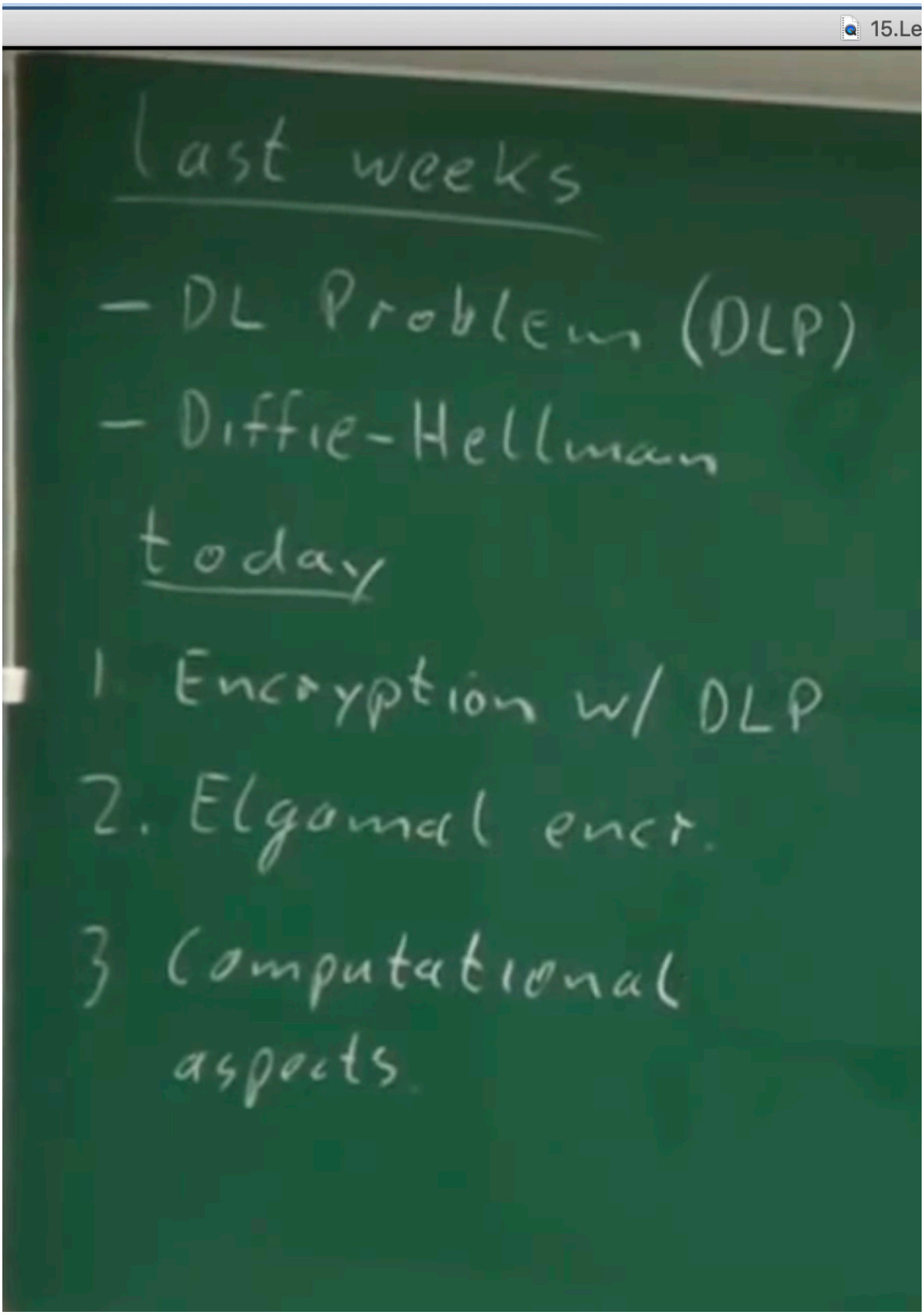
Lecture 15 Elgamal Encryption Scheme

Till now we have used DL problem only for Diffie Hellman Key exchange but as professor paar mentioned in previous lectures that we can do many things with DL problem

Today we are going to do Encryption using DL problem, ie how to encryption data.

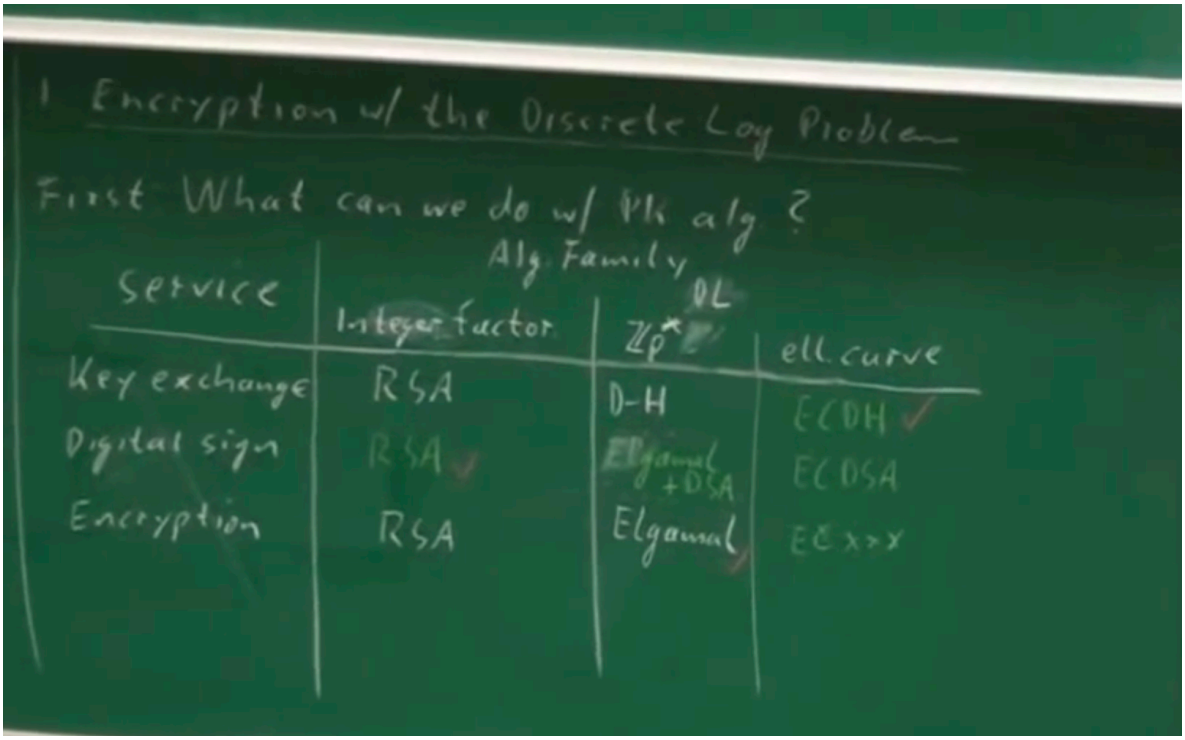
Agenda for today:



1. **Encryption with DLP**
2. **Elgamal Encryption**
3. **Computational issues**

Encryption with the Discrete Logarithm problem:
First what we can do with Public Key Cryptography:

Service	Algorithm family		
	Integer Factorization	DL Zp*	DL Elliptic Curve
Key Exchange	RSA	D-H	ECDH
Digital Signature	RSA	Elgamal Digital Signature Algorithm	ECDSA
Encryption	RSA	Elgamal	EC***



There are 4 or other types of public key algorithms, but people are not using them as key sizes are very huge in MB's and these algorithms are slow and hence not used in practise.

