$, $B_2(x_6) = 0$.

$B_3(x_5) = -2$, $B_3(x_6) = 0$

⋮

$$C_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

$C_0(x_5) = C_0(x_6) = 0$

$C_1(x_5) = C_1(x_6) = 0$

$C_2(x_5) = C_2(x_6) = 0$

$C_3(x_5) = C_3(x_6) = 0$

$C_4(x_5) = C_4(x_6) = 0$

$C_5(x_5) = 1$, $C_5(x_6) = 0$ } AND gates are only x_5, x_6.

$C_6(x_5) = 0$, $C_6(x_6) = 1$

We have a secret s which is not completely known to anyone (only shares of s are given to everyone)

Let G be the generator of an elliptic group.

$E \rightarrow$ encryption fn

$$\left.\begin{aligned} E(s) &= sG \\ E(s^2) &= s^2G \\ &\vdots \\ E(s^d) &= s^dG. \end{aligned}\right\} \text{ are all given/known}$$

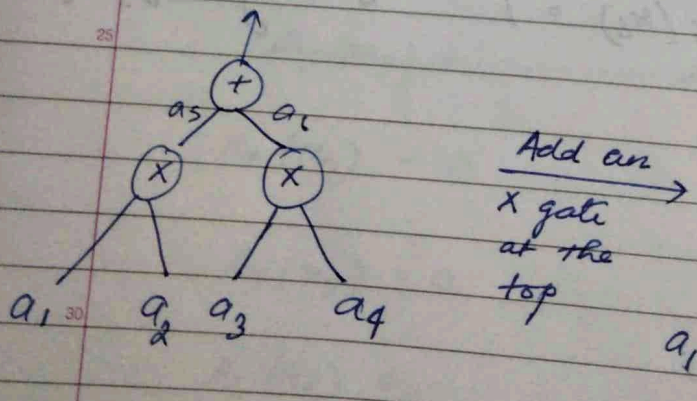
$$E(ax^2 + 3x + 5) = aE(x^2) + 3E(x) + 5$$

$E(as^d) = aE(s^d)$. $\quad E(5) = 5$.

Additive encryption

Since $E(s), \ldots, E(s^d)$ are all known, we can find $E(\text{any arbitrary point})$.

If last gate is not X gate



$m = 2$

final op $= a_5 + a_6$

cannot be computed (we considered only X gates)

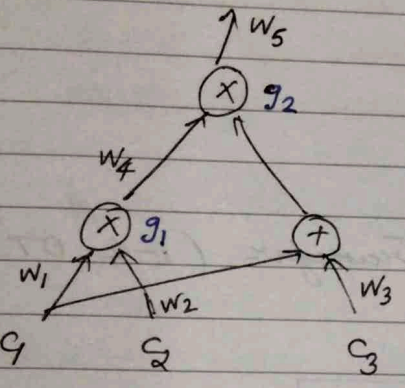
$m = 3$.

Pairing → generally elliptic groups used

If $P(x)$ is divisible by $T(x)$
then $P(x) = H(x).T(x)$

So it is enough to prove to verifier that $e(P(x)G, G) = e(H(x)G, T(x)G)$

Encryption is $E(A_i(\tau)) = A_i(\tau).G$.



Find QAP

$m = 2$ (g_1, g_2).

$A_0(r_4) = 0$	$A_0(r_5) = 0$
$A_1(r_4) = 1$	$A_1(r_5) = 0$ → because even though w_1 enters r_5 from left, it already passes through g_1 gate (we cannot consider it again)
$A_2(r_4) = 0$	$A_2(r_5) = 0$
$A_3(r_4) = 0$	$A_3(r_5) = 0$.
$A_4(r_4) = 0$	$A_4(r_5) = 1$.
$A_5(r_4) = 0$	$A_5(r_5) = 0$.

$A_0(x) = A_3(x) = A_2(x) = A_5(x) = 0$.

