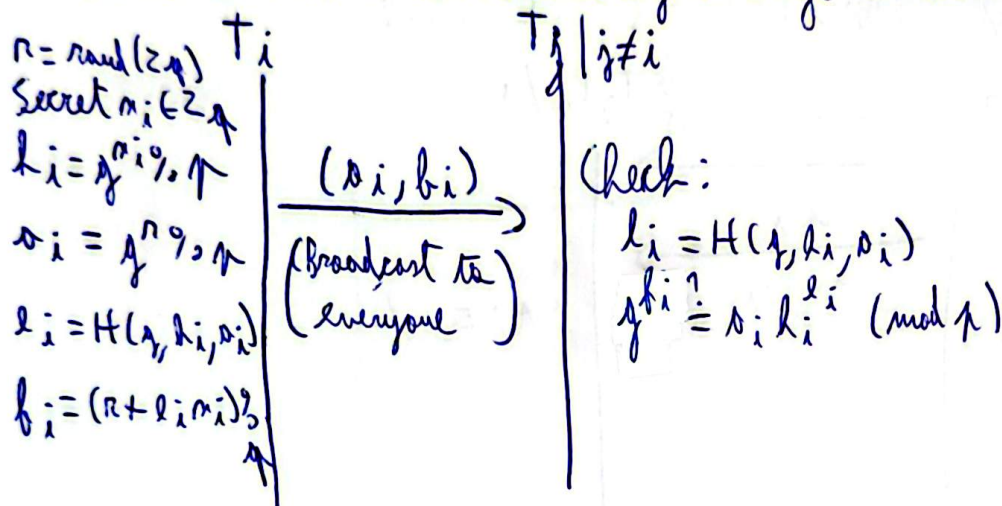
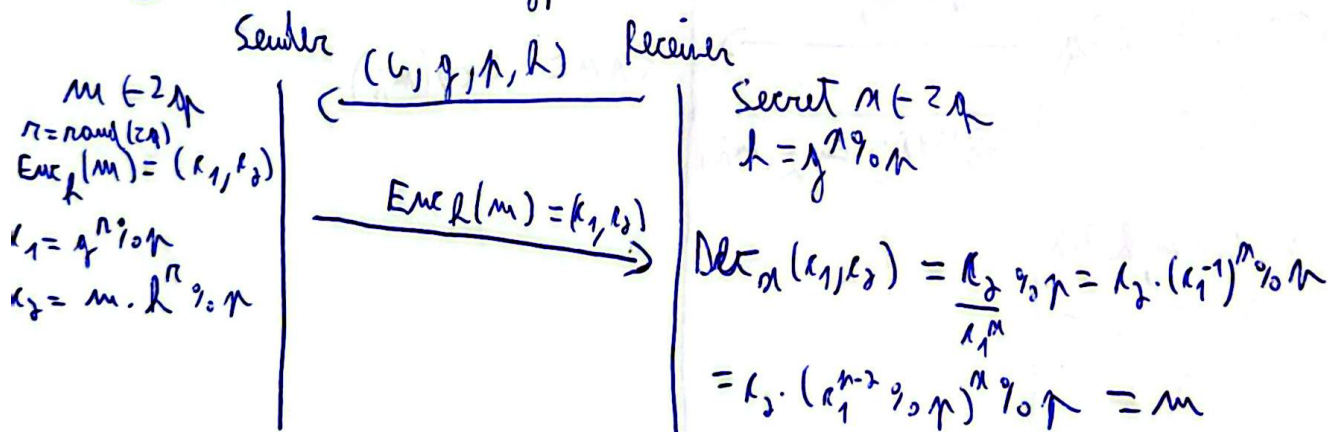


TP3: (Q1)

1) To prove knowledge of a secret key we can simply use a Schnorr protocol but we have to make it non-interactive in order to not be overloaded by a huge amount of messages.



2) El Gamal Encryption:



There is no proof that sender is a Trustee. ~~Trustee~~

Each Trustee has to prove to an other that he knows the secret associated to one of the public keys of the list list when broadcasting a message or anyone can enter the group.