Develop an Encryption scheme from D-H key exchange:
Previously we through that we will use Diffie-Hallman Key exchange to generate the key and use AES to encrypt using that key but now we want to use Native D-H for Encryption too so here is Elgamal encryption.

Alice alpha and p are domain parameters Bob

A = alpha^a mod p
a is private key of Alice

B = alpha^b mod p
b is private key for Bob

B^a mod p = KAB A^b mod p = KAB

Above is the plain old diffie-hellman

So question is what we can do to encrypt the data now ? XOR, yes we can do but Key size should be same as Message, what else we can do ? we can multiply too

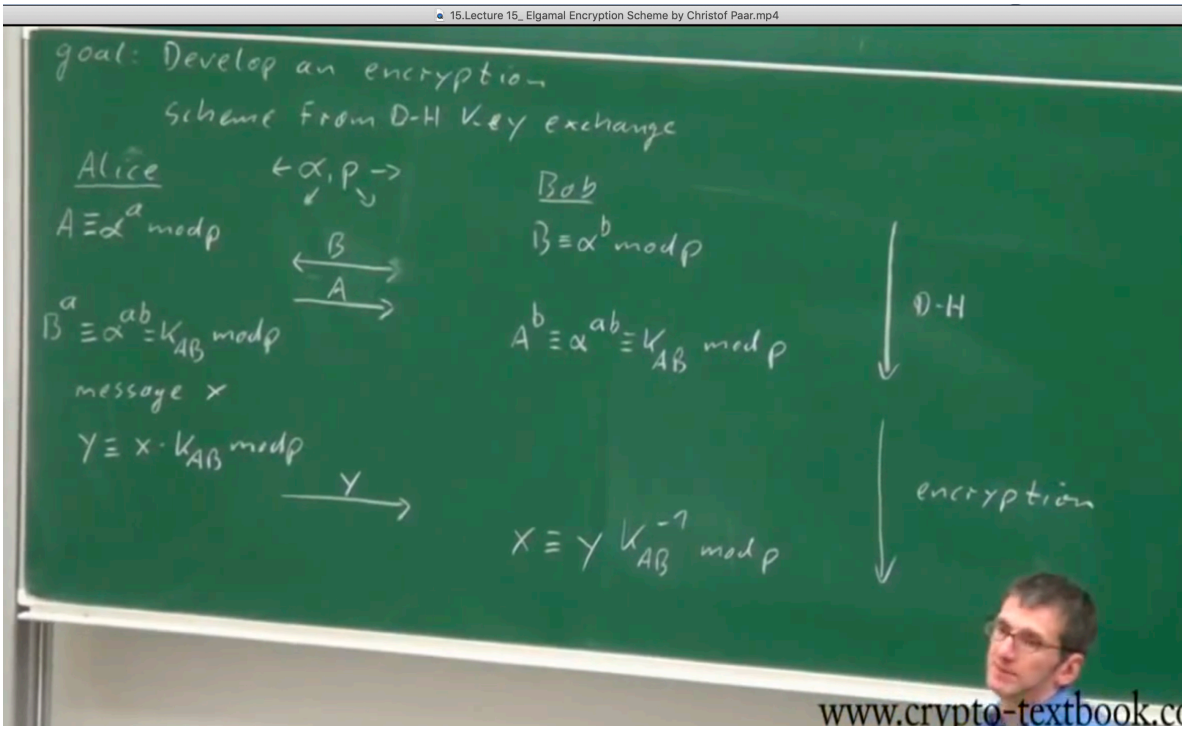
Message * KAB
say message is X
———————————————————————————————————

Y = X.KAB mod p

What does bob needs to do ? divide correct actually multiply with the inverse.

so bob does is X = Y. KAB^-1 mod p

a small important doubt which came into my mind is how we are assured that KAB will have inverse ? but then what is the condition of finding inverse that both the numbers should have GCD as 1 so KAB and p should have GCD as one and as we know p is prime and KAB is mod p value so KAB can have a value as {1,2,3,4,...., p-1} and these all numbers are coprime to p **hence we can say that GCD is 1 so inverse will always be there.**



Elgamal encryption is same like this but when you look into books you find them very different but they are not. Elgamal is identical to this.

in Books variable names are changed and there is some reordering of parameter passing as **this makes it more efficient.**

Elgamal Encryption —>
it is a variant of Diffie-Hellman.
There are some differences, in D-H we are using domain parameters which are known to everyone but in Elgamal Bob has to compute alpha and P and while sending public key, bob has to send the domain parameters.

Remarks:
Elgamal encryption was invented around 1985., it is very similar to Diffie Hellman but with Reordering of steps. it is also spelled as ELGamal.

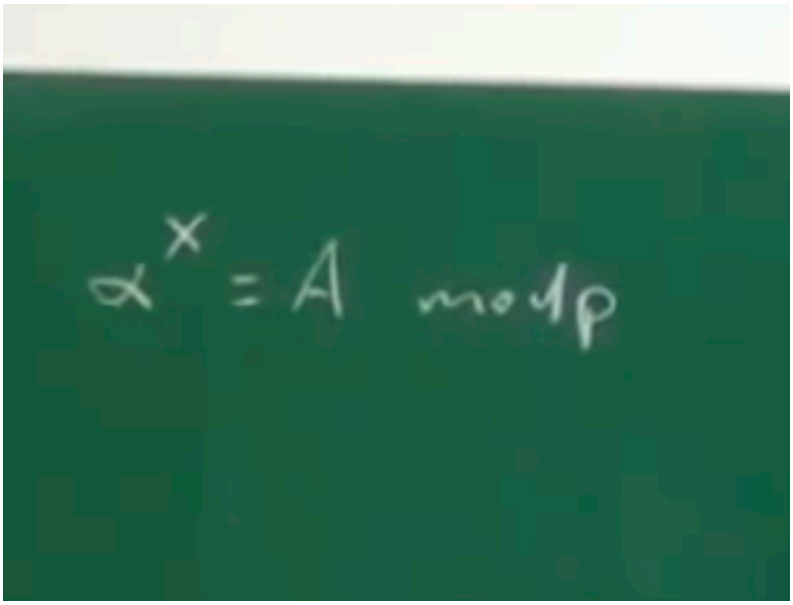
Alice Bob
_____

If we look at the Public key encryption, we had the analogy that
if you want to send a letter, in old days alice and bob both should know the key

