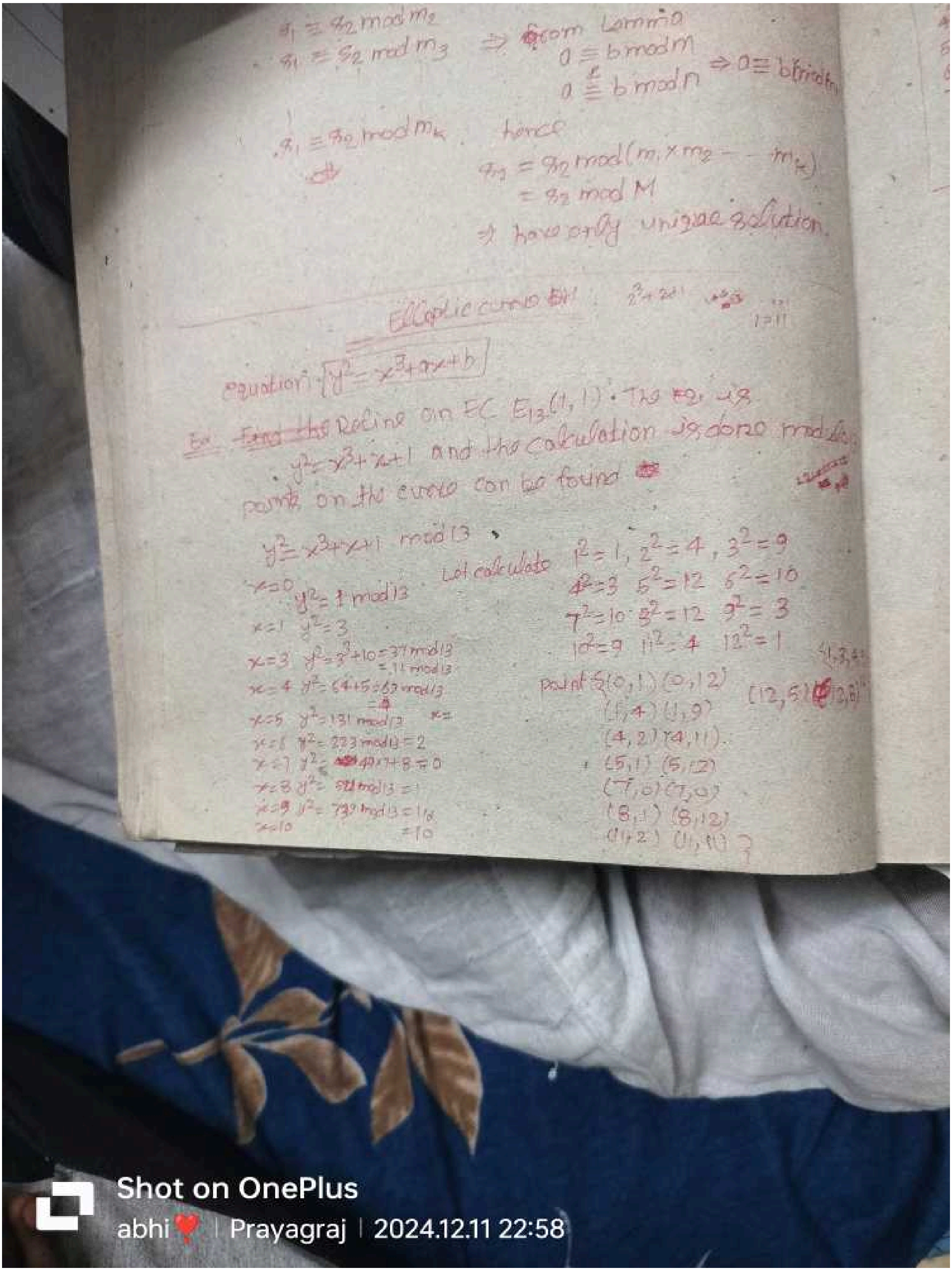


$r_1 \equiv r_2 \bmod m_2$

$r_1 \equiv r_2 \bmod m_3$ ⇒ from Lemma

$a \equiv b \bmod m$

$a \overset{?}{\equiv} b \bmod n$ ⇒ $a \equiv b \bmod m_n$

$r_1 \equiv r_2 \bmod m_k$. hence

$r_1 = r_2 \bmod (m_1 \times m_2 - - m_k)$

$= r_2 \bmod M$

⇒ have only unique solution.

Elliptic curve EH $2^3+2\geq1$ 17:11

equation: $\boxed{y^2 = x^3+ax+b}$

Ex: ~~Find the~~ Define on EC $E_{13}(1,1)$. The eq is $y^2 = x^3+x+1$ and the calculation is done modulo points on the curve can be found

