Cryptography - Day 2

August 11, 2020

Asymmetric Encryption

- Private Key
- Public Key

Private and Public Key

The key which you keep private is called private key and the one which you publicly share, is called public key!

Why is it used ?

Passwords

- Password Hashing
- Dictionary Attacks
- Salting (Intro)

Hashing

Hashing is a process to convert a given input into a fixed length output, based on certain computation on the given input!

Given an input of length ‘**x**’, MD5 hashing algorithm will always produce an output of length 128 bits!

Why Hashing?

Indexing

Why Hashing?

Indexing

Integrity Checks

Why Hashing?

Indexing

Integrity Checks

Password Storage

Why Hashing?

Indexing

Integrity Checks

Password Storage

Bit Commitment

Why Hashing?

Indexing

Integrity Checks

Password Storage

Bit Commitment

Cryptocurrency

Properties of Hash Functions

One - Way

Collision Free

Avalanche Effect!

Hashes

- MD5
- SHA

MD5

MD5 (“**Where is my mug?**”) = 0740e0a670c22e9264a3563fd23fd1b7

MD5 (“**Where is my jug?**”) = bc37f1c43264f02a2191ea3d063bf47e

MD5 (“**B**”) = 9d5ed678fe57bcca610140957afab571 (B = “01000010”)

MD5 (“**C**”) = 0d61f8370cad1d412f80b84d143e1257 (C = “01000011”)

SHA-256

SHA256 (“**B**”) B = “01000010”

df7e70e5021544f4834bbee64a9e3789febc4be81470df629cad6ddb03320a5c

SHA256 (“**C**”) B = “01000011”

6b23c0d5f35d1b11f9b683f0b0a617355deb11277d91ae091d399c655b87940d

Why is it One-Way ?

Hashing converts an input of any given length into a fixed length output! Therefore, you have loss of data!

Example:

MD5(“1”) will produce an output of len 128 bits

MD5(Image) will produce an output of len 128 bits

MD5(Video) will produce an output of len 128 bits

MD5(Hard Disk) will produce an output of len 128 bits

Hashing (vs) Encryption

Hashing	Encryption
Cannot be reversed (1-Way)	Can be reversed (2-Way): Decryption
Key is not needed always	Needs a key
Loss of information	Information is retained
Used for Integrity	Used for Confidentiality

Can be break hashes ?

What does it mean to break a hash ?

Should we try brute force ? Is it worth the effort ?

Dictionary Attack

A dictionary attack is based on trying all the strings in a pre-arranged listing.

In contrast to a brute force attack, where a large proportion of the key space is searched systematically, a dictionary attack tries only those possibilities which are deemed most likely to succeed.

Dictionary attacks are very easily defeated using …….

Salting

A cryptographic salt is made up of random bits added to each password instance before its hashing. Salts create unique passwords even in the instance of two users choosing the same passwords.

MD5 (“password”) = 5F4DCC3B5AA765D61D8327DEB882CF99

5F4DCC3B5AA**BCE**765D6**FG**1D**HDW**8327DEB882CF99

5F4DCC3B5AA765D61**WAFS**D8327**IQLJ**DEB**P**882CF99

Key Exchange

- Diffie–Hellman
- Let’s break it!

Diffie–Hellman

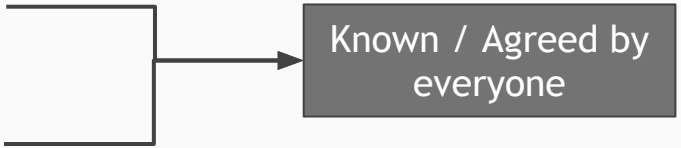
The point is to agree on a “key” that two parties can use for a symmetric encryption, in such a way that an eavesdropper cannot obtain the key.

Let’s see the working!

Terminology

‘p’ - A huge prime number

‘g’ - A generator number



‘a’ - Alice’s private number

‘b’ - Bob’s private number

Generating ‘p’ and ‘g’

Do we need to generate ‘p’ and ‘g’ every time we need to do a Diffie Hellman key exchange ?

Generating ‘p’ and ‘g’

Do we need to generate ‘p’ and ‘g’ every time we need to do a Diffie Hellman key exchange ?