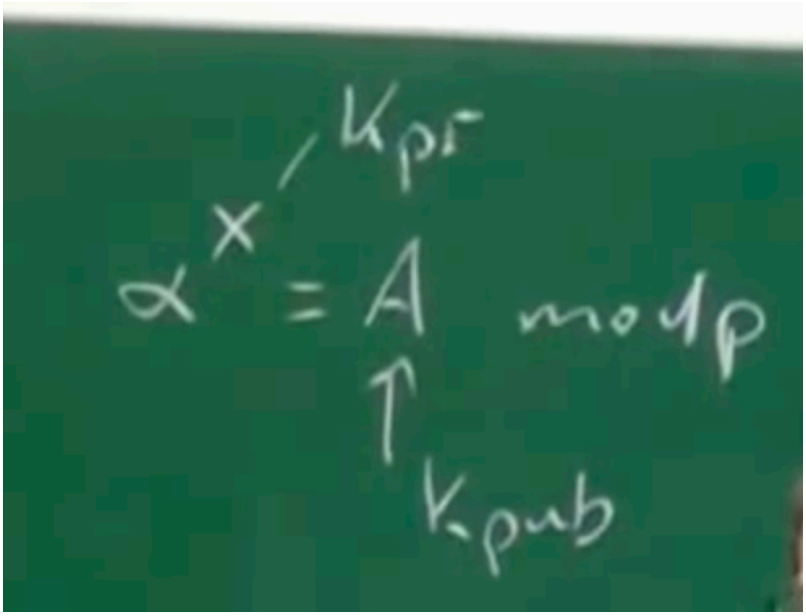
say Bob is the receiver of the message, if bob sets the public encryption scheme then he need to generate the 2 key pairs (Public and Private Key) and Bob is going to push the public key.

Chooses P (Prime element) and alpha

Kpr = d belongs to {2,3....,p-2}
we are calling private key as d becoz private key is used for decryption



Question what is the public key and private key in above equation ? What is hard to compute -> “X” correct, alpha and P are domain parameters so A is the public key.
and this is always the case with Discrete Logarithm systems, exponent is private key and what is computed is non-secrete key.



Kpub = Beta = alpha^d mod p

(Beta, p and alpha)
<——————

as alpha and p are domain paramteers till now but now as bob computed it so we need to send them

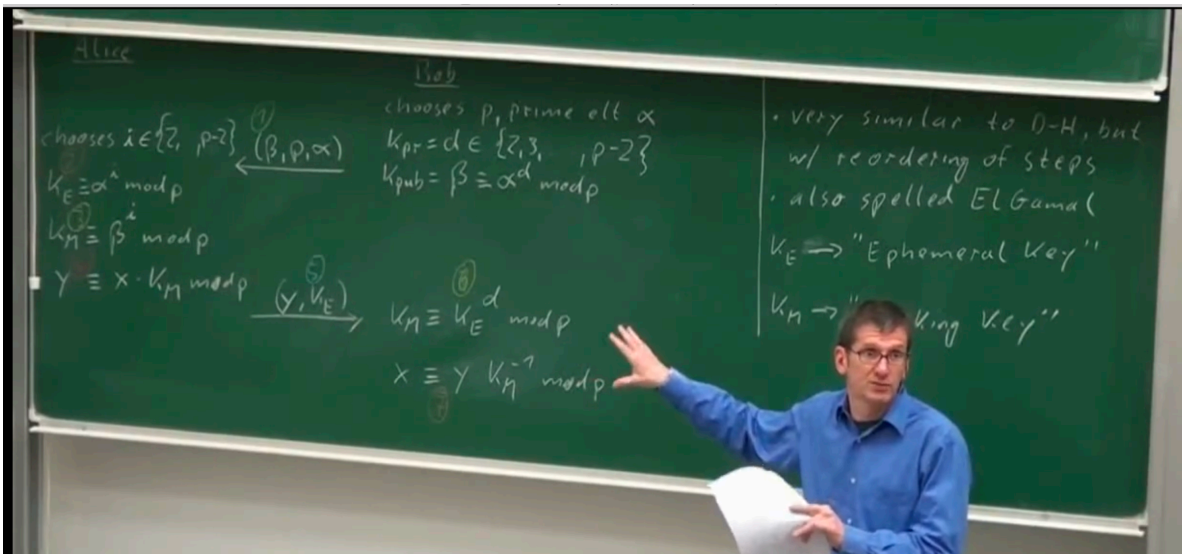
chooses i from {2, … , p-2}
Ke = alpha ^ i mod p
e meaning is Ephemeral key -> Temporal Key
Session Key = Km = Beta ^ i mod p = Kab in diffie hellman

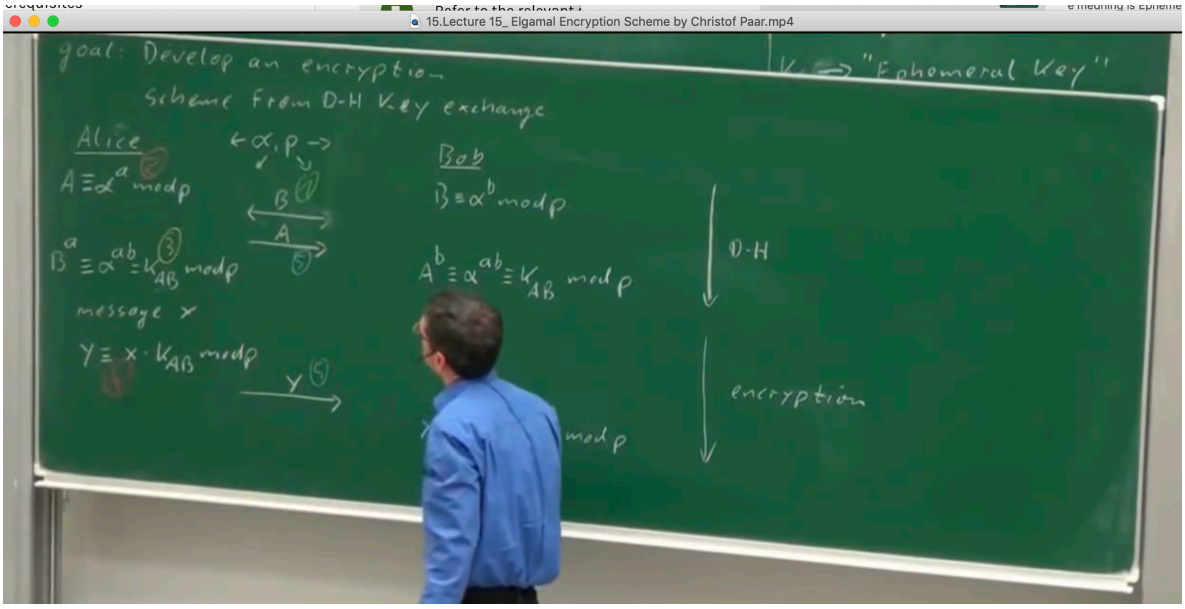
m stands for Masking key ie used by elgamal encryption scheme for multiplication.

now as **Alice has session key so there is nothing stoping alice from doing encryption.**
So alice computes Y = x*KAB mod p , KAB is called Km here so
Y = x*Km mod p.

