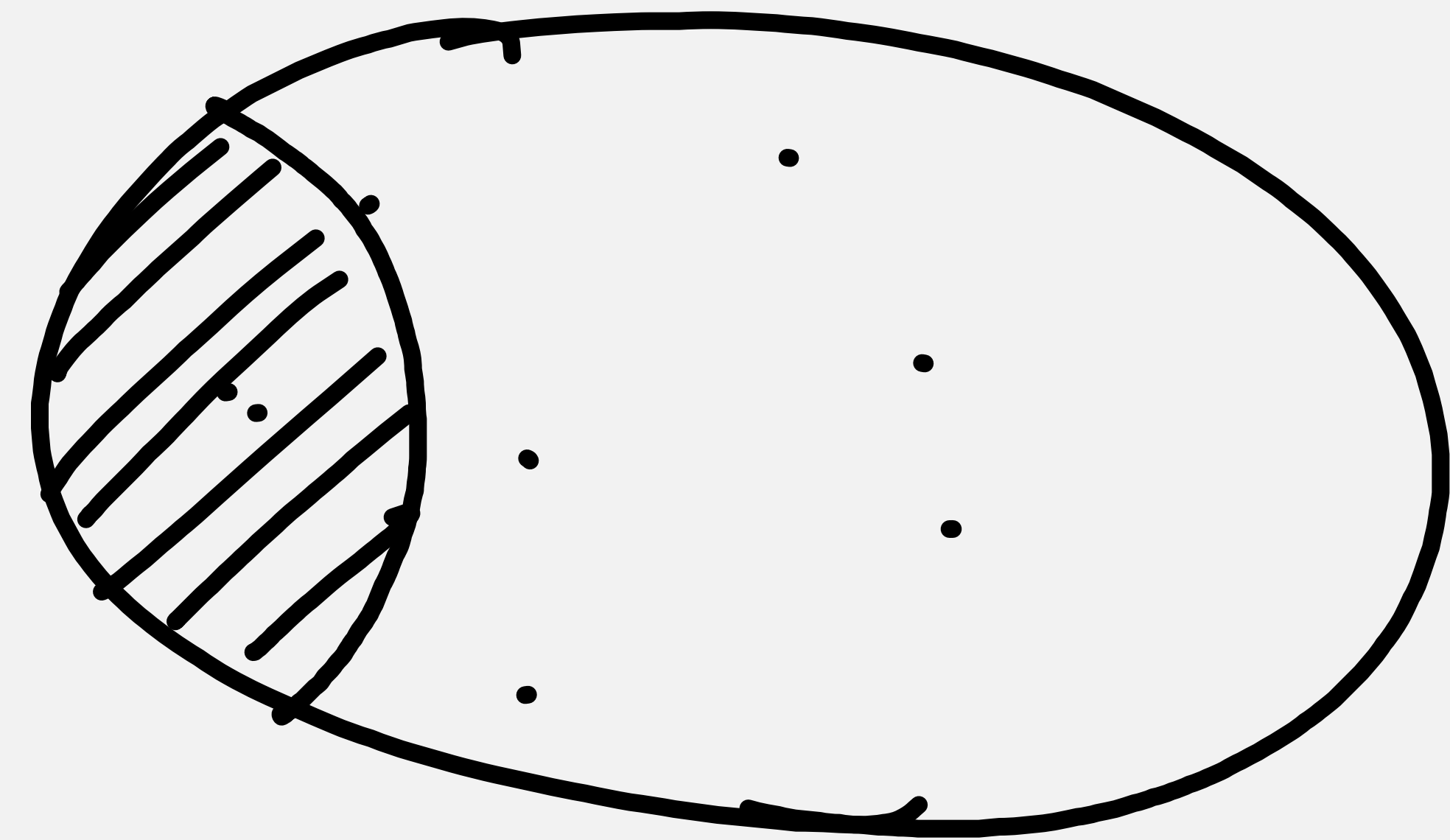
