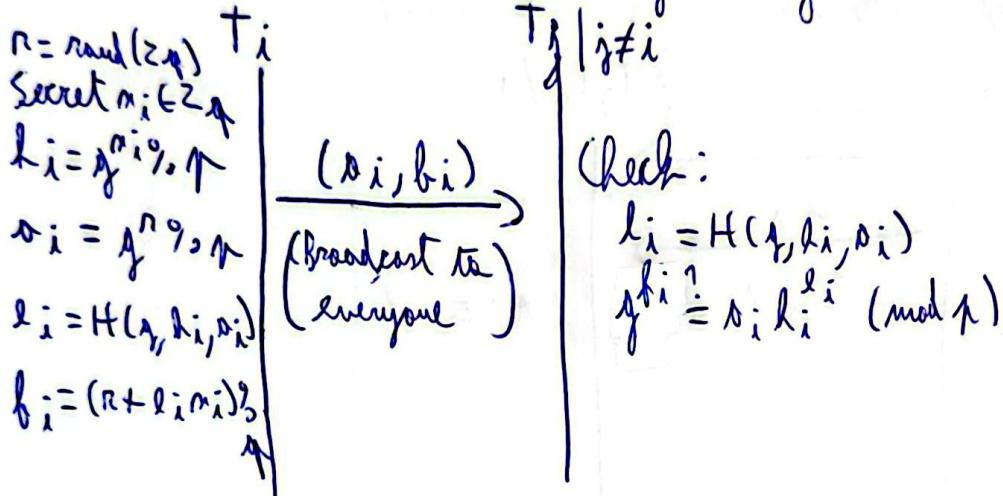


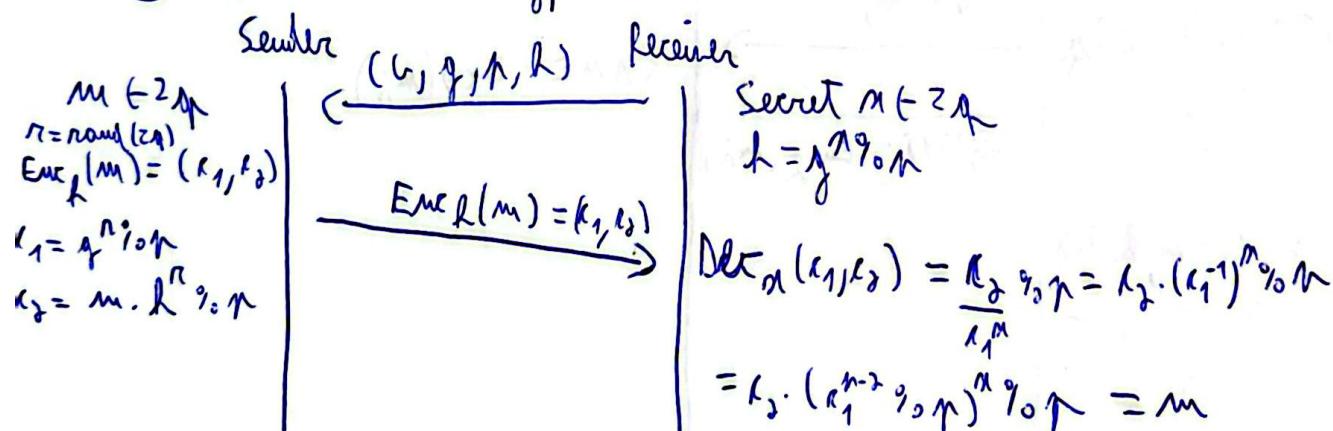
Privacy:

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) ElGamal Encryption:



There is no proof that sender is a Trustee.