(Y, Ke -> Public Key ie Ephemeral key)
——————————————————>

Km = Ke^d mod p
x = Y*Km^-1 mod p

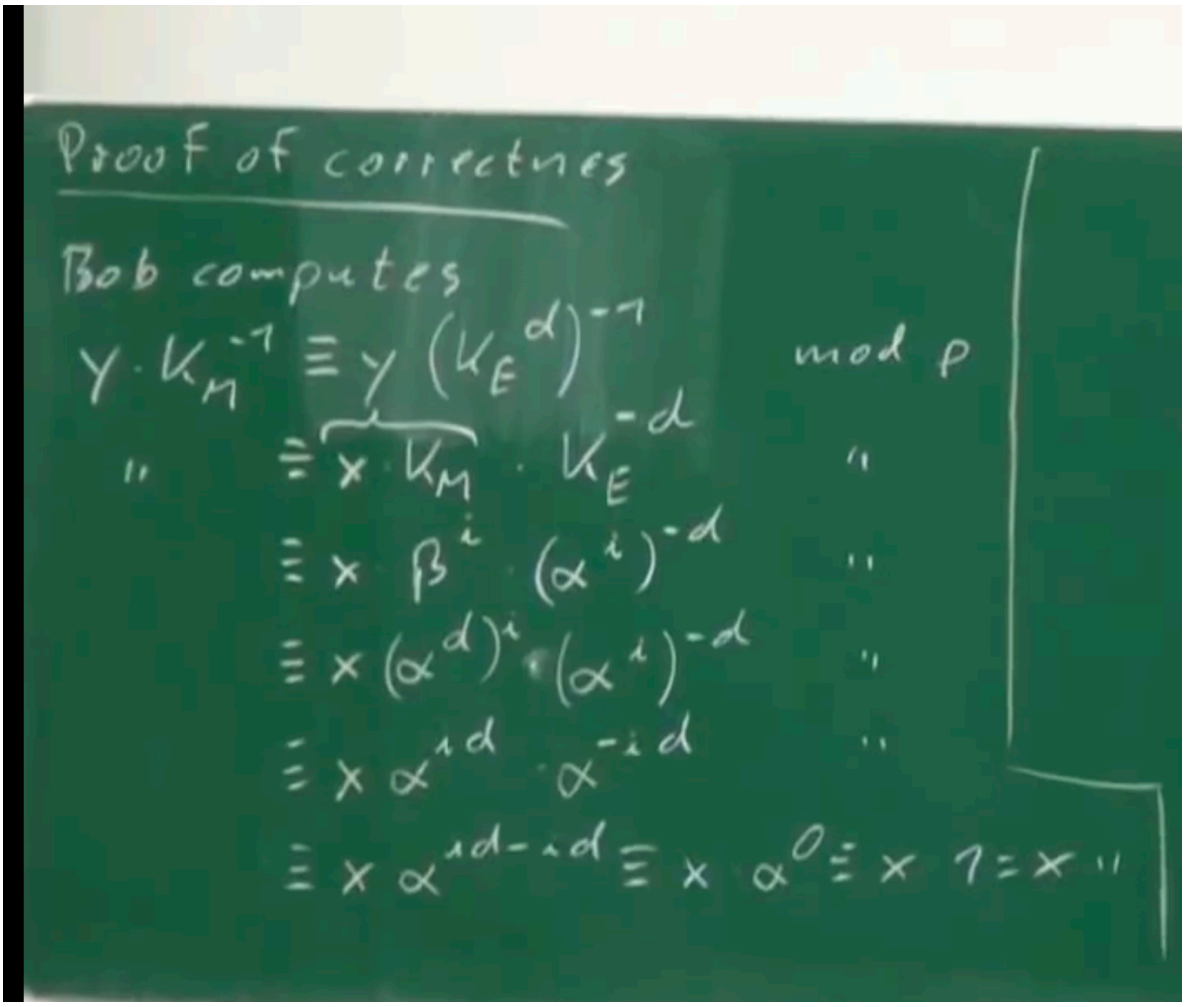




Please match Sequence numbers for more understanding.
Now what is left for bob ? he need to compute the Km and then inverse of Km is multiplied with Y to compute x.

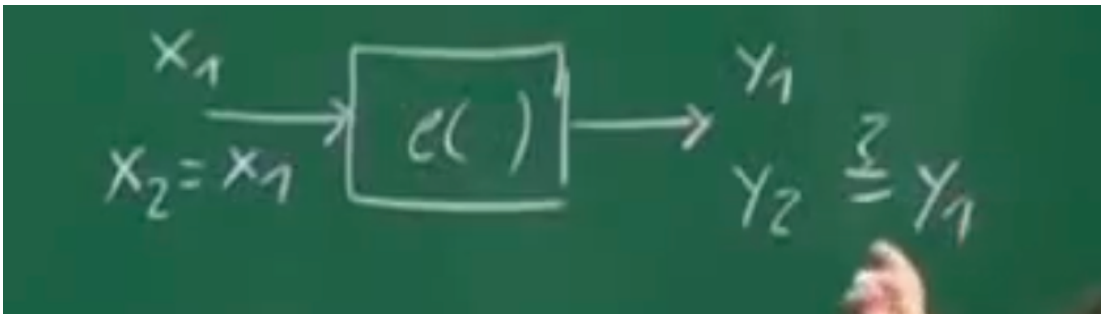
Proof of Correctness:
Does when Bob decrypts he actually get the plain text which alice encrypted.