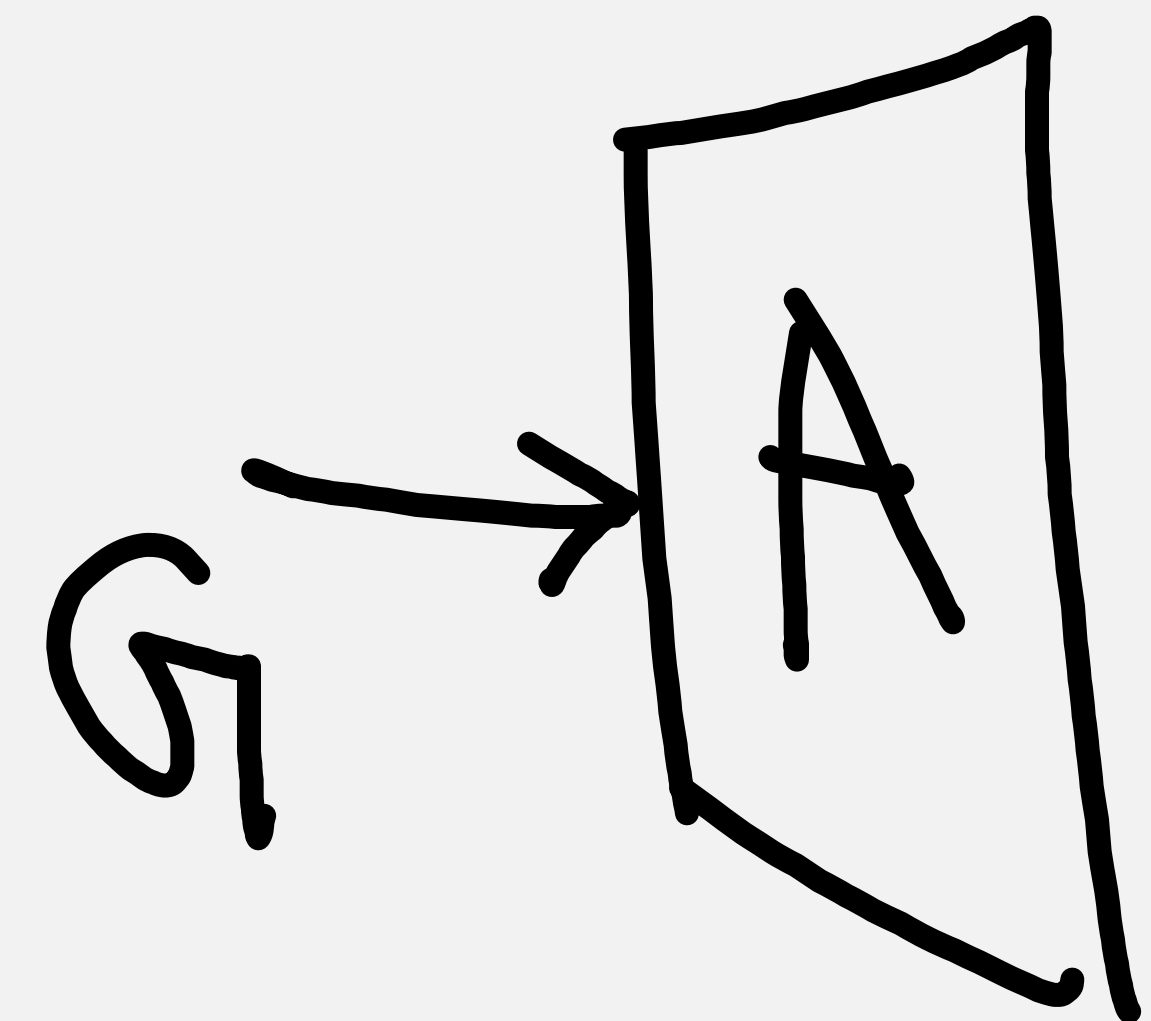