A Trustee can cheat. If one compute $u_2 = h_2 = g^{x_2} = g^{u_1} / \prod_{i=1}^{n-1} g^{x_i} \pmod{p}$

When the general result $\prod_{i=1}^n g^{x_i} = \prod_{i=1}^{n-1} g^{x_i} \cdot g^{x_n} = g^y$ where y is a controlled value by the malicious trustee

TP.3.1

Q) ~~Using Algorithm~~ SKC for EL control

Secret $m_i \in \mathbb{Z}_q$
 $h_i = g^{m_i} \pmod{p}$
 $H = \prod_{i=1}^n h_i \pmod{p}$
 Shared Public Key
 Now classic ElGamal
 with it:

t_i $\xrightarrow{\text{Agreement}(G, g, p)}$ $T_i | i \neq i$
 $\xrightarrow{h_i}$
 $\xleftarrow{\forall i | i \neq i: h_i}$ SAME: h

$m_i \in \mathbb{Z}_q$
 $\text{Enc}_L(m_i) = (r_1, r_2)$
 $n = \text{rand}(\mathbb{Z}_q)$
 $r_1 = r^{n/q} \pmod{q}$; $r_2 = m_i \cdot h^{n/q} \pmod{q}$
 Now the black Trustee has to collaborate to generate a shared private key.
 $k_i = r_1^{x_i} \pmod{q}$

$$\begin{array}{c} \text{Agree on } (G, g, \uparrow) \\ \hline h_i \\ \hline \text{Disagree: } h_j \end{array} \quad \Bigg| \quad \begin{array}{c} T_i | i \neq i \\ \text{SAME: } h \end{array}$$

Engl (m_i) →

$$\frac{d_i}{\leftarrow \rightarrow \forall j \neq i: d_j}$$

* n is unknown in clear!!

$$A = \prod_k A_k \pmod n = \prod_k x_1^{m_k} \pmod n$$

*

$$(\Rightarrow) A = x_1^{\sum_k m_k} = x_1^n \mid n = \sum_i m_i \pmod n$$
$$\text{Per}_d(r_2) = r_2 \cdot d^{-1} \% n = m_i$$

Q2) Voting

TP.3.2

Exponential Interactive ElGamal:

Voter _i	Ballot
$r_i = \text{rand}(2q)$ $\text{note}_i \in \{0, 1\}$ $m_i = g^{\text{note}_i} \mod p$ $\text{Enc}_k(m_i) = (r_1, r_2)$ $\begin{cases} r_1 = g^{r_i} \mod p \\ r_2 = m_i h^{r_i} \mod p \end{cases}$	$\text{Secret } x \in \mathbb{Z}_q$ $h = g^x \mod p$ $\alpha_i = \text{Enc}_k(m_i) = (r_1, r_2)$ $\prod_{i=1}^n \alpha_i$ <i>make a disjunctive proof to prove that α_i carry a binary note !!</i> $\alpha = \prod_i \alpha_i \mod p = (\prod_i r_{1i}, \prod_i r_{2i})$ $= (g^{\sum_i r_{1i}}, g^{\sum_i \text{note}_i} \cdot h^{\sum_i r_{1i}})$ $= (g^R, g^S \cdot h^R) \mid R = \sum_i r_{1i} \wedge S = \sum_i \text{note}_i$ <i>If S is small then (in practice it is), $\text{Dec}^k S$ is decryptable.</i> $\text{Dec}_x(g^R, g^S \cdot h^R) = \text{Dec}_x(\alpha)$ $= (g^S \cdot h^R) \cdot [(g^R)^{-1}]^x \mod p$ $= g^S \cdot (g^x)^R \cdot [(g^R)^{-1}]^x \mod p$ $= g^S \mod p$ $S = \text{discrete-log}(g, \text{Dec}_x(\alpha), h)$