Bob computes:

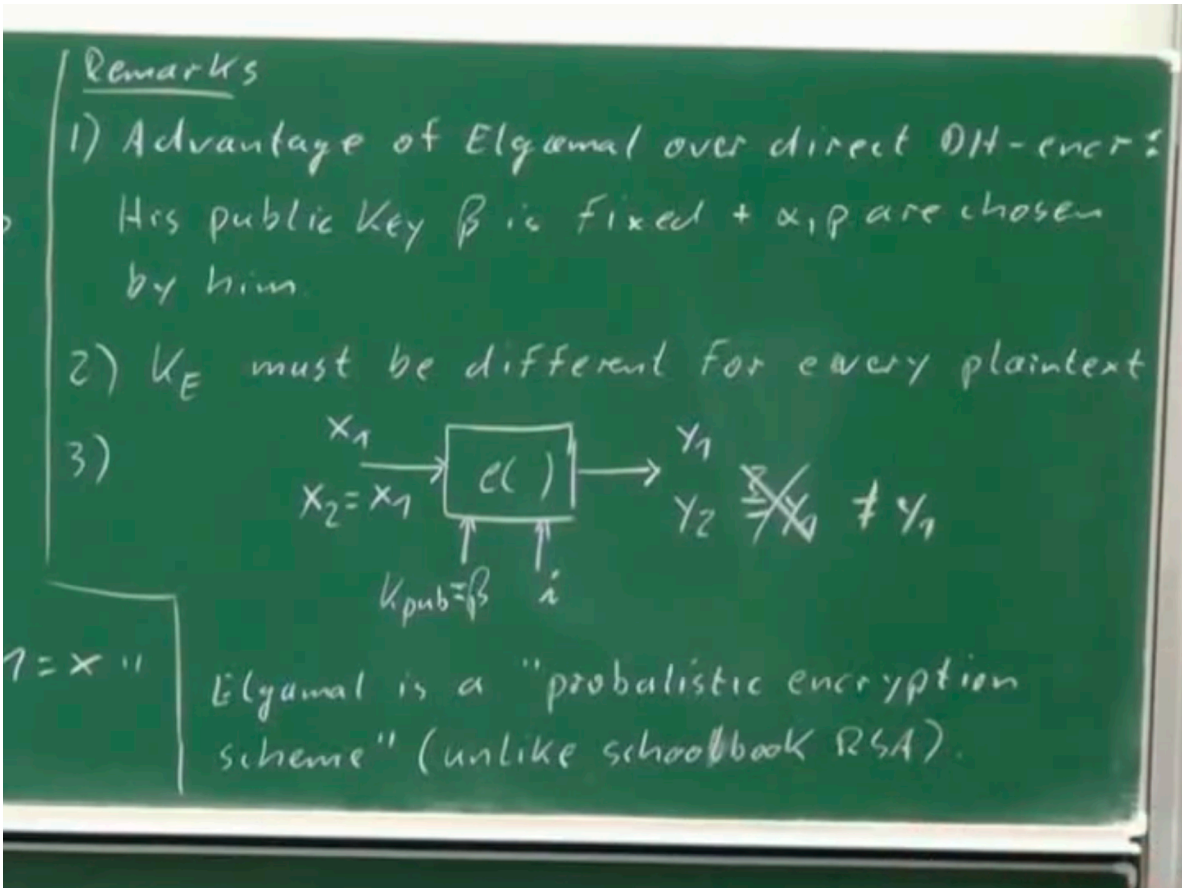


Remarks:

1. Advantage of Elgamal vs Direct Diffie Hellman encryption: His public key beta is fixed and alpha, p are chosen by him.
2. Every time you send the message to bob the Ke which is the public key of alice must be different for very plaintext. So Beta public key remains same and Ke changes everytime and that is the reason Ke is called Ephemeral key/temporal key. **Professor Paar will show in the lecture later why Ke needs to be changed for every plaintext.** That is one key ie beta needs to remain same and other public key ie Ke will change and therefore it is called ephemeral key.
3. **Question:** you bought something from amazon.com with some credit card number say X1 and then after an hour you again used same credit card to buy crypto books so does Y1, y2 ie the encrypted message will be same of different ?



They will be different because we will choose always a new Ke ie we choose a new i and hence our new masking key so you will get different output which is very good cryptographically.



Actually ecnryption function needs a randomised parameter which is “i” here which is causing Ke to change everytime we encrypt the plaintext.

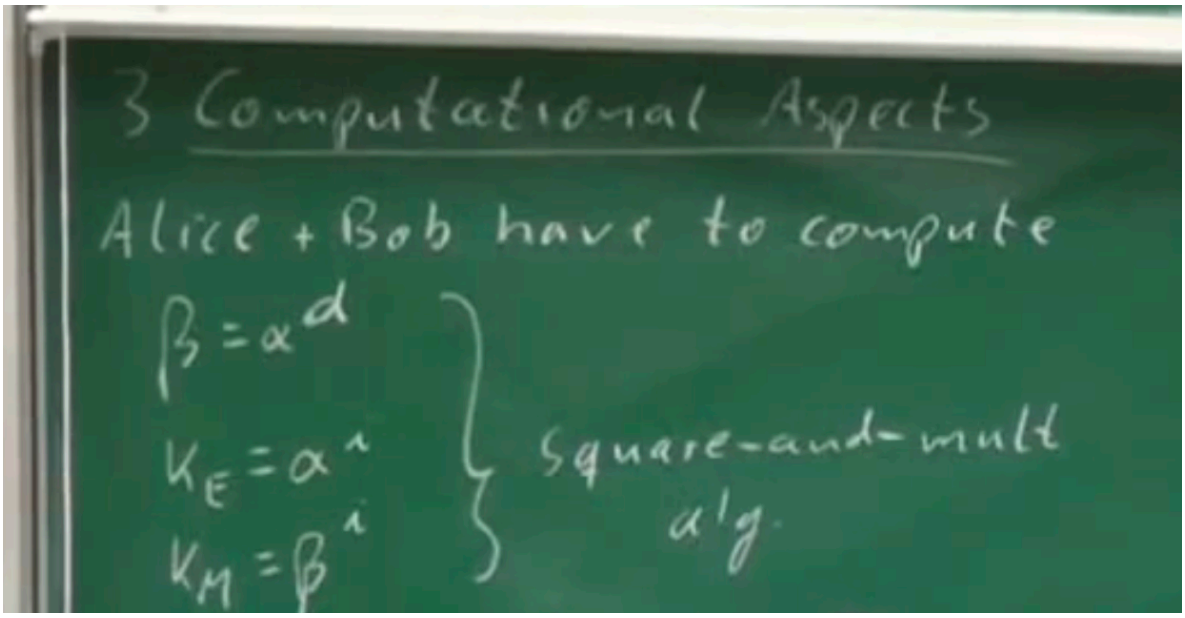
This makes Elgamal as Probabilistic encryption scheme.

we had talked about probabilistic encryption schemes while we are doing modes of operation chaptor, we need to make AES probabilistic then what needs to be done ? we have a parameter called IV which needs to be changed every time we are encrypting for achieving the probabilistic encryption scheme.