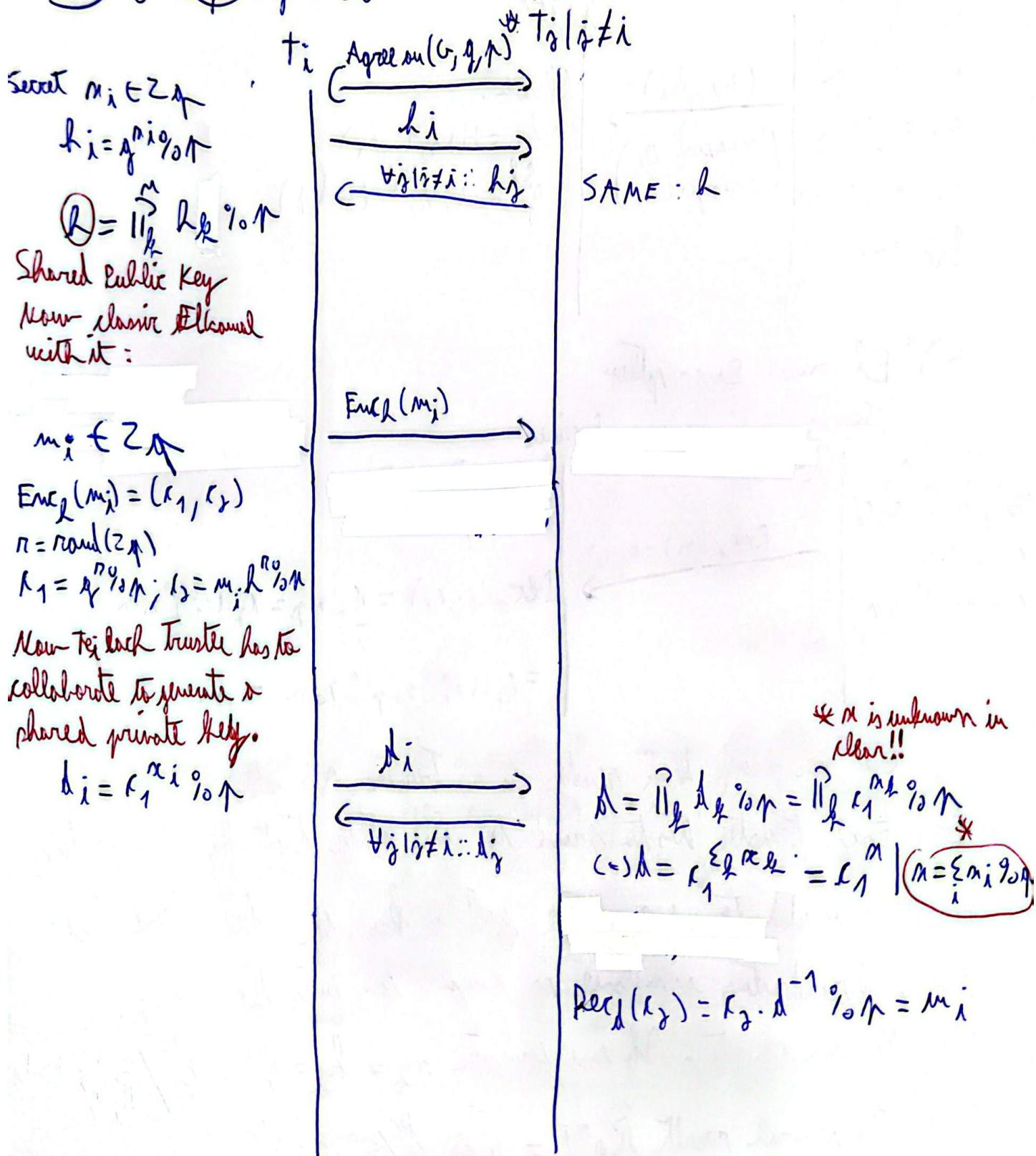
Each Trustee has to prove to another that he knows the secret associated to one of the public keys of the list when broadcasting a message or anyone can enter the group.

A Trustee can cheat. If one compute $m_2 = h_2 = g^{r_2} = g^r / \prod_{i=1}^{n-1} g^{r_i} \pmod{N}$

Then the general result $\prod_{i=1}^n g^{m_i} = \prod_{i=1}^{n-1} g^{m_i} \cdot 1 / \prod_{i=1}^{n-1} g^{m_i} = g^r$ where g is controlled and r is random.