$y^2 = x^3+x+1 \bmod 13$

Let calculate $1^2=1,\ 2^2=4,\ 3^2=9$

$4^2=3\ \ 5^2=12\ \ 6^2=10$

$7^2=10\ \ 8^2=12\ \ 9^2=3$

$10^2=9\ \ 11^2=4\ \ 12^2=1$

$x=0$ $y^2 = 1 \bmod 13$

$x=1$ $y^2=3$

$x=3$ $y^2 = 3^3+10 = 37 \bmod 13$
$= 11 \bmod 13$

$x=4$ $y^2 = 64+5 = 69 \bmod 13$
$= 4$

$x=5$ $y^2 = 131 \bmod 13$ $x=$

$x=6$ $y^2 = 223 \bmod 13 = 2$

$x=7$ $y^2 = 49\times7+8 = 0$

$x=8$ $y^2 = 521 \bmod 13 = 1$

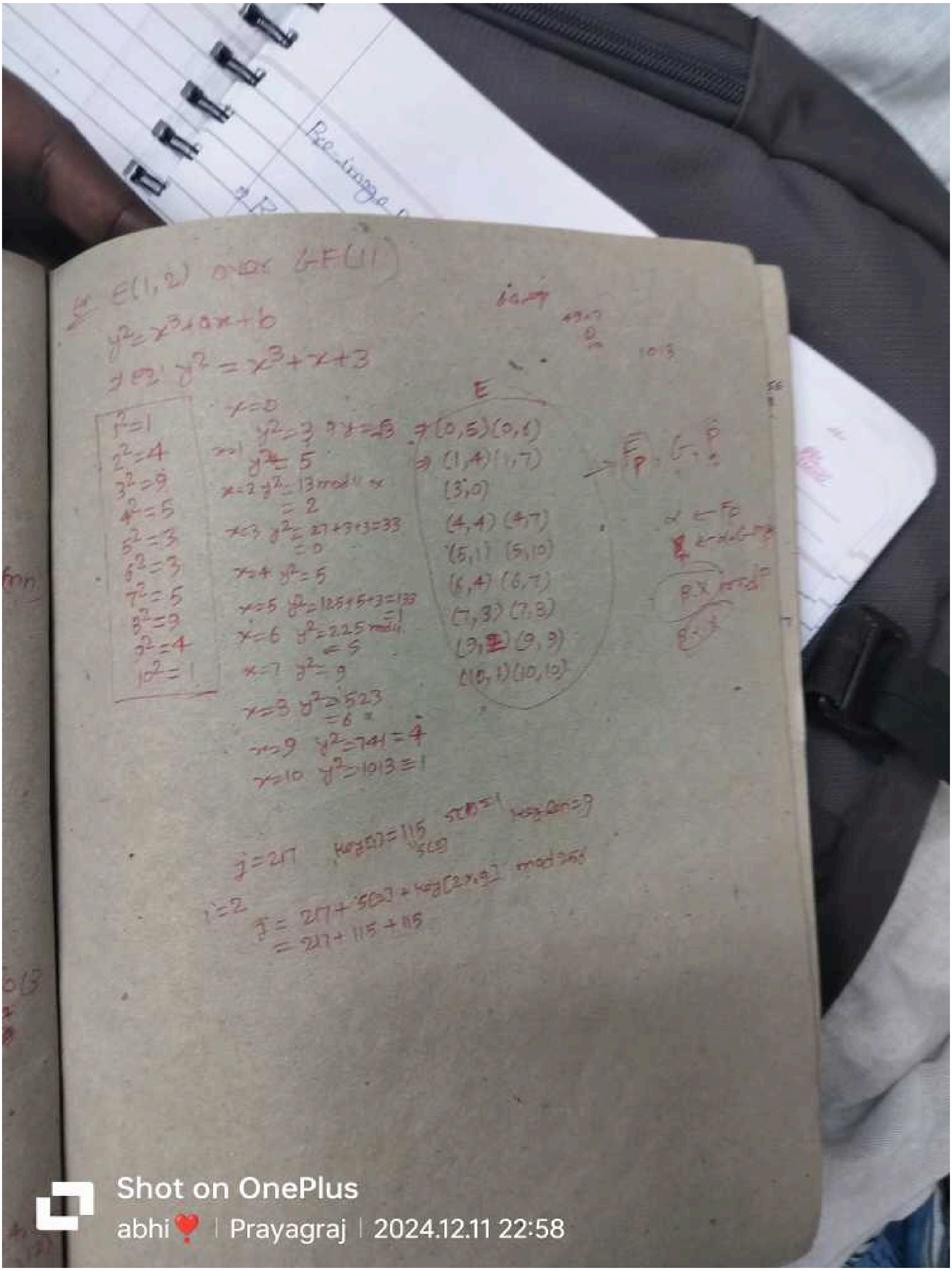
$x=9$ $y^2 = 739 \bmod 13 = 10$

$x=10$ $=10$

points $\{(0,1)\ (0,12)$
$(1,4)\ (1,9)$
$(4,2)\ (4,11)$
$(5,1)\ (5,12)$
$(7,0)\ (7,0)$
$(8,1)\ (8,12)$
$(11,2)\ (11,11)\}$

$(12,5)\ (12,8)$

1,3,4



Q. E(1,2) over GF(11)

$y^2 = x^3 + ax + b$

y eq: $y^2 = x^3 + x + 3$

$1^2 = 1$

$2^2 = 4$

$3^2 = 9$

$4^2 = 5$

$5^2 = 3$

$6^2 = 3$

$7^2 = 5$

$8^2 = 9$

$9^2 = 4$

$10^2 = 1$

x=0

$y^2 = 3 \Rightarrow y = \sqrt{3}$

x=1 $y^2 = 5$

x=2 $y^2 = 13 \bmod 11$ ✗ $= 2$

x=3 $y^2 = 27+3+3 = 33$ $= 0$

x=4 $y^2 = 5$

x=5 $y^2 = 125+5+3 = 133$ $= 1$

x=6 $y^2 = 225 \bmod 11$ $= 5$

x=7 $y^2 = 9$

x=8 $y^2 = 523$ $= 6$ ✗

x=9 $y^2 = 741 = 4$

x=10 $y^2 = 1013 \equiv 1$

E

⇒ (0,5) (0,6)

⇒ (1,4) (1,7)

(3,0)

(4,4) (4,7)

(5,1) (5,10)

(6,4) (6,7)

(7,3) (7,8)

(9,2) (9,9)

(10,1) (10,10)

→ $\frac{E}{p}$, (-, p)

α ← F_p

β ← α·G=β

P.X mod P

β·x

6a+9

49+7

8

1013

58

j=217 key[i]=115 S[9] S[i]=1 key[len]=9

i=2

j = 217 + S[2] + key[2%9] mod 256

= 217 + 115 + 115

Modular Exponentiation algorithm:-

basically the main agenda is to calculate,

$$\boxed{x = a^b \bmod n}$$

For basic:

```
x=1
for(i=0 to b)
{
    x=(x×a)mod n
}
return x
```

Ex: $9^8 \bmod 11$

$8 = 1000$

1	0	0	0
b_3	b_2	b_1	b_0

$x \leftarrow 1$
$y \leftarrow 9$

$i=0 \quad y \leftarrow (9\times 9) \bmod 11 = 4$
$i=1 \quad y \leftarrow (4\times 4) \bmod 11 = 5$
$i=2 \quad y \leftarrow (5\times 5) \bmod 11 = 3$
$i=3 \quad b_3=1 \; x \leftarrow (1\times 3) \bmod 11 = 3$

convert b in binary

$\Rightarrow b$	b_{k-1}	b_{k-2}	-	- - -	b_1	b_0
	2^{k-1}	2^{k-2}	—	- -	2^1	2^0

$$\Rightarrow x = a^b \bmod n$$
$$= (a^{b_0 2^0 + b_1 2^1 + b_2 2^2 + \cdots + b_{k-1} 2^{k-1}}) \bmod n$$
$$= (a^{b_0 2^0} \cdot a^{b_1 2^1}\, a^{b_2 2^2} \cdots a^{b_{k-1} 2^{k-1}}) \bmod n$$
$$= \left(\prod_{i=0}^{k-1} a^{b_i \cdot 2^i}\right) \bmod n$$

for binary bit $b_i \begin{matrix} \nearrow 0 \\ \searrow 1 \end{matrix}$ $\Rightarrow$ when $b_i=0$ then $a^{0\cdot 2^i} = 1$
when $b_i = 1$ then $a^{1 \cdot 2^i} = (a^2)^i$

so

$$\Rightarrow x = \left(\prod_{\substack{i=0 \\ b_i \neq 0}}^{k-1} (a^2)^i\right) \bmod n$$

```
x ← 1, y ← a
for(i=0 to k-1)
{
    if(b_i == 1)
    {
        x ← (x × y) mod n
    }
    y ← (y × y) mod n
}
return x
```