# Random Self-Reducibility:-



$HAM = \{ G : G \text{ has a hamiltonian cycle} \}$



Suppose  $A$  is an algorithm that solves  $D \log$  for a randomly chosen  $u \leftarrow \$G$  with prob at least  $\epsilon$ .

We want to construct an algorithm  $B$  that can solve  $D \log$  for any  $u$  with prob at least  $\epsilon$ .

$B(u)$ ,  $u$  is an arbitrary element of  $G$ .  $G$  is a cyclic group of order  $q$ , and  $\langle g \rangle = G$ .

1.  $\sigma \leftarrow \$\mathbb{Z}_q$
2.  $u_1 \leftarrow u \cdot g^\sigma$  →  $u_1$  is random
3.  $\alpha_1 \leftarrow A(u_1)$
4. if  $g^{\alpha_1} \neq u_1$ , return  $\perp$   
 else return  $(\alpha_1 - \sigma) \bmod q$ .

$$\begin{aligned} P[B \text{ succeeds}] \\ = P[A \text{ succeeds}] \geq \epsilon \end{aligned}$$

$$\# \text{ of trials} := \left( n \times \frac{1}{\epsilon} \right).$$

$$1 - (1 - \epsilon)^{n/\epsilon}$$

Claim:  $u_1 \leftarrow \$G_2$ . (Homework).

$$P_\delta[u_1 = u] = \frac{1}{|G_2|}, \text{ where } u \in G_2.$$

Security definition of collision-resistant hash fn:

$$\mathcal{H} = \{ h_i : \{0,1\}^* \rightarrow \{0,1\}^n \}$$

$$\begin{array}{ccc} \text{Chall} & & \text{A} \\ \hline h \leftarrow \$ \mathcal{H} & \longrightarrow & A(h) \end{array}$$

$\xleftarrow{x, x'}$   
if  $h(x) = h(x') \wedge$   
 $x \neq x', \text{ return } 1.$

## Collision resistant hash f<sub>H</sub> from RSA assumption.

Let  $e$  be a prime number;  $n$  is the modulus.

and  $y \leftarrow \$ \mathbb{Z}_n^*$ . Let  $I_e = \{0, 1, \dots, e-1\}$ .  $(n, e, y)$  is the key.

$H_{n,e,y} : \mathbb{Z}_n^* \times I_e \rightarrow \mathbb{Z}_n^*$ , defined as follows:

$$H_{n,e,y}(a, b) := a^e y^b \bmod n.$$

We want to prove that if RSA assumption is hard, then  $H$  is collision resistant.

Recall the RSA assumption:

RSA' assumption:

Chall

$$y \leftarrow \$_{Z_n^*}$$

$$\begin{array}{c} \xrightarrow{y} \\ \xleftarrow{x} \end{array}$$

wins if  $x^e = y \pmod n$ .

$\text{RSA}'_{\text{adv}}(A, e) := \Pr[A \text{ wins the above game}]$

$$\begin{aligned} \text{RSA}'_{\text{adv}}(B, e) &= \text{RSA}'_{\text{adv}}(A, e) \cdot \Pr[y \in Z_n^*] \\ &\quad + 1 \cdot \Pr[y \notin Z_n^*] \\ &\geq \text{RSA}'_{\text{adv}}(A, e) \underbrace{[\Pr[y \in Z_n^*] + \Pr[y \notin Z_n^*]]}_{=1} \end{aligned}$$

Thm:

Let  $e > 2$  be an odd integer. For every adversary  $A$ ,  $\exists$  an adversary  $B$  such that  $\text{RSA}'_{\text{adv}}(A, e) \leq \text{RSA}_{\text{adv}}(B, e)$ .