TP.3.1
 3) Elast - Shamir how transformation removes the interactive aspect of a sigma protocol by making the challenge generation deterministic. The challenge t is computed as an hash value of the public parameters (g, p) and the commitment (α) as done in point 1.

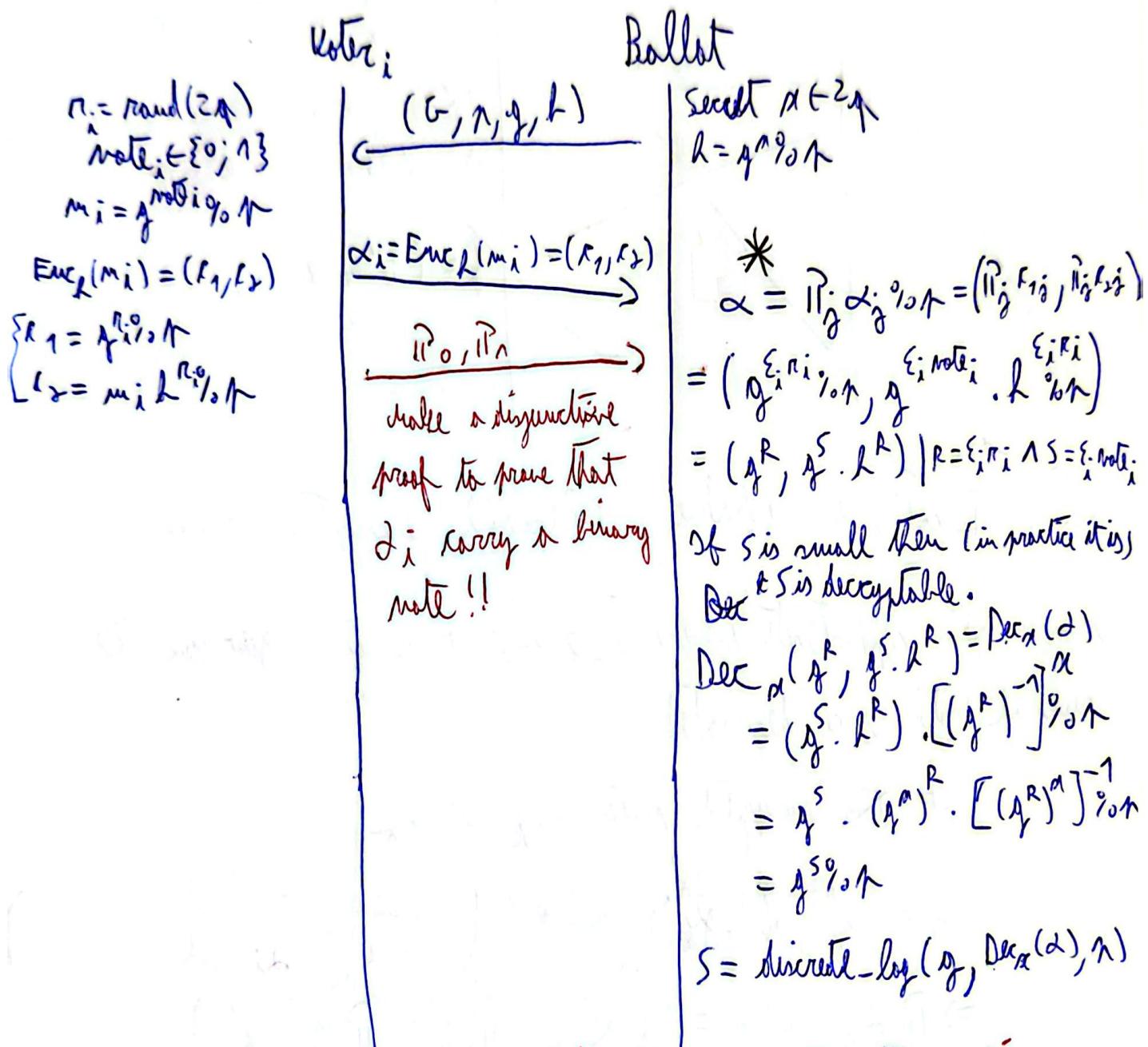
~~(*)~~) ~~Interactive Algorithm~~: SKC for EL channel



Q2) Voting

TP.3.2

Exponential Mix Interactive ElGamal:



+ A key 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 the discrete log and say if $g^S \circ \uparrow = g$ then yes or 1 then no.