Chapter 3: Computational Aspect
we need to compute exponentiation ie for Bobs public key we need to compute

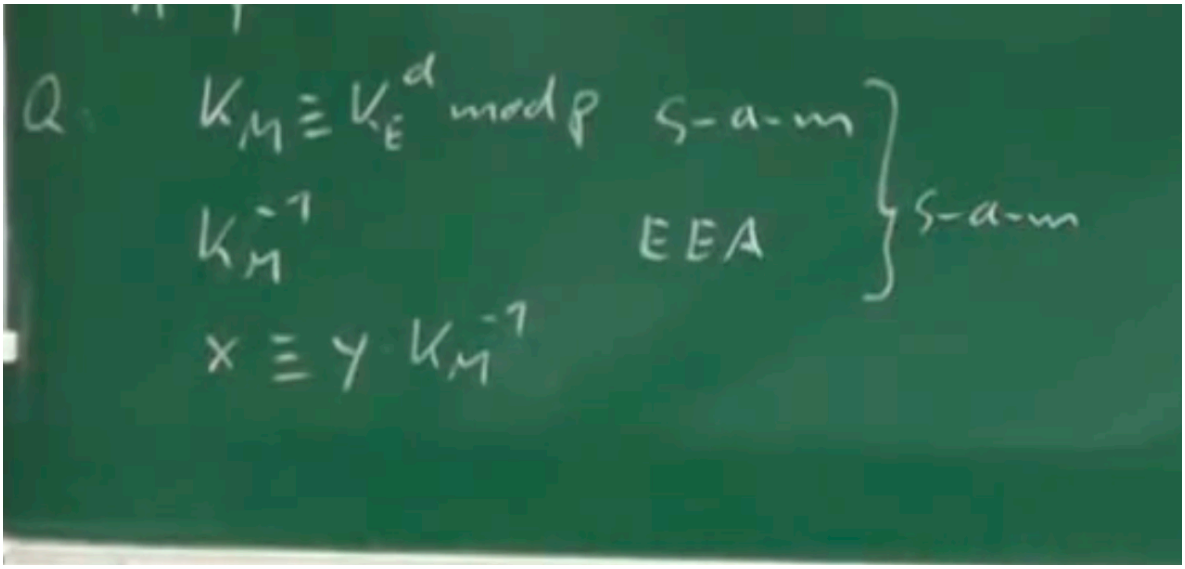
alpha^d mod p where d is choosen from {2,3,...,p-1}
similarly for alice we need to compute Ephemeral key which is Ke = alpha ^ i mod p where i is the private key of Alice

Question is how do we do this exponentiation ? which algorithm ?
Answer: *Square and Multiply Algorithm*



Important thing which is is extra in Elgamal then Basic Diffie Hellman is Multiplication step and then Inverse Multiplication step as Multiplication step is trivial

Question but how can we compute the inverse multiplication step ?
Answer: we can compute the inverse using Extended Euclidean algorithm and then multiply



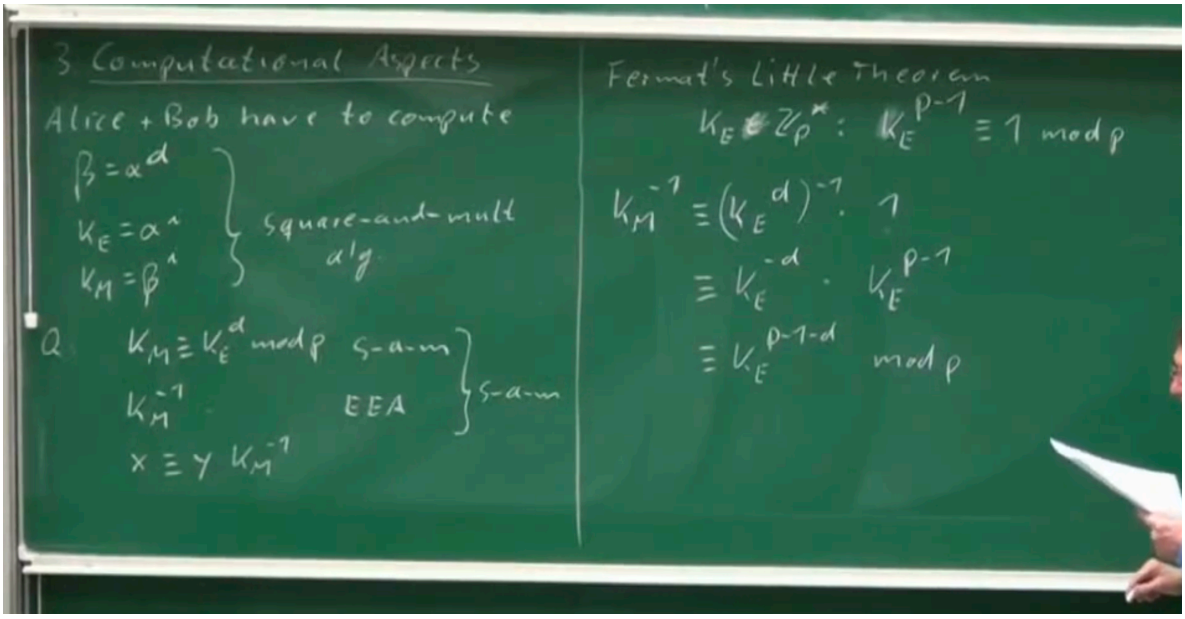
Above is the Naive way to handle inversion implementation so what can be done?

Actually we can combine the inverse step in Square and Multiply algorithm.
Question is How ?
Fermat's little theorem:
Remember we have done the fermat's little theorem which states that

Ke belongs to Zp* then Ke^|Zp*|-1 = 1 mod p
or we can write it as Ke^p-1 = 1 mod p

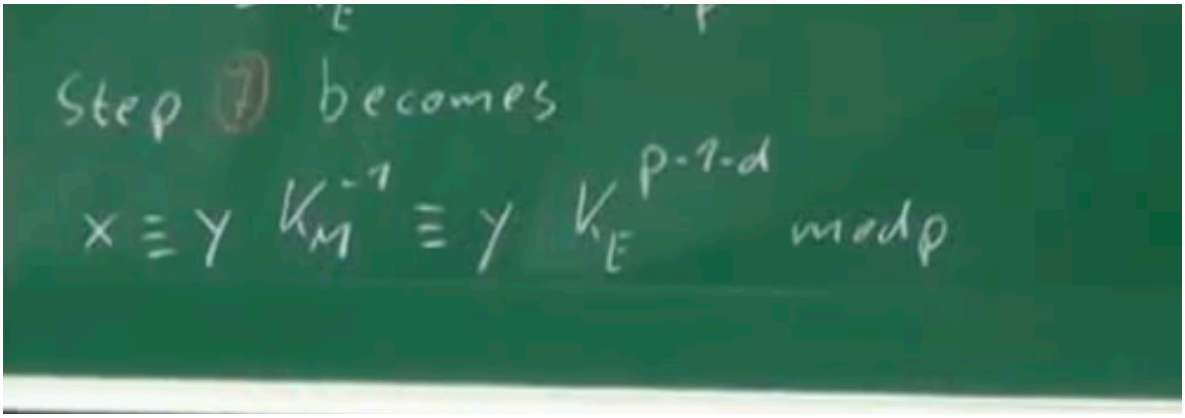
so we want inverse of Masking key so what is Km => Ke^d mod p
so km^-1 = ke^-d mod p
now

we know fermat's little theorem so we will write :
km^-1 =(ke^-d mod p) * 1
km^-1 = (ke^-d mod p) * (Ke^p-1 **)**
km^-1 = (ke^-d + p -1 mod p)
km^-1 = (ke^ p - d -1 mod p)



if we think more on this then we know **-d** is kind of inverse only so we here using fermat's little theorem changed this to positive number so that we can using Square and Multiply easily
Also this is my personal assumption that we should choose d from {2,3, ...,p-2} to always ensure that p-d-1 > 0