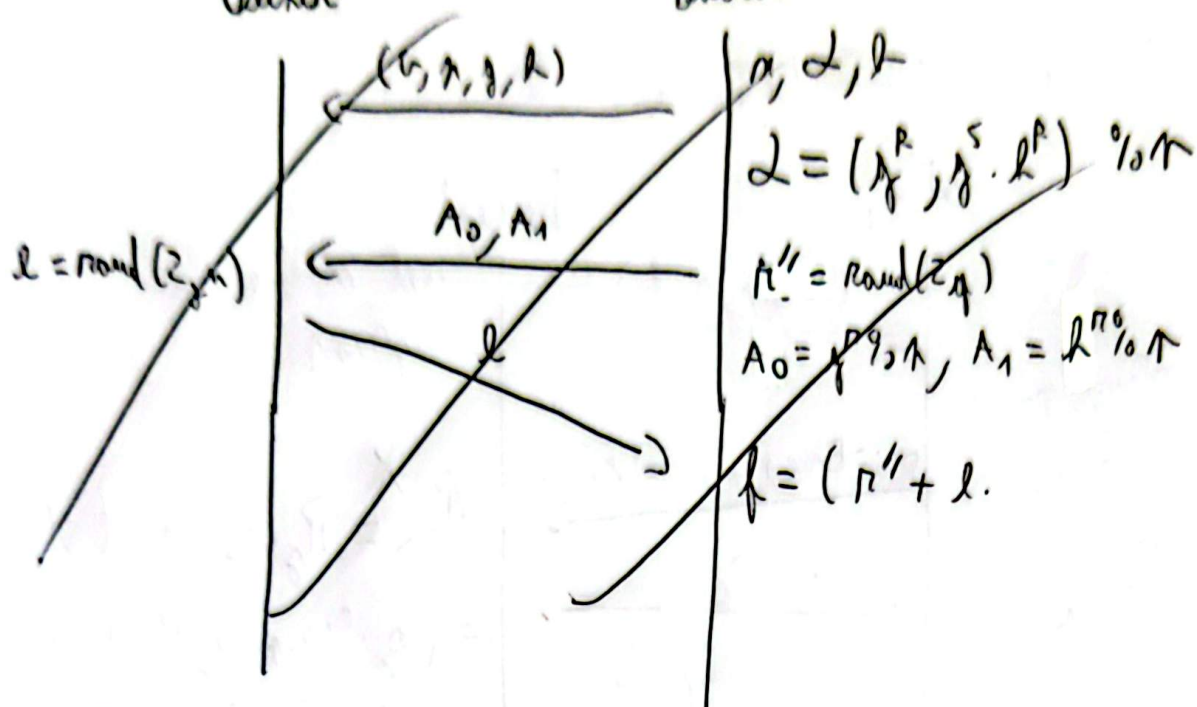
- 1) \hookrightarrow + A *Disjunctive proof* is made to ensure the note is a *binary value*.
- 1) Check if $S = 1$ for yes or $S = 0$ for no. We can *reverse remove* the discrete log and say if $g^S \mod p = g$ then yes or 1 then no.
- 2) * α is a multiplication of all encrypted notes α_i when *dec* decrypted by homomorphism gives the sum of note.
- 3) ~~We can adapt the Disjunctive OR proof to test more values !!~~ Check if $\text{Enc}_k(g^3 \mod p) = \alpha$

3)

Dealer

Ballot

TP.3.3



Q3) Exactly as above, ElGamal encryption gives homomorphic ciphertexts where the product of them give the sum of votes once decrypted.

Let the encrypted notes α_i such that

$$\alpha_i = (g^{r_i}, g^{r_i} \cdot h^{n_i}) \pmod{p} \quad \left(\text{we use a disjunctive proof to ensure that } \alpha_i \text{ encodes a binary note.} \right)$$

$$\Rightarrow \prod_i \alpha_i \pmod{p} = \prod_i (g^{r_i}, g^{r_i} \cdot h^{n_i}) \pmod{p}$$

$$= (g^R, g^V \cdot h^R) \pmod{p} \quad | \quad R = \sum_i r_i \quad \wedge \quad V = \sum_i n_i$$