2) \ast α is a multiplication of all encrypted notes α_i when decrypted by homomorphism gives the sum of note.

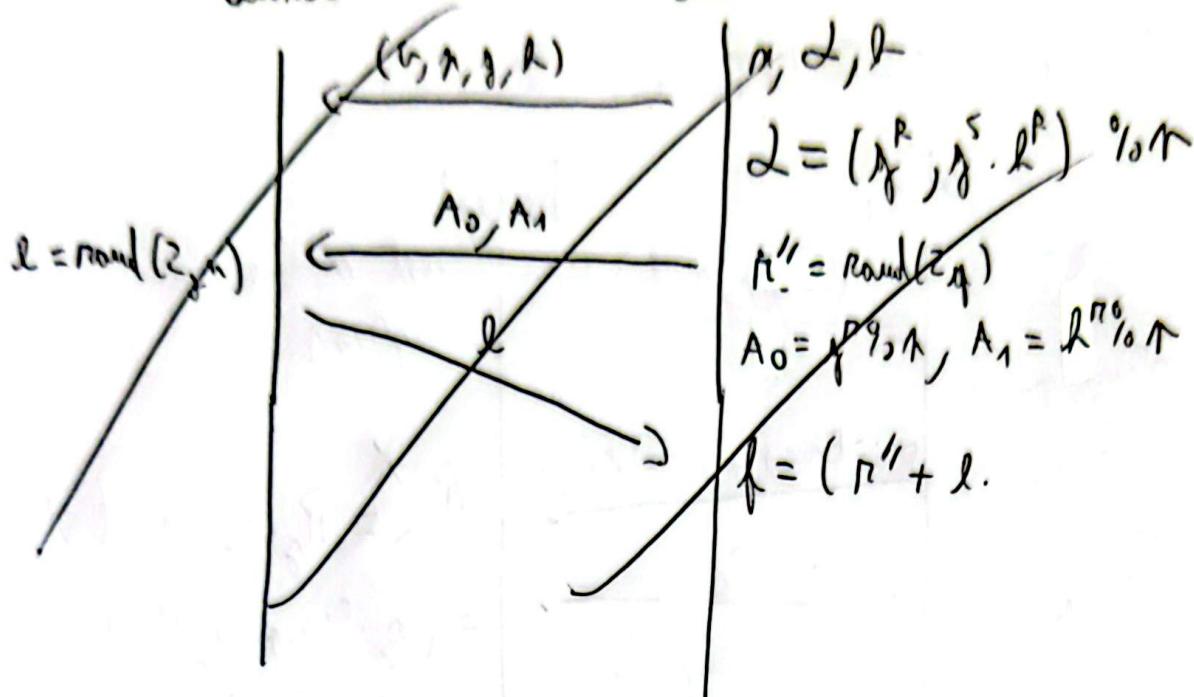
3) we can adapt the OR Disjunctive OR proof to test more notes!! (Check if $\text{Enc}_\lambda(g^3 \circ \uparrow) = \alpha$)

3)

Choker

Ballot

TP.3.3



(Q3)¹ Exactly as above, ElGamal encryption gives homomorphic ciphertexts where the product of them give the sum of votes once decrypted.