Below is wrong way and explains why it is wrong=>
While writing this i also thought that Ke can be replaced with alpha^i but that is not useful as power will remain still negative and also i is not available to bob as it is private key of Alice

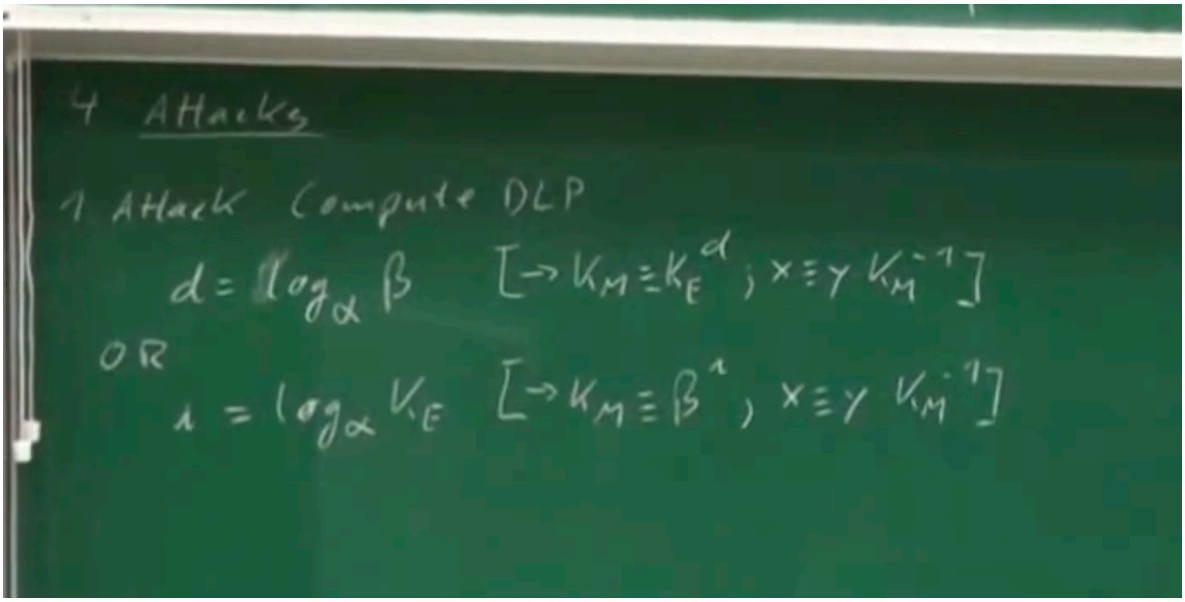


Attacks:
We will talk about 2 attacks:
First Attack:
we can attack Diffie hellman ie Discrete Algorithm Problem or we can attack the encryption.

So what does oscar wants: He want to know the X, he knows most of the parameters of algorithm but what are the two parameters which he doesn't know ?

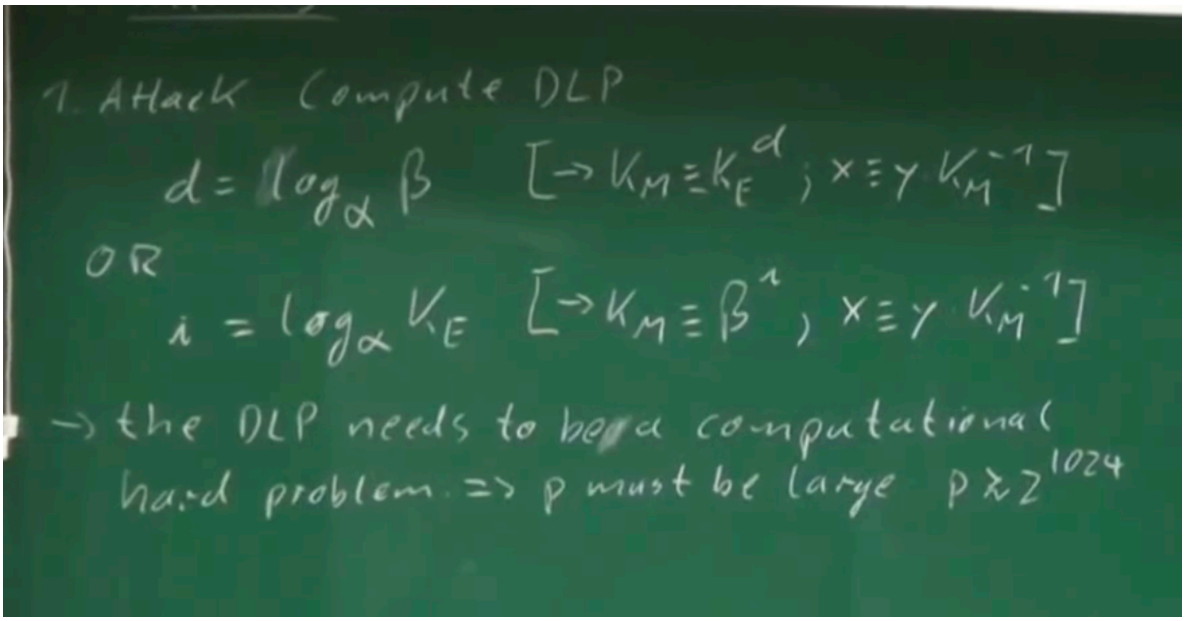
1. **i and**
2. **d ie both the private keys.**

First attack is Compute the discrete Logarithm problem ie we tries to find d which is
Log beta/log alpha mod p then he can compute Km and then he can decrypt and same way he can compute i using Log Ke/ log alpha



Problem here is he has to compute the Discrete Logarithm problem. so the DLP

needs to be computationally hard problem.
So as a designer we want discrete Logarithm problem is very hard and what we need to do for making it hard ? ie how can we ensure that is cannot be computed using pocket calculator ?
Making P large is p > 1024 bit length atleast.



there are other bunch of requirements on P and Alpha which we have not read/ discussed but main thing for us as per the information we have, is P should be large.

We have discussed earlier that if P is not large then there are 2 attacks where one attack was square root attack which makes half the complexity ie if key is 200 bits then that makes it 100 bit complex and that was the reason EC is 160 bits because 80 bits is tough to compute currently. Other attacks is out of scope of this lecture.