Let  $w$  be the event that  $B$  wins.

$$W = (W \wedge y \in Z_n^*) \vee (W \wedge y \notin Z_n^*)$$

$$\Pr[W] = \Pr[W \wedge y \in Z_n^*] + \Pr[W \wedge y \notin Z_n^*]$$

$$\begin{aligned} &= \underbrace{\Pr[W | y \in Z_n^*]}_{=1} \cdot \Pr[y \in Z_n^*] + \underbrace{\Pr[W | y \notin Z_n^*]}_{=1} \cdot \Pr[y \notin Z_n^*] \\ &= 1 \end{aligned}$$

## Shamir's Trick :-

Given  $y \in \mathbb{Z}_n^*$  along with an integer  $f$  that is relatively prime to  $e$ , and  $w$  such that  $w^e = y^f$ , we can easily compute  $x$  such that

$$x^e = y.$$

Example: let  $f=2$ ,  $e=3$  and  $w$  such that  $w^3 = y^2$

$$x = y/w$$

Lemma: There is an efficient algorithm that takes as input  $n, e, y, f, w$  such that  $\gcd(e, f) = 1$ ,  $w, y \in \mathbb{Z}_n^*$  such that  $w^e = y^f$ , and outputs  $x$  such that

$$x^e = y.$$

pf: Since  $\gcd(e, f) = 1$   
 $\exists s, t$  such that  $es + ft = 1$ .

$$w^e = y^f.$$

$$w^{et} = y^{ft}$$

$$y^{es} \cdot w^{et} = y^{es+ft}$$

$$\Rightarrow \underbrace{(y^s \cdot w^t)}_x^e = y$$

Proof of CR based on RSA.

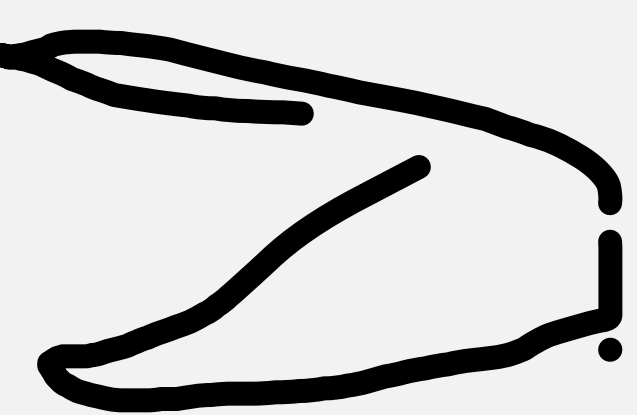
Recall  $H_{n,e,y}(a,b) = a^e y^b$ .

$(a,b), (a',b')$  such that  $H_{n,e,y}(a,b) = H_{n,e,y}(a',b')$ .