Euclid algorithm

→ This algorithm is use to find the GCD of two number

→ GCD(a, b) = ?

q $r_2)\overline{r_1}$
r_2

```
$r_1 \leftarrow a$; $r_2 \leftarrow b$
~~for~~ while($r_2 > 0$)
{
    $q = \lfloor r_1 / r_2 \rfloor$
    $r = r_1 - q \cdot r_2$
    $r_1 \leftarrow r_2$
    $r_2 \leftarrow r$
}
return $r_1$
```

Extended euclid algorithm

→ This algorithm is extended version of computing GCD as well as multiplicative inverse of given two numbers

Ex: $a^{-1} \bmod b$

```
$r_1 \leftarrow a$; $r_2 \leftarrow b$
$t_1 \leftarrow 0$; $t_2 \leftarrow 1$
while($r_2 > 0$)
{
    $q = \lfloor r_1 / r_2 \rfloor$
    $r = r_1 - q \cdot r_2$
    $t = t_1 - q \cdot t_2$
    $r_1 \leftarrow r_2$
    $r_2 \leftarrow r$
    $t_1 \leftarrow t_2$
    $t_2 \leftarrow t$
}
~~return~~
if($r_1 == 1$)
{
    return $t_1$;
}
else
{
    return NULL
}
```

Fermat's theorem

if p is prime number and $a \in Z$ such that

$p \nmid a$ (p does not divides a)

then $\boxed{a^{p-1} \equiv 1 \bmod p}$

Proof:-

Let $Z_p^* = \{1, 2, 3, \text{-----} (p-1)\}$ since p is prime

multiply with a in each elements of set

$X = \{a, 2a, 3a, \text{----} a(p-1)\} \bmod p$

if apply mod p in each element of set

$\Rightarrow X = \{a \bmod p, 2a \bmod p, \text{----} a(p-1) \bmod p\}$

since a is coprime with p hence all the elements of X belong to Z_p^* only

$\Rightarrow$ we can say that

$X = Z_p^* \Rightarrow 1 \le x_i \le (p-1)$

~~2 4 5~~

$\Rightarrow$ Let assume $x_i \equiv 0 \bmod p$

$\Rightarrow k \cdot a \equiv 0 \bmod p$ where $k \in \{1 \text{---} (p-1)\}$

from this either $\underset{\substack{\downarrow \\ \text{not possible} \\ \text{since } k \text{ is from } Z_p^*}}{k \bmod p = 0}$ or $\underset{\substack{\Downarrow \\ p \mid a \text{ which} \\ \text{contradict} \\ \text{the assumption}}}{a \bmod p = 0}$

Let assume

* $x_i \equiv x_j \bmod p$ where $i \neq j$

$\Rightarrow k_1 \cdot a \equiv k_2 a \bmod p \Rightarrow (k_1 - k_2) a \equiv 0 \bmod p$

$\Rightarrow$ either $(k_1 - k_2) \bmod p = 0$ or $\underset{\substack{\Downarrow \\ p \mid a \text{ contradict the} \\ \text{assumption}}}{a \bmod p = 0}$

*From above assumptions we concluded that

$X \equiv Z_p^*$

Lets multiply all element of X & Z_p^* both side

$\Rightarrow$

$X = \{a, 2a, 3a, \text{---} \; a(p-1)\}$

$Z_p^* = \{1, 2, 3, \text{---} \; (p-1))\}$

$\Rightarrow$

$\{a.(2a).(3a) \text{---} a(p-1)\} \bmod p \equiv \{1.2.3. \text{---} (p-1)\} \bmod p$

$\Rightarrow a^{p-1}(1.2.3 \text{---} (p-1)) \equiv \{1, 2 \text{---} (p-1)\} \bmod p$

eliminate common termp from both side

$\Rightarrow \boxed{a^{p-1} \equiv 1 \bmod p}$

proved

$\Rightarrow \frac{a^{p-1}}{a} \equiv \frac{1}{a} \bmod p \Rightarrow \boxed{a^{p-2} \equiv a^{-1} \bmod p}$

Euler's totient function:-

if n is a prime number then $\phi(n) = (n-1)$

no.ofelements d[illegible] with'n'

Euler's theorem:-

1st version: if $\gcd(a, n) = 1$ then $\boxed{a^{\phi(n)} \equiv 1 \bmod n}$

2nd version: if $n = p \times q$ ~~where p & q are coprime~~

then $\boxed{a^{k\phi(n)+1} \equiv a \bmod n}$ wher $k \in Z_n$

Proof:

$Z_n^* = \{x_1, x_2 - - - - - - x_{\phi(n)}\}$

there only $\phi(n)$ values in Z_n^*

where $\phi(n) = (p-1)(q-1)$

multiply all the values of Z_n^* with a

$X = \{ax_1, ax_2 - - - - - , ax_{\phi(n)}\}$

apply mod n

$X = \{\underbrace{ax_1 \bmod n, - - - - - ax_{\phi(n)} \bmod n}_{\text{all elements lies in } Z_n^*}\}$

⇒ $X = Z_n^*$

take product both side of all elements

⇒ $(ax_1 \cdot ax_2 - - - - - ax_{\phi(n)}) \bmod n \equiv (x_1, x_2 - - x_{\phi(n)}) \bmod n$

⇒ $\boxed{a^{\phi(n)} \equiv 1 \bmod n}$ 1st. proved.

case1: Let assume a is coprime with n

then $a^{\phi(n)} \equiv 1 \bmod n$

apply exp(k) both side

⇒ $a^{k\phi(n)} \equiv 1^k \bmod n$

since a is coprime means a^{-1} exist under mod n

⇒ multiply both side with a

⇒ $\boxed{a^{k\phi(n)+1} \equiv a \bmod n}$ proved

Case2: Lets assume a is not coprime with n.