Let two encrypted notes α_i and α_j

$$\alpha_i = (g^{r_i}, g^{v_i} \cdot h^{r_i}) \pmod{p} \quad \begin{array}{l} \text{(we use a disjunctive proof to} \\ \text{ensure that } \alpha_i \text{ encodes a binary note.)} \end{array}$$

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

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

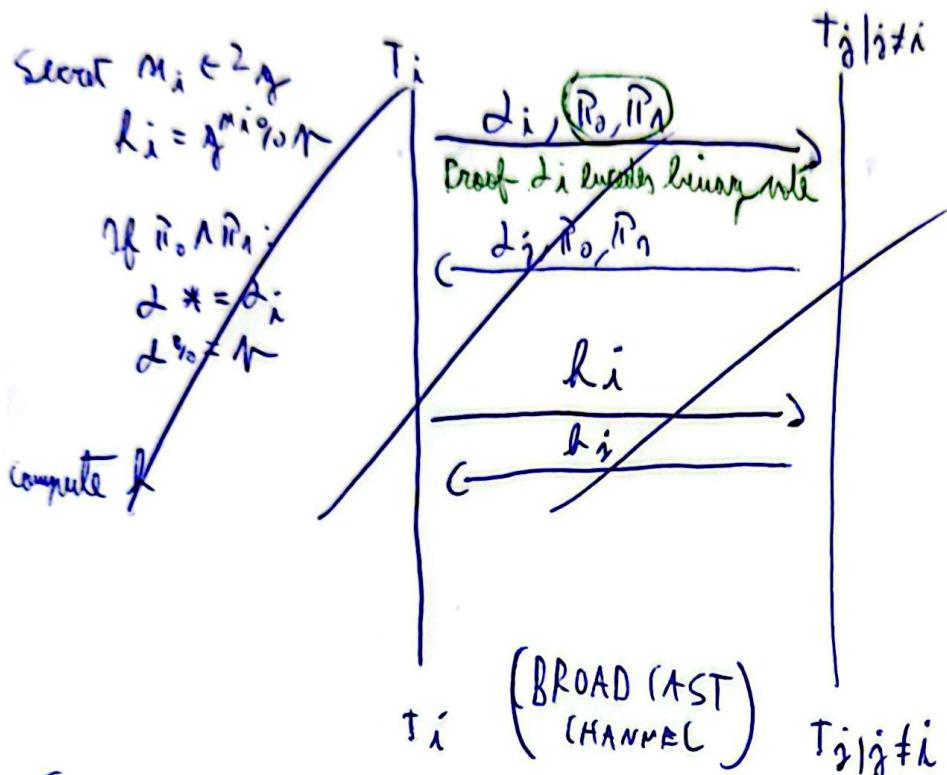
V is the sum of all votes and we can now normally decrypt $\alpha = \prod_i \alpha_i$ proceeding

as a normal ElGamal Decryption: $(g^m)^R$

$$\text{Dec}_\alpha(\alpha) \approx \alpha_2 (r_1^m)^{-1} \% p = g \cdot h^R \cdot [(g^R)^m]^{-1} \% p = g^V \% p$$

Then if V is small enough compute discrete-log $(r_1, g^V, g) = V = \sum_i r_i !!$

2) We want to explicitly compute the secret key ^{If one single party} we can TP. 3.4 use Distributed Key Generation for El Gamal.



Secret $m_i \in \mathbb{Z}_p^*$
 $h_i = g^{m_i} \% p$
Compute $\lambda = \prod_i h_i \% p$
 $m_i \in \{0, 1\}$