$$a^e y^b = (a')^e (y)^{b'}$$

$$\Rightarrow \left(\frac{a}{a'}\right)^e = y^{b'-b}$$

$$\Rightarrow \text{let } f = (b'-b), w = \left(\frac{a}{a'}\right)$$



# Diffie-Hellman key-exchange:

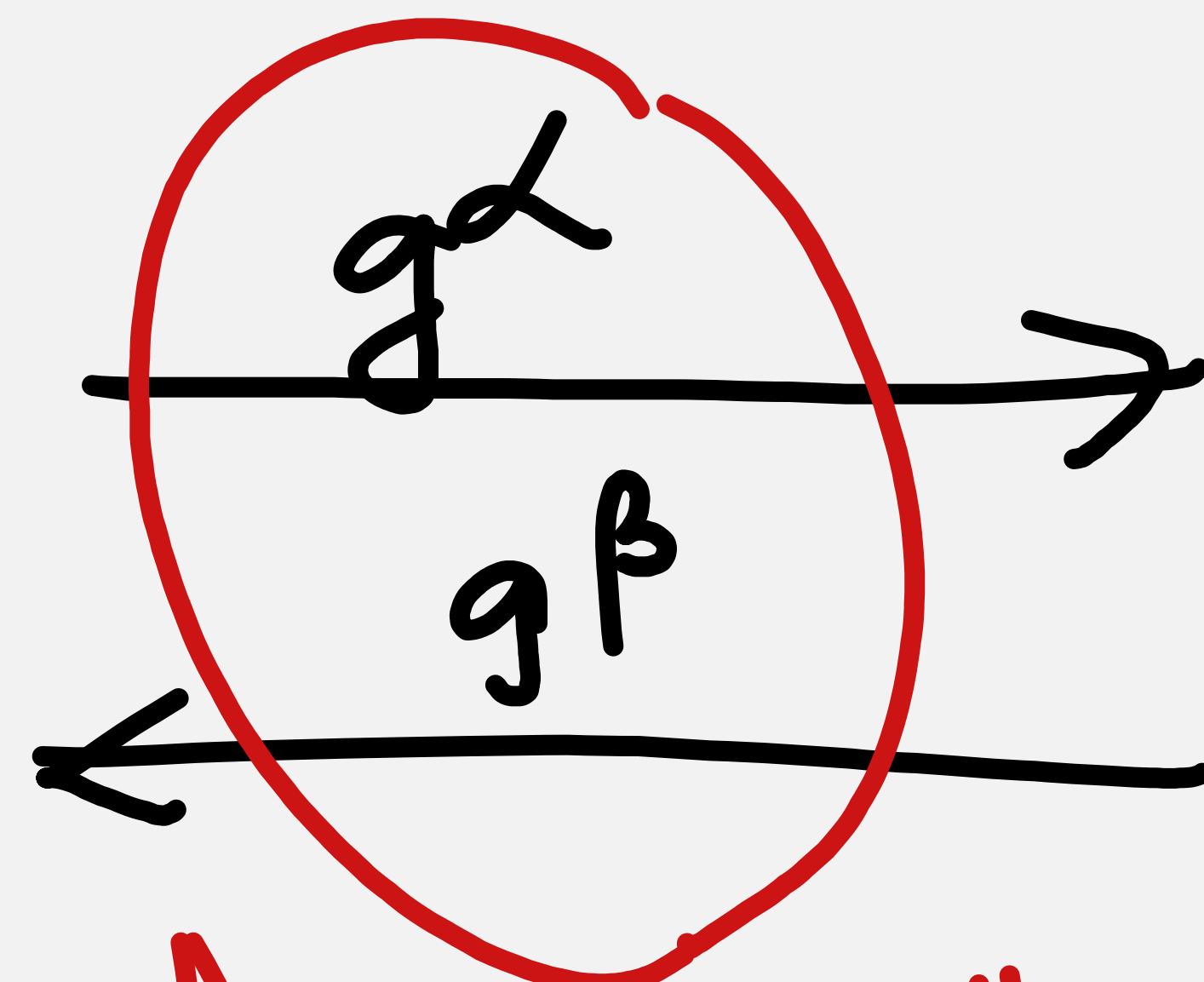
- Alice will compute a fn.  $E$ .
- Bob will compute a fn.  $E$ .
- Both of them will jointly compute a fn.  $F$ .

key-exchange protocol:  $(G, q, g)$

1. Alice sample  $\alpha \leftarrow \$ \mathbb{Z}_q$  and compute  $E(\alpha) := g^\alpha$  and sends  $E(\alpha)$  to Bob.  
over an insecure channel.
2. Bob sample  $\beta \leftarrow \$ \mathbb{Z}_q$  and compute  $E(\beta) := g^\beta$  and sends  $E(\beta)$  to Alice.  
over an insecure channel.
3. Alice computes  $F(\alpha, \beta) := (E(\beta))^\alpha := g^{\alpha\beta}$ .
4. Bob computes  $F(\alpha, \beta) := (E(\alpha))^\beta := g^{\alpha\beta}$  }  $\rightarrow$  shared secret key.