⇒ then $\quad$ since $n = p \times q$

case1: $a \in \{1\times p, 2\times p, ---- p\times(q-1)\}$

case2: $a \in \{1\times q, 2\times q, ----- q\times(p-1)\}$

Let take case1:

⇒ $a = K_i p$ wher $K_i \in \{1, 2 --- (q-1)\}$

from Fermat's Law $q \nmid a$

⇒ $a^{q-1} \equiv 1 \bmod q$

~~here~~ $q-1 = \phi(n)$ here

⇒ $a^{\phi(n)} \equiv 1 \bmod q$

apply $\phi(p)$ in power both side

$a^{\phi(q)\cdot\phi(p)} \equiv 1 \bmod q$

⇒ $a^{\phi(n)} \equiv 1 \bmod q$

⇒ $a^{K\phi(n)} \equiv 1 \bmod q$ $\quad$ ⇒ $q)\, a^{K\phi(n)}\, (K_2$

$\overline{\quad 1 \quad}$

⇒ $a^{K\phi(n)} = (qK_2 + 1)$

we can write

$1 = a^{K\phi(n)} - qK_2$

⇒ $a^{K\phi(n)} = \underline{1 + qK_2}$

multiply with a both side

⇒ $a^{K\phi(n)+1} = aqK_2 + a$

$a^{K\phi(n)+1} = K_1 \cdot K_2 \cdot p \cdot q + a$

$a^{K\phi(n)+1} = K_3 \cdot n + a$

apply mod n both side

$a^{K\phi(n)+1} \equiv \overset{0}{\cancel{K_3 n \bmod n}} + a \bmod n$

$$\boxed{a^{K\phi(n)+1} \equiv a \bmod n}$$

→ Hash function

designing hash function

Miller Rabbin Algorithm:- for primality testing

Input: n where $n > 2$

output: Composite (Yes) / No (Prime)

Steps:

Calculate k and m from n
such that $(n-1) = 2^k \cdot m$ [note: m is odd]

```
1.  $a \xleftarrow{R} \{2, \ldots (n-1)\}$
2.  $b \leftarrow a^m \bmod n$
3.  if ($b \equiv 1 \bmod n$) {return No}
4.      for($i=0$ to $(k-1)$)
        {
5.       if ($b \equiv -1 \bmod n$) then return No;
6.       else
          {
7.          $b \leftarrow b^2 \bmod n$
          }
        }
8.      return Yes ;
```

Ex $n = 561$

$n-1 = 560 \Rightarrow 2^4 \cdot 35$

$k = 4,\ m = 35$

1. take $a = 2$ $\{2 \ldots 560\}$
2. $b \leftarrow 2^{35} \bmod 561 \equiv 263$

$$\begin{array}{r|l} 2 & 560 \\ 2 & 280 \\ 2 & 140 \\ 2 & 70 \\ & 35 \end{array}$$

2^9 560 512 48

2^{10} 1024 560 464

$2^{560} \equiv 1 \bmod 561$

$2^5 \cdot 35$

Proof: Yes biased Monte-carlo algorithm

Assume: n is prime

from algo $a^m \not\equiv 1 \bmod n$

for $i=0$ $\quad a^m \not\equiv -1 \bmod n$

$i=1$ $\quad a^{2m} \not\equiv -1 \bmod n$

e-image problem Resistant:
→ Hash function
designing hash function

= p×2

Miller Rabbin Algorithm: for primality testing

Input: n where $n > 2$

output: Composite (Yes) / No (Prime)

Steps:

calculate k and m from n
such that $(n-1) = 2^{k} \cdot m$ [note: m is odd]

```
1.  $a \xleftarrow{R} \{2, ---- (n-1)\}$
2.  $b \leftarrow a^{m} \bmod n$
3.  if ($b \equiv 1 \bmod n$) { return No }
4.      for(i=0 to(k-1)}
        {
5.        if ($b \equiv -1 \bmod n$) then return No;
6.        else
          {
7.          $b \leftarrow b^{2} \bmod n$
          }
8.      }
        return Yes ;
```

Ex $n = 561$

$n-1 = 560 \Rightarrow 2^{4} \cdot 35$

$k=4$, $m=35$

1. take $a=2$ $\{2 - - - 560\}$
2. $b \leftarrow 2^{35} \bmod 561 \equiv 263$

$$
\begin{array}{r|l}
2 & 560 \\
2 & 280 \\
2 & 140 \\
2 & 70 \\
 & 35
\end{array}
$$

2^{9} 560
512
(48)

2^{10}

1024
560

$2^{560} \equiv 1 \bmod 561$

$2^{4} \cdot 35$

Proof: Yes biased Monte-carlo algorithm

Assume: n is prime

from algo $a^{m} \not\equiv 1 \bmod n$

for $i=0$ $\quad a^{m} \not\equiv -1 \bmod n$

$i=1$ $\quad a^{2m} \not\equiv -1 \bmod n$

$i=k-1$ $\quad a^{[illegible]} \not\equiv -1 \bmod n$

Accord^{n} to Fermat's law

$a^{n-1} \equiv 1 \bmod n.$ where $n \nmid a$

$\boxed{n-1 = 2^{k} \cdot m}$

Given $a^{2^{(k-1)} m} \not\equiv -1 \bmod n$

$a^{2^{k} m} \equiv 1 \bmod n$

apply root both side

$a^{2^{k-1} m} \equiv \sqrt{1} \bmod n$

and we know $a^{2^{k-1} \cdot m} \not\equiv -1 \bmod n$

hence $a^{2^{k-1} \cdot m} \equiv 1 \bmod n$

apply $\sqrt{\ }$ on both sid

$a^{2^{k-2} \cdot m} \equiv \sqrt{1} \bmod n$

$\rightarrow a^{2^{k-2} m} \equiv 1 \bmod n$ it contradict our assuption

→

...age problem Resistant:
→ Hash function

... designing hash function

RSA Digital signaturer

keyGen:

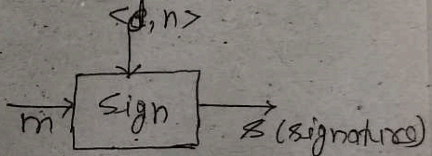
1. $p \leftarrow$ largprime, $q \leftarrow$ largeprime
2. compute $n \leftarrow p \times q$
3. Compute $\phi(n) \leftarrow \phi(p) \times \phi(q) = (p-1)(q-1)$
4. $d \xleftarrow{R} Z^*_{\phi(n)}$
5. $e \leftarrow d^{-1} \bmod \phi n$

Output: privatekey$<d,n>$
publickey$<e,n>$

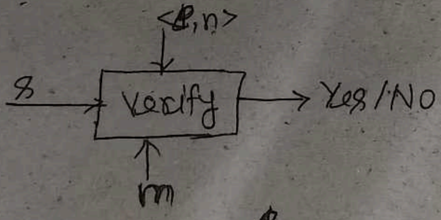
Sign:- Given: $m, <d,n>$

$m \in Z^*_{\phi(n)}$

1. calculate the hash of m
2. $s \leftarrow \{H(m)\}^d \bmod n$



Verify:



⇒ calculate $(s)^e \bmod n$ ~~= H(m) mo~~

if $s^e \bmod n == H(m) \bmod n$ then return yes
otherwise No.

Correctness Proof:

take LHS. $s^e \bmod n = (H(m)^d)^e \bmod n$

$= H(m)^{ed^{-1}} \bmod n$

$= H(m) \bmod n$

$=$ RHS proved

Soundness Proof:-

RSA digital signature sounds as long as RSA problem is hard and underlaying hash function is collision resistant.

1st: define Attacker problem

Attacker probl:-

Given: $\langle e,n \rangle$, $q_1^{\in \text{Poly}}$ (no. of hash queries), $q_2^{\in \text{poly}}$ (no. of sign queries)

Find : (m', s')

s.t. : $s^e \equiv$ ~~A~~ $H(m') \bmod n$

where $(m', s') \neq (m_i, s_i) \ \forall i$

2nd: RSA problem:-

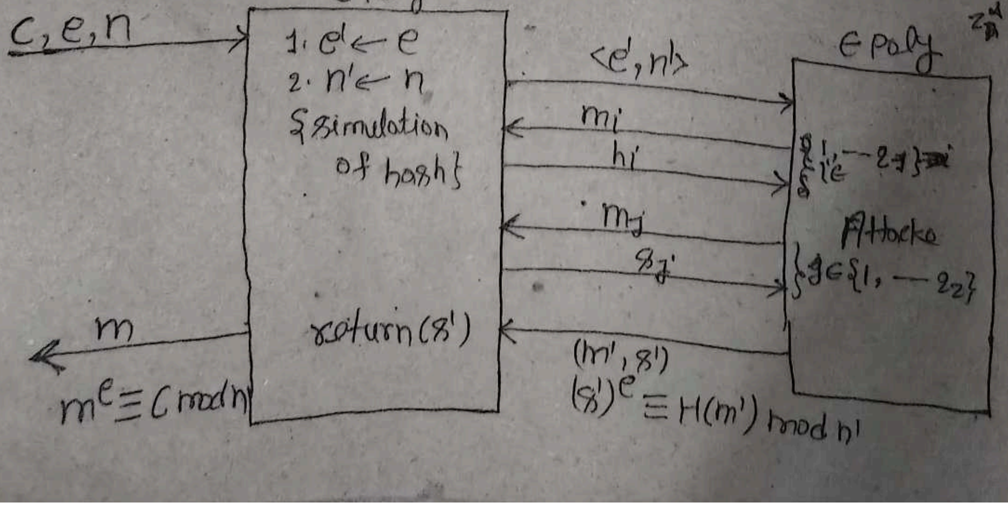
Given: $c, \langle e, m \rangle$

Find: m

s.t. : $c \equiv m^e \bmod n$ is hard

Start of proof:

Let assume RSA problem is easy, then reduce the ~~Digital~~ attacker problem to RSA problem



→ Hash function

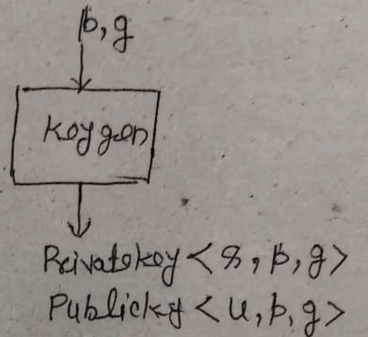
for designing hash function

than

ElGammal Encryption Schemer

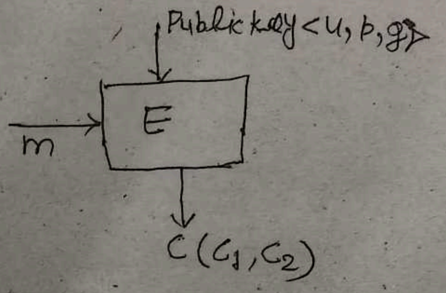
1. Key Gen:

   1. $p \leftarrow$ largprime
   2. $g \leftarrow$ generator of Z_p^*
   3. $x \xleftarrow{R} Z_p^*$
   4. $u \leftarrow g^x \bmod p$



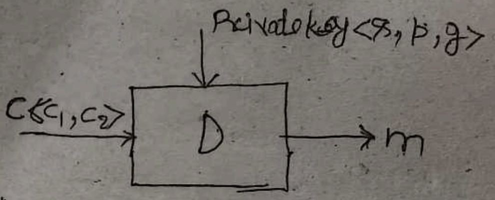
2. Encryption:

   1. $r \xleftarrow{R} Z_p^*$
   2. $c_1 \leftarrow g^r \bmod p$
   3. $c_2 \leftarrow (m \cdot u^r) \bmod p$



3. Decryption:

   1. Calculate $c_1^x \bmod p$
   2. Calculate $(c_2 \cdot (c_1)^{-x}) \bmod p$
   3. $m = (c_2 \cdot (c_1^x)^{-1}) \bmod p$



Correctness Proof:-

R.H.S $(c_2 \cdot c_1^{-x}) \bmod p$

$[\{m \cdot u^r\} \cdot (g^{-r})^{x}] \bmod p$

$= [m \cdot u^r \cdot g^{-rx}] \bmod p$ put $u = g^x$

$= [m \cdot (g^x)^r \cdot g^{-rx}] \bmod p$

$= [m \cdot g^{rx} \cdot g^{-rx}] \bmod p = [m \cdot g^0] \bmod p$

$= m$ = L.H.S proved

Soundness proof of ElGammal ENcryption:

- ElGammal Encryption sounds as long as Diffi-H problem is hard.