$$\text{Enc}_L(g^{m_i}) = (f_{i1}, f_{i2}) \% p$$

$$d_i = f_{i1} \% p$$

$$\text{compute } d = \prod_i d_i \% p$$

$$\alpha^* = d$$

$$\alpha^* = \text{Enc}_L(g^{m_i}) -$$

$$d_i = f_{j1} \% p$$

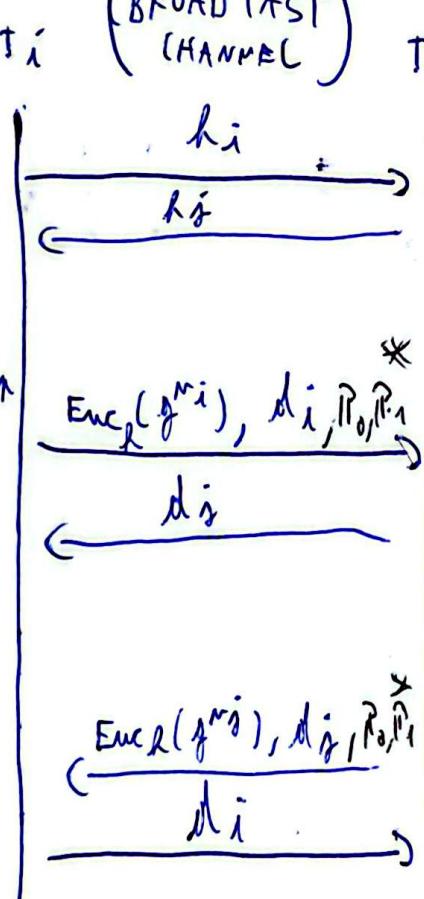
$$\alpha^* = d$$

$$\alpha^* = \text{Enc}_L(g^{m_j})$$

$$\text{At the end } \alpha = (_, _)$$

$$\text{Dec}_D(\alpha_2) = (\alpha_2 \cdot D^{-1}) \% p$$

$$= g^{e_i m_i \% p}$$



* Disjunctive proof
ensuring that $\text{Enc}_L(g^{m_i})$
encodes a binary note.

Distinctive Proof

\Rightarrow Break the Exponential ELGamal encryption a random binary root.

$$g^R = gR \quad | \quad \begin{cases} R = r_0 \cdot r_1^{-1} \\ g^R = g_{r_0} \cdot r_0^{-1} \end{cases} \quad (\text{mod } p) \\ R^k = r_1 \cdot (r_1 \cdot r_1^{-1})^k \quad (\text{mod } p)$$

70

Secret msg
 $r = g^a \cdot h$

P

(G, N, f, h)

V

$$\begin{cases} n = \text{rand} \in \{0; 1\} \\ k = \text{Enc}_R(A_R^n) = (g_{r_0}^n, g_{r_1}^n \cdot h^n) \quad (\text{mod } p) \\ \begin{cases} r = (r_0, r_1) \end{cases} \end{cases}$$

$$x_{1-N} \cdot f_{1-N} = \text{rand}(Z_A^{2^2})$$

$$\begin{cases} x_{1-N, 0} = g_{r_1-N}^{k_{1-N}} \cdot (r_0^{-1})^{x_{1-N}} \cdot g_{r_0}^N \\ x_{1-N, 1} = R_{1-N}^{k_{1-N}} \cdot [(r_1 \cdot (A_R^{N})^{-1})^{f_{1-N}}] \end{cases}$$

$$r'_R = \text{rand}(Z_A)$$

$$\begin{cases} x_{N, 0} = g_{r_1-N}^{k_{N, 0}} \cdot h \\ x_{N, 1} = R_{N, 0}^{k_{N, 1}} \cdot h \end{cases}$$

$$\begin{cases} (x_{N, 0}, x_{N, 1}, R_N), (x_{1-N, 0}, x_{1-N, 1}, R_{1-N}) \end{cases}$$

Check: $n = \text{Dec}_R(r) \Leftrightarrow A_R^{N, 0} \cdot h$

$$x \stackrel{?}{=} x_n + x_{1-N}$$

$$\begin{cases} g_{r_N}^{k_N} = h_{r_0} \cdot r_0^{k_N} \cdot (g_{r_1} \cdot h)^N \\ R_N = x_{N, 1} \cdot (r_1 \cdot r_1^{-1})^N \cdot (g_{r_0} \cdot h) \end{cases}$$

$$\begin{cases} g_{r_{1-N}}^{k_{1-N}} = x_{1-N, 0} \cdot r_0^{k_{1-N}} \cdot (g_{r_1} \cdot h)^N \\ R_{1-N} = x_{1-N, 1} \cdot (r_1 \cdot r_1^{-1})^N \cdot (g_{r_0} \cdot h) \end{cases}$$