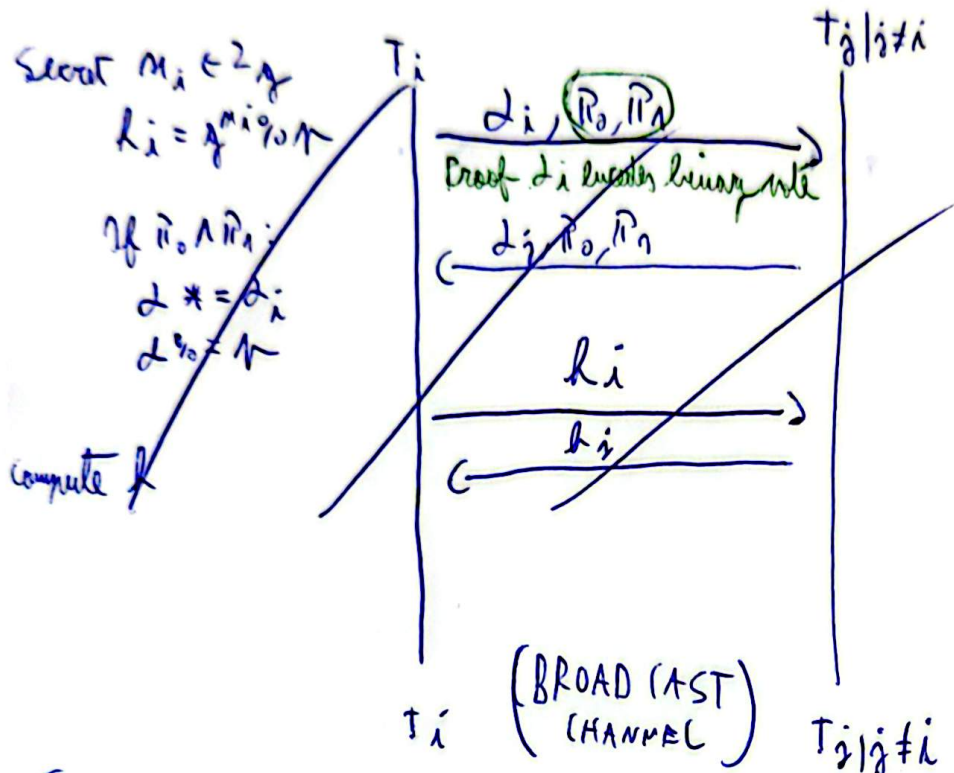
V is the sum of all notes and we can now normally decrypt $\alpha = \prod_i \alpha_i$ proceeding

as a normal ElGamal Decryption: $(g^R)^{-1}$

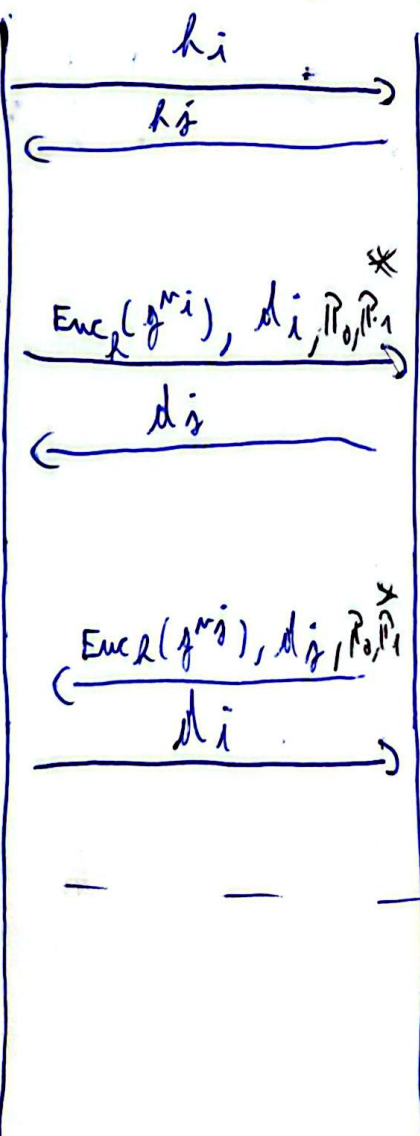
$$\text{Dec}_\alpha(\alpha) = \alpha_2 (\alpha_1^{-1}) \% p = g^V \cdot h^R \cdot [g^R]^{-1} \% p = g^V \% p$$