Attacker problem:

Given: g, p, c, u

Find: m

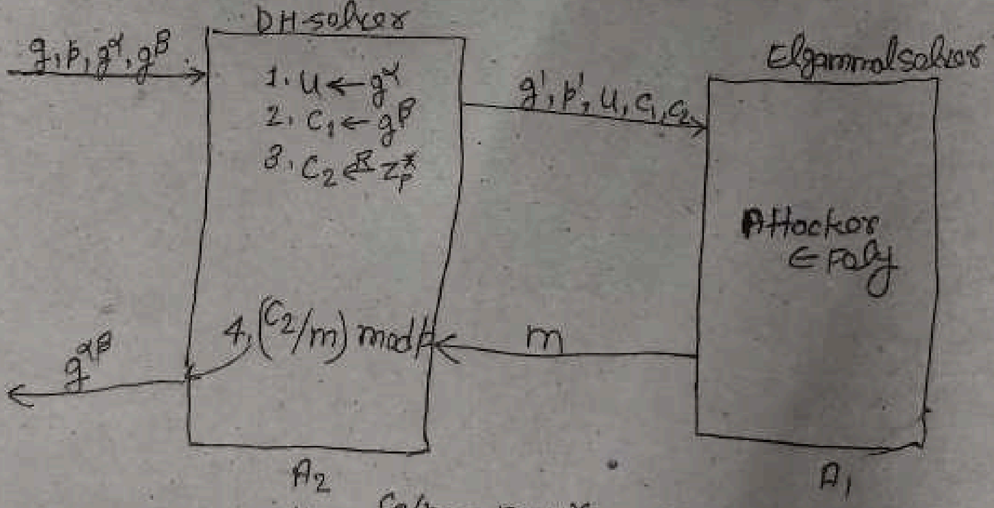
S.t. : decryption of c under the pvt key corresponding to 'u' is 'm'

Assumption: DH problem is hard

DH-problem:

Given: $g, p, g^{\alpha}, g^{\beta}$

Find: $g^{\alpha\beta} \bmod p$

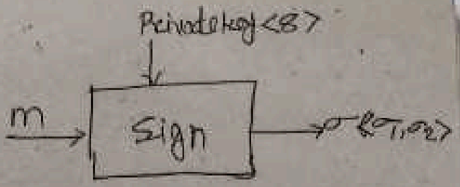

DH-solver
$g, p, g^{\alpha}, g^{\beta}$
1. $u \leftarrow g^{\alpha}$
2. $c_1 \leftarrow g^{\beta}$
3. $c_2 \xleftarrow{R} Z_p^*$
g', p', u, c_1, c_2
Elgammalsolver
Attacker $\in$ Poly
4. $(c_2/m) \bmod p$
m
$g^{\alpha\beta}$
A_2
A_1

A_2 $\Downarrow$ Poly time

$$c_2/m = \frac{m.u^r}{m} = u^r = g^{r\alpha}$$

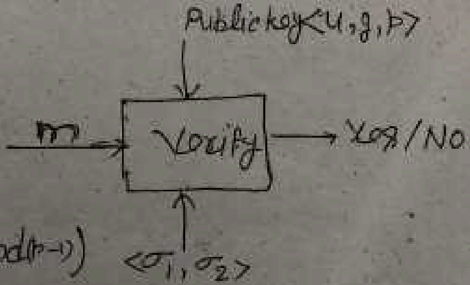
ed for designing hash function
random oracle model then

Sign:



1. $r \xleftarrow{R} Z^*_{(p-1)}$
2. $\sigma_1 \leftarrow g^r \bmod (p-1)$
3. $\sigma_2 \leftarrow ((m - s\sigma_1) r^{-1}) \bmod (p-1)$

3. Verify:



if($(u^{\sigma_1} . \sigma_1^{\sigma_2}) \bmod (p-1) = g^m \bmod (p-1)$)
then return ~~yes~~ otherwise
false.

Correctness Proof:

L.H.S $u^{\sigma_1} . \sigma_1^{\sigma_2} \bmod (p-1)$

$= [u^{\sigma_1} . \sigma_1^{(m - s\sigma_1).r^{-1}}] \bmod (p-1)$

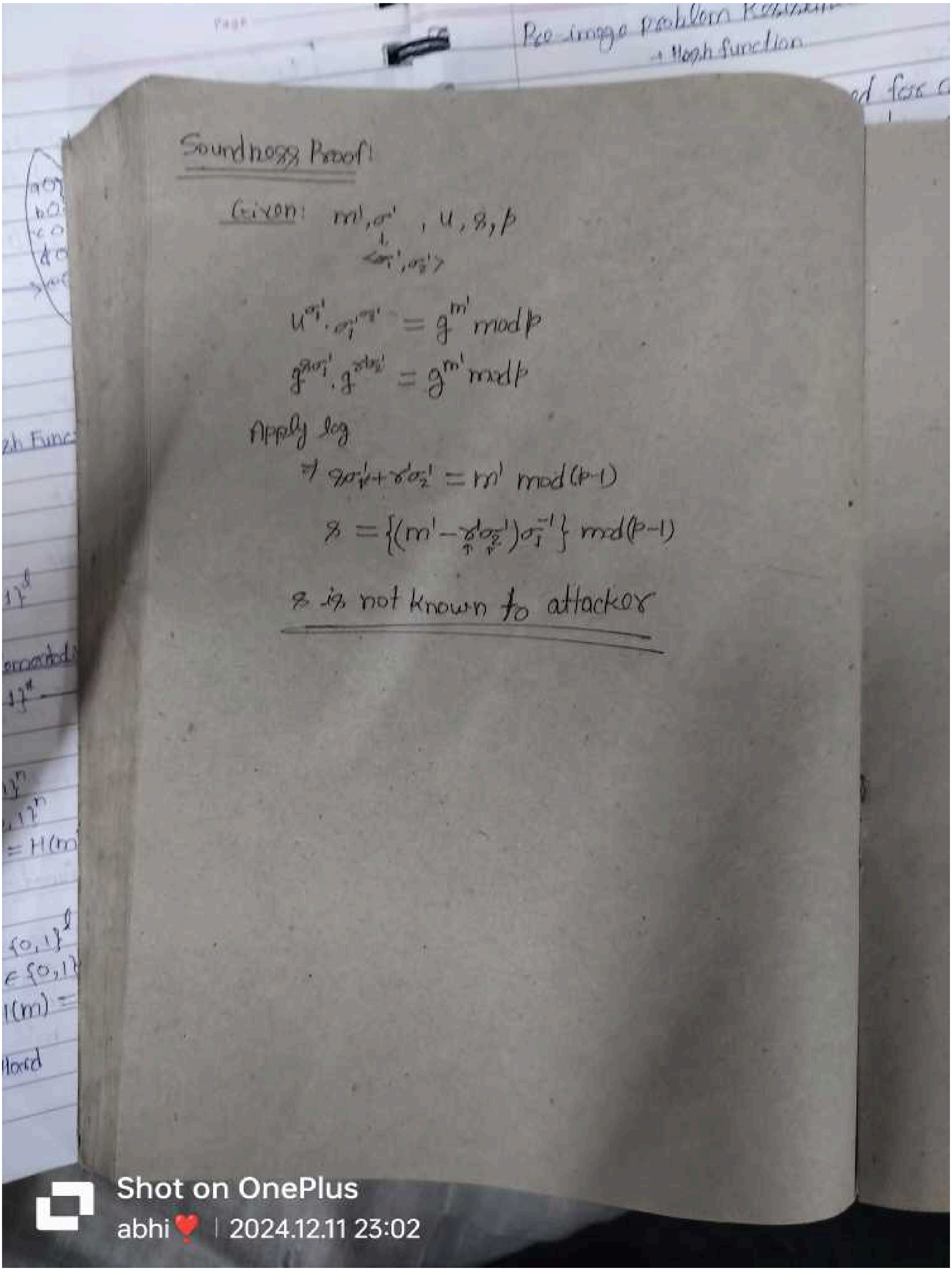
$= [u^{\sigma_1} . g^{r(m - s\sigma_1)r^{-1}}] \bmod (p-1)$ $\because \sigma_1 = g^r$