Second Attack:
This attack happens much more in practice when you are not careful.

As we know we need Ephemeral Key which is temporary and should not be used twice so Ephemeral key is coming from i which is chosen randomly from the group {2,3,...,p-2} but if random number generator fails due to some reason and you are getting same i out.

may be i looks quite large but same i is returned then there can be some bad things which can happen.
Attack Scenario :
Ke = alpha ^ i mod p
Km = beta ^ i

Say between 2 communications **i** didn't change:
so

Alice Bob

Alice has to send encrypted message and Ephemeral Key

(Y1, Ke)

Y1 = X1*Km ————> X1 = Y1*Km^-1

(Y2, Ke)

————>

Y2 = X2*Km X2 = Y2^Km^-1

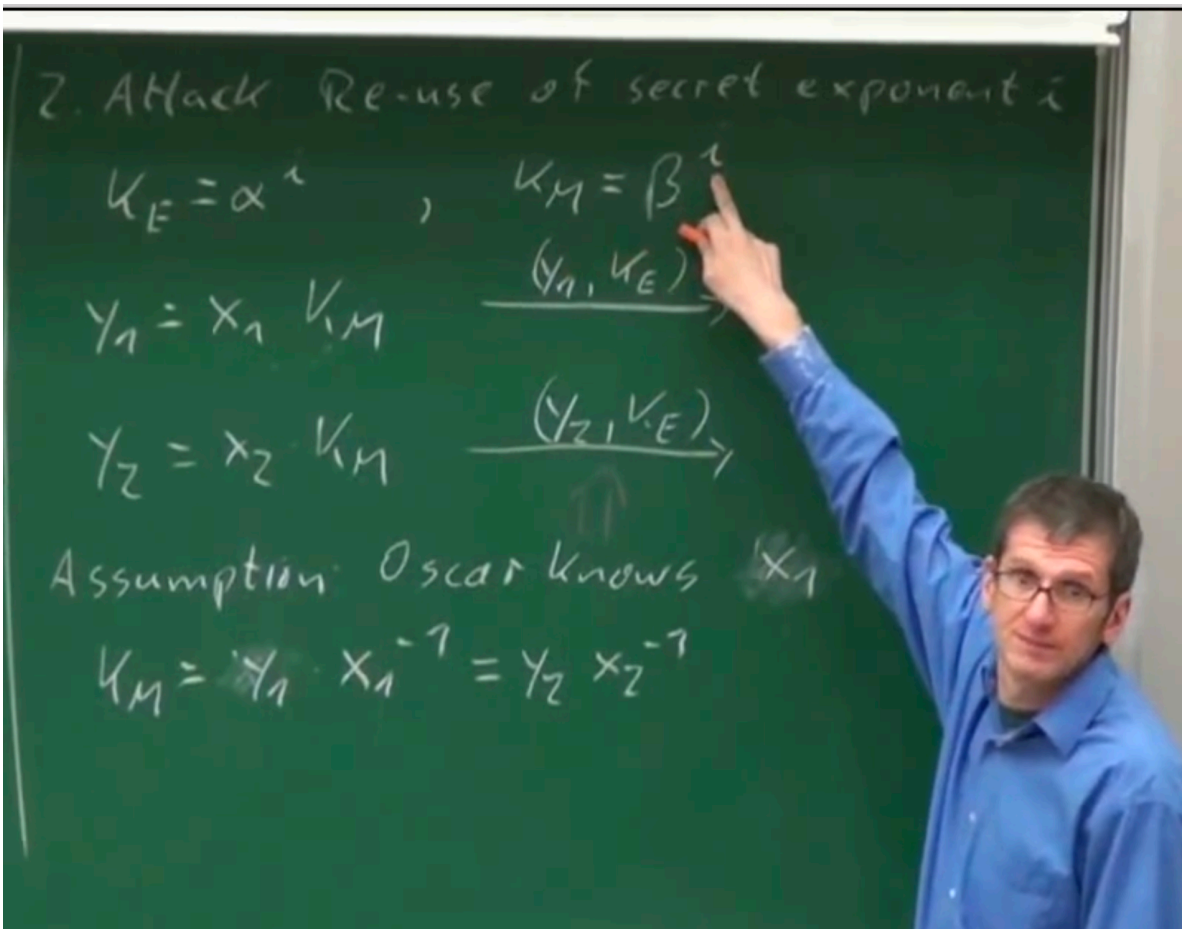
If Oscar has observed these two call then

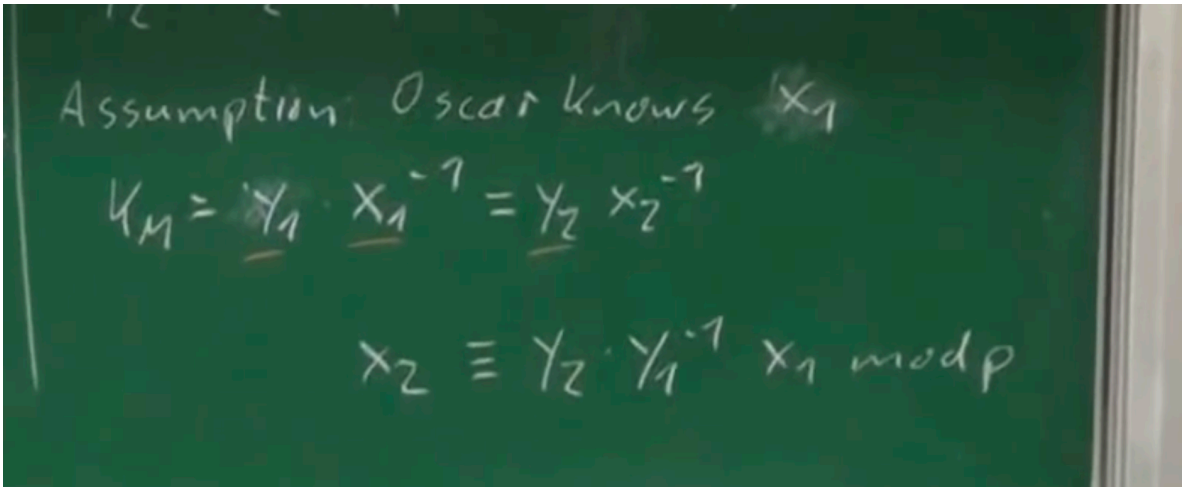
Question: can we find out that same random generator has failed ?

Answer: Yes, How ?

Actually Ke will be same for both the messages so that means i is also same for both the messages and hence Km is also same for both the messages.

So assumption that oscar knows X1 then he can compute Km using equation Km = Y1*X1^-1 and then he can compute X2 using the same as he knows Km and Y2 both in second equation.





Now everything that alice encrypt from now onwards can be decrypted by oscar.