$A_1(x) = \dfrac{x - r_5}{r_4 - r_5}$

$B_0(r_4) = 0$	$B_0(r_5) = 0$
$B_1(r_4) = 0$	$B_1(r_5) = 1$
$B_2(r_4) = 1$	$B_2(r_5) = 0$.
$B_3(r_4) = 0$	$B_3(r_5) = 1$
$B_4(r_4) = 0$	$B_4(r_5) = 0$
$B_5(r_4) = 0$	$B_5(r_5) = 0$

→ 1 if output of r_4 gate is constant (wire 0)

$C_0(r_4) = 0$
$C_1(r_4) = 0$
$C_2(r_4) = 0$
$C_3(r_4) = 0$
$C_4(r_4) = 1$
$C_5(r_4) = 0$

$C_0(r_5) = 0$
$C_1(r_5) = 0$
$C_2(r_5) = 0$
$C_3(r_5) = 0$
$C_4(r_5) = 0$
$C_5(r_5) = 1$

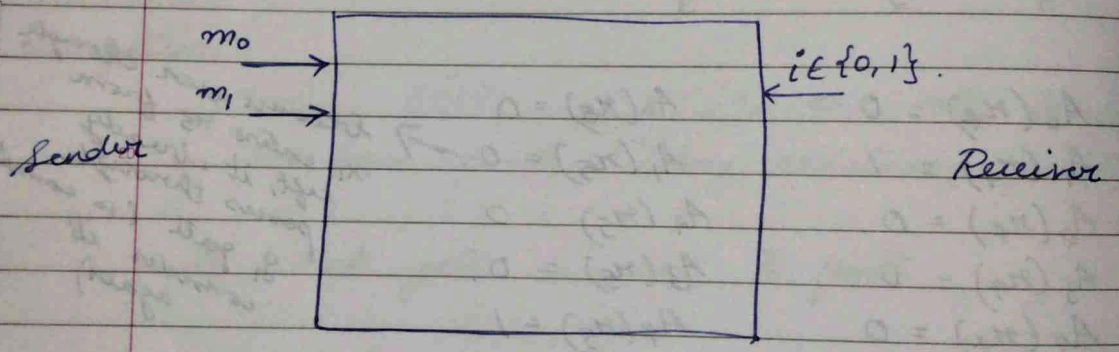
$C_0(x) = C_1(x) = C_2(x) = C_3(x) = 0.$

$$C_4(x) = \frac{x - r_5}{r_4 - r_5}$$

$$C_5(x) = \frac{x - r_4}{r_5 - r_4}$$

26/11/22

Oblivious Transfer (1-2 OT protocol)



Requirements :

1. If receiver asks for m_i then he should not know anything about m_{1-i}
2. Sender should not which message the receiver is asking

El Gamal Cryptosystem :

Enc(m) : $C = \{C_1 = g^k , C_2 = m y^k\}$.

Dec(x, C) : $m = \frac{C_2}{C_1^x}$ $\quad ((g^x)^k = y^k)$.

x is private key known to the recv.

Receiver :

Choose x_i and compute $y_i = g^{x_i}$

Randomly choose y_{1-i}

$PB = (y_i, y_{1-i})$. are both sent to the sender

Sender does not know which key out of y_i, y_{1-i} corresponds to x_i

Sender :

~~(Say $y_i = y_0$).~~

Encrypts m_0 using y_0, m_1 using y_1.

~~$C_1 = \{g^{k_1}, m_0 y_0^{k_1}\}$.~~

Enc(m_i, y_i) : $C_1 = g^{k_i}$ $\quad C_2 = m_i y_i^{k_i}$

Enc(m_{1-i}, y_{1-i}) : $C_3 = g^{k_{1-i}}$ $\quad C_4 = m_{1-i} \cdot y_{1-i}^{k_{1-i}}$

sends $\underbrace{(C_1, C_2)}_{C_{m_i}}$ and $\underbrace{(C_3, C_4)}_{C_{m_{1-i}}}$ to recv

Receiver:

Knows x_0 but not x_1.
⇒ Receiver can decrypt C_0 but not C_1.

Drawback:

Receiver can cheat and compute y_i and y_{1-i} after choosing 2 private keys x_i and x_{1-i}

same cohort

Improved scheme:

Sender chooses a public key y and sends it to receiver

Receiver chooses x_i and computes y_i
He then finds y_{1-i} such that

$$y_i \cdot y_{1-i} = y.$$

↓

Finding public key from this PB has only neg. prob.

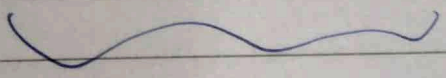
SimHash:

2 similar things ⇒ hash is same so that they are placed in the same group

(unlike ideal hash that gives a completely different output even for a small change in msg).

(Ex):

& My search topics	Cohorts	Ads.
Book	m_0	1
Travel, Journey (same cohort)	m_1	2 → about your travel
Movie		3
		4



Recv

↳ I have to fetch adds corresponding to m_1 ⇒ basically ask for ad 2 without revealing it to sender