$= [u^{\sigma_1} . g^{m - s\sigma_1}] \bmod (p-1)$ put $u = g^s \bmod p$

$= [(g^s)^{\sigma_1} . g^{m - s\sigma_1}] \bmod (p-1)$

$= [g^{s\sigma_1} . g^{m - s\sigma_1}] \bmod (p-1)$

$= g^m \bmod (p-1)$



Soundness Proof:

Given: $m', \underset{\substack{\downarrow \\ \langle \sigma_1', \sigma_2' \rangle}}{\sigma'}, u, x, p$

$$u^{\sigma_1'} \cdot \sigma_1'^{\sigma_2'} = g^{m'} \bmod p$$

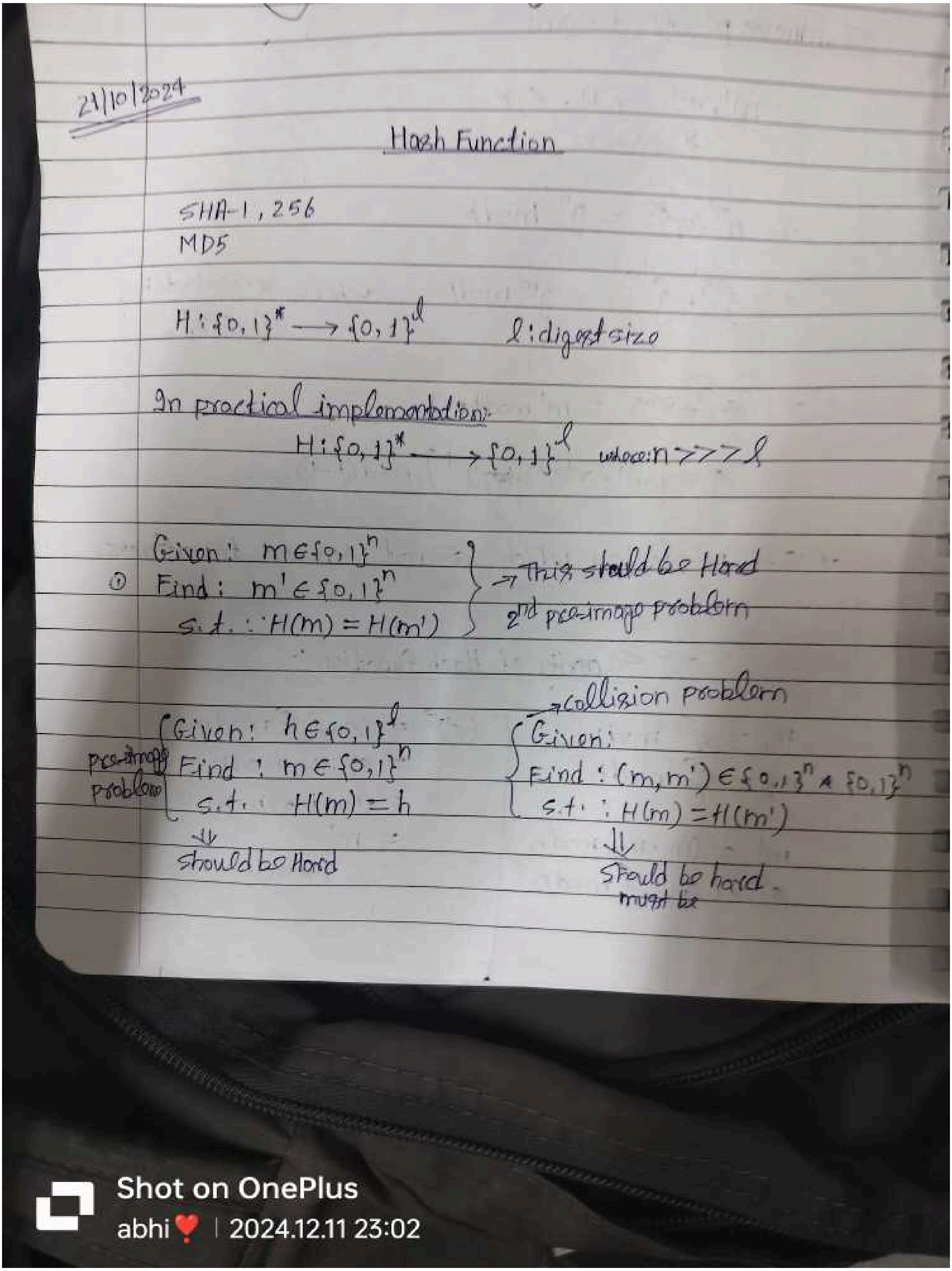
$$g^{x\sigma_1'} \cdot g^{\gamma\sigma_2'} = g^{m'} \bmod p$$