Then if V is small enough compute $\text{discrete-log}(p, g^V, g) = V = \sum_i n_i !!$

2) To avoid to explicitly compute the secret key ^{if one single party} we can use Distributed Key Generation for El Comsol. TP.3.4



Secret $m_i \in \mathbb{Z}_q$
 $h_i = g^{m_i} \mod p$
 compute $f = \prod_{i=1}^n h_i \mod p$
 $m_i \in \{0, 1\}$
 $Enc_R(g^{m_i}) = (r_{i1}, r_{i2}) \mod p$
 $d_i = r_{i1}^{m_i} \mod p$
 compute $d = \prod_{i=1}^n d_i \mod p$
 $D^* = d$
 $\Delta^* = Enc_R(g^{m_i})$
 $d_i = r_{i1}^{m_i} \mod p$
 $D^* = d$
 $\Delta^* = Enc_R(g^{m_i})$
 At the end $\Delta = (\Delta_1, \Delta_2)$
 $Dec_D(\Delta) = (\Delta_2 \cdot D^{-1}) \mod p$
 $= g^{f \cdot m_i} \mod p$



* Disjunctive proof
 ensuring that $Enc_R(g^{m_i})$
 encodes a binary note.

DISTINCTIVE PROOF

=> Prove an exponential FL-CRAMAL

$$g^k = a \cdot k^2 \quad \Bigg| \quad g^{k^2} = a_0 \cdot c_0^2 \cdot (c_1 \cdot k^m)^2 \pmod{N}$$

10

$$N = \text{mod} \in \{0, 1\}$$

P

$$\begin{cases} K = \text{Enc}_K(g^N) = (g^N, g^N \cdot k^N) \pmod{N} \\ K = (k_0, k_1) \quad | \quad n = \text{rand}(2q) \end{cases}$$

$$k_{1-N}, k_{1-N} = \text{rand}(2q^2)$$

$$\begin{cases} k_{1-N,0} = g^{k_{1-N}} \cdot (c_0^{-1})^{k_{1-N}} g_0^N \\ k_{1-N,1} = k_{1-N} \cdot ([k_1 \cdot (g^N)^{-1}])^{k_{1-N}} \end{cases}$$

$$r'_N = \text{rand}(2q) \quad \% \uparrow$$

$$\begin{cases} r_{N,0} = g^{r'_N} g_0^N \\ r_{N,1} = k^{r'_N} \uparrow \end{cases}$$

$$l_0 = l - l_1 (\% N) \quad \Delta \text{ if } l - l_{1-N} < 0$$

$$l'_0 = (n' + l_0 \cdot \pi) \% \uparrow \quad \text{then } l_{N,1} = l_{N,1} + (l - l_{1-N})$$

$$(c, n, g, k)$$

V



$$K) (k_N, k_{1-N})$$

l

$$(k_N, l_N, k_N), (k_{N,1}, l_{1-N,1}, k_{1-N,1})$$

$$\text{Secret } n \in \mathbb{Z}_q$$

$$k = g^{n^2} g_0^N$$

$$l = \text{rand}(\mathbb{Z}_q^m)$$

$$\text{Dec}_K: D = \text{Dec}_K(K) = g^{N^2} g_0^N$$

$$\begin{cases} l = l_N + l_{1-N} \end{cases}$$

$$g^{l_N} = r_{N,0} \cdot c_0^{l_N} (\% N)$$

$$k^{l_N} = r_{N,1} \cdot (k_1 \cdot c_0^{-1})^{l_N} (\% N)$$

$$g^{l_{1-N}} = r_{1-N,0} \cdot c_0^{l_{1-N}} (\% N)$$

$$k^{l_{1-N}} = r_{1-N,1} \cdot (k_1 \cdot c_0^{-1})^{l_{1-N}} (\% N)$$