# Security Requirement.

Alice  
 $\alpha \leftarrow \mathbb{Z}_q^*$



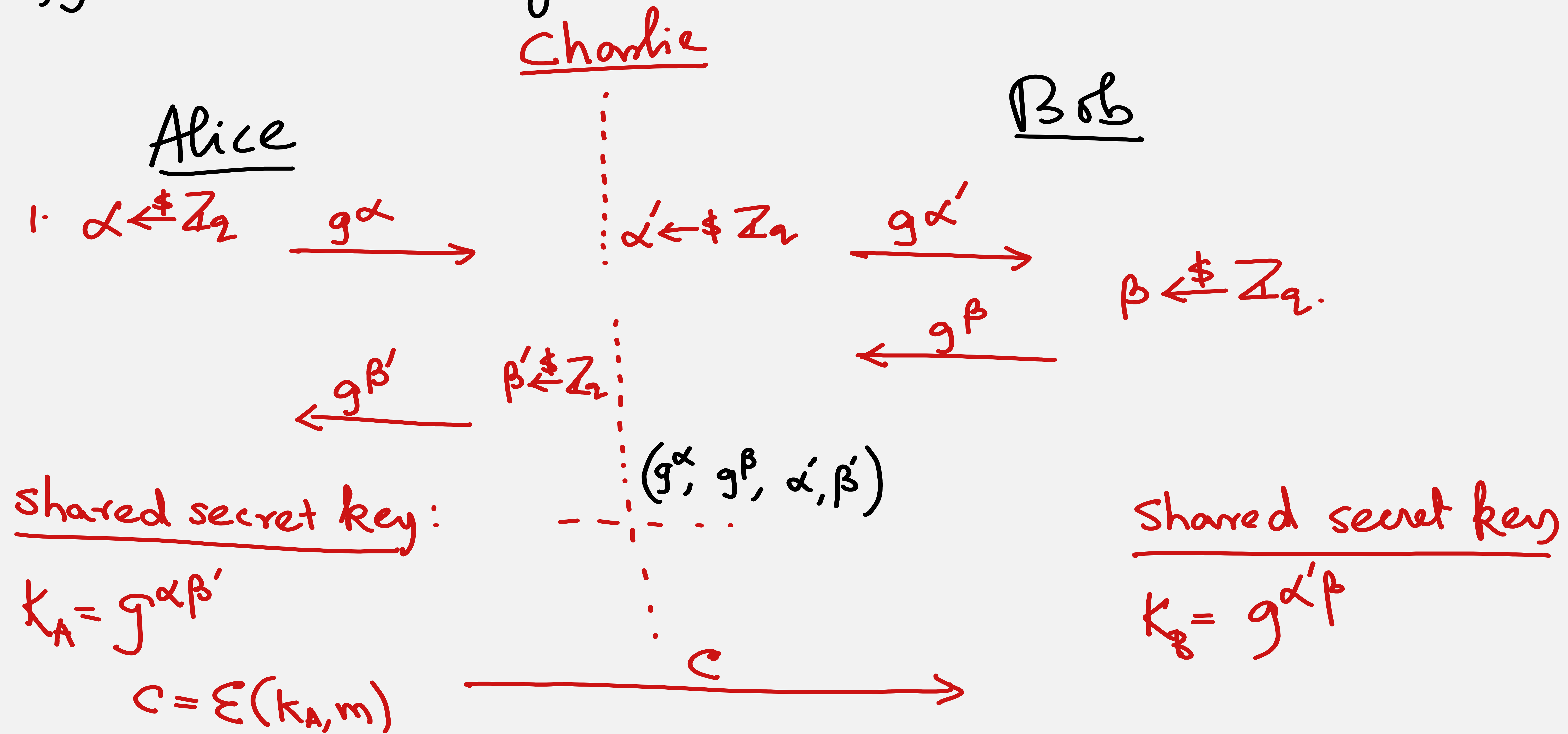
"passive".

Bob

$\beta \leftarrow \mathbb{Z}_q^*$

The goal of the adversary is to compute the shared secret key  $g^{\alpha\beta}$  from  $(g^\alpha, g^\beta) \rightarrow$  computational Diffie-Hellman.

Diffie-Hellman key-exchange is not secure against active adversaries.



Remedy:- Authenticated Key-exchange protocol.

Exercise:

Try to think designing a key exchange protocol based on RSA  
assumption. which should be  
secure against "passive adversaries".