Apply log

$$\Rightarrow x\sigma_1' + \gamma'\sigma_2' = m' \bmod (p-1)$$

$$x = \{(m' - \underset{\uparrow}{\gamma'}\underset{\uparrow}{\sigma_2'})\sigma_1'^{-1}\} \bmod (p-1)$$

x is not known to attacker



21/10/2024

Hash Function

SHA-1, 256
MD5

$H : \{0,1\}^* \longrightarrow \{0,1\}^l$ l : digest size

In practical implementation:

$H : \{0,1\}^* \longrightarrow \{0,1\}^l$ where $n >>> l$

① $\left.\begin{array}{l} \text{Given: } m \in \{0,1\}^n \\ \text{Find: } m' \in \{0,1\}^n \\ \text{s.t.: } H(m) = H(m') \end{array}\right\}$ → This should be Hard
2nd pre-image problem

pre-image problem $\left\{\begin{array}{l} \text{Given: } h \in \{0,1\}^l \\ \text{Find: } m \in \{0,1\}^n \\ \text{s.t. } H(m) = h \end{array}\right.$

⇓
should be Hard

→ collision problem

$\left\{\begin{array}{l} \text{Given:} \\ \text{Find: } (m, m') \in \{0,1\}^n \times \{0,1\}^n \\ \text{s.t.: } H(m) = H(m') \end{array}\right.$

⇓
Should be hard.
must be

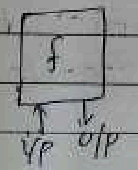
Pre-imago problem Resistant:
→ Hash function

⇒ Radom Oracle model : used for designing hash function
if hash funcⁿ follow random oracle model then
it must be collision resistant.

How we prove?
⇒ Using stastical test we can verify that hash function is
collision resistant.

18/11/2024

Random Oracle Model:



① Repeatability (same i/p → same o/p)
2. Independence : if two i/p highly co-related but o/p are non-corelated
③ Non-corelatibity.

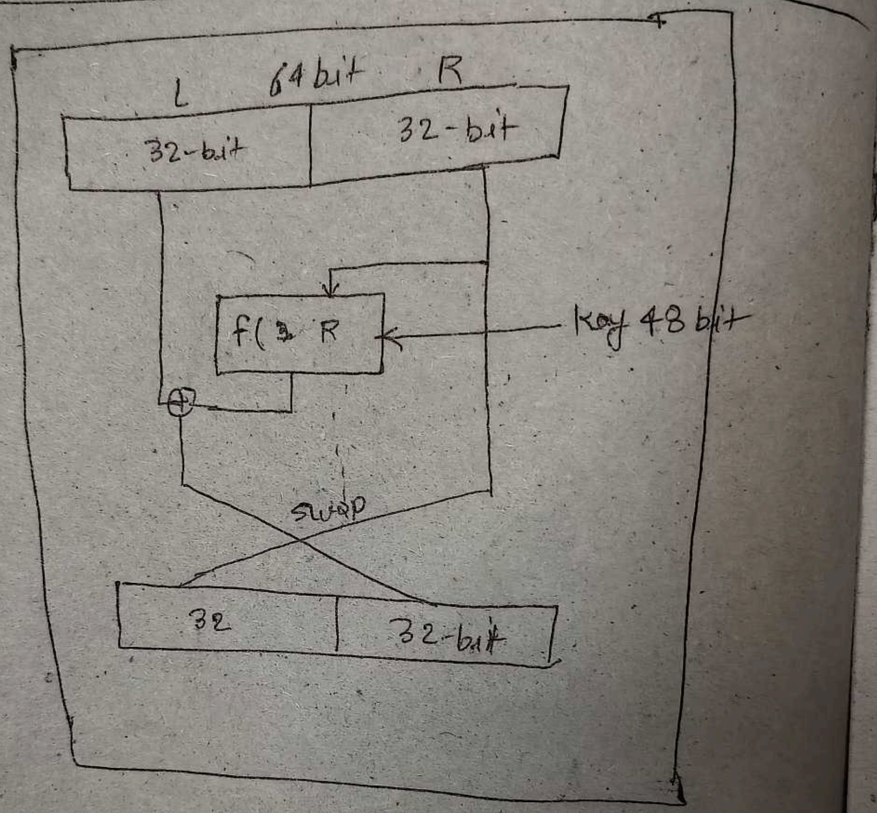
(x_1, y_1) (x_2, y_2) - - - - (x_n, y_n)
⇒ $f(x_{n+1}) = ?$

* ROM designed for hash function.

$$s = \{(m' - \underset{\uparrow}{r'}\underset{\uparrow}{\sigma_2'})\sigma_1^{-1}\} \bmod (p-1)$$

s is not known to attacker

DES:



acity ✓

Gamal + Elleptic curve cryptography.

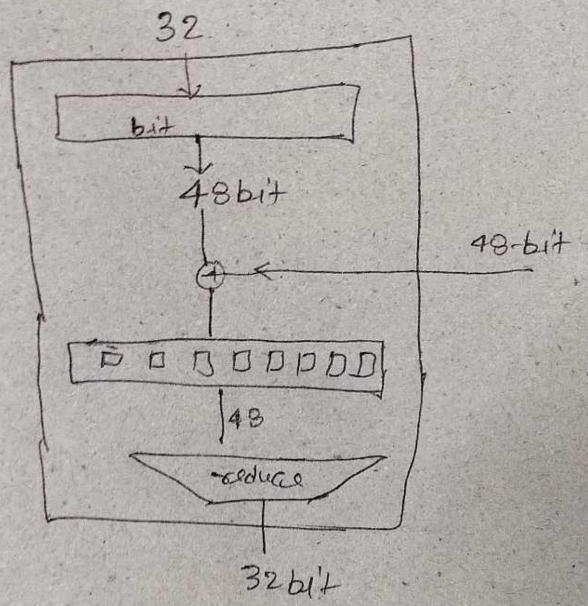
Digital Signature. ✓

xtended Euclid algo. ✓

9) Formatis Theorem. ✓

10) Euler's theorem (both version) ✓

1) Structure of DES & AES


32
bit
48bit
48-bit
48
reduce
32bit