Finding a large prime number ‘p’ is computationally heavy, so since ‘p’ and ‘g’ are going to be known by everyone, why not use some ‘p’ and ‘g’ which someone else had used ?

Alice
(a, g, p)

Public (Internet)
(g, p)

Bob
(b, g, p)

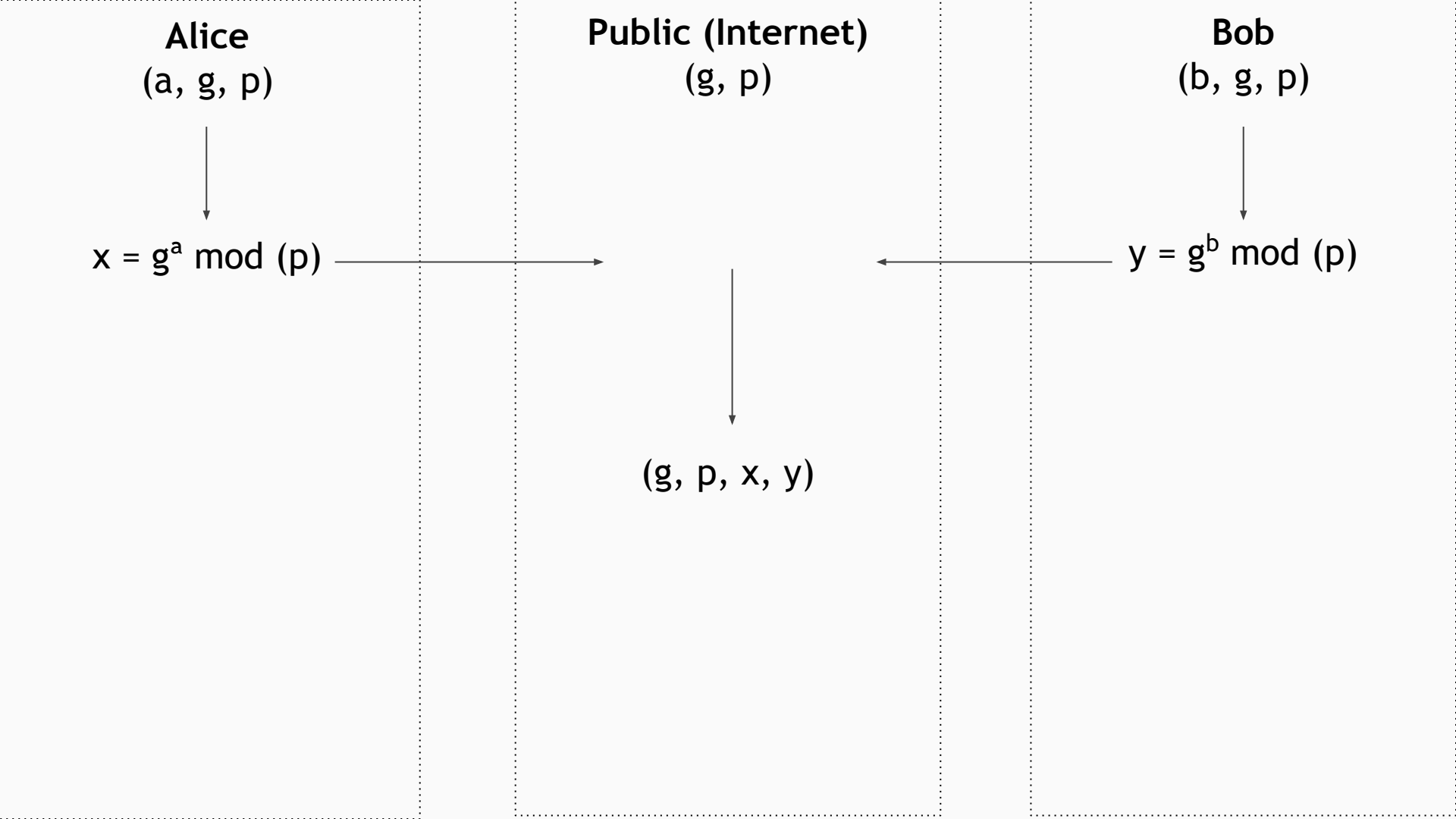
Alice
(a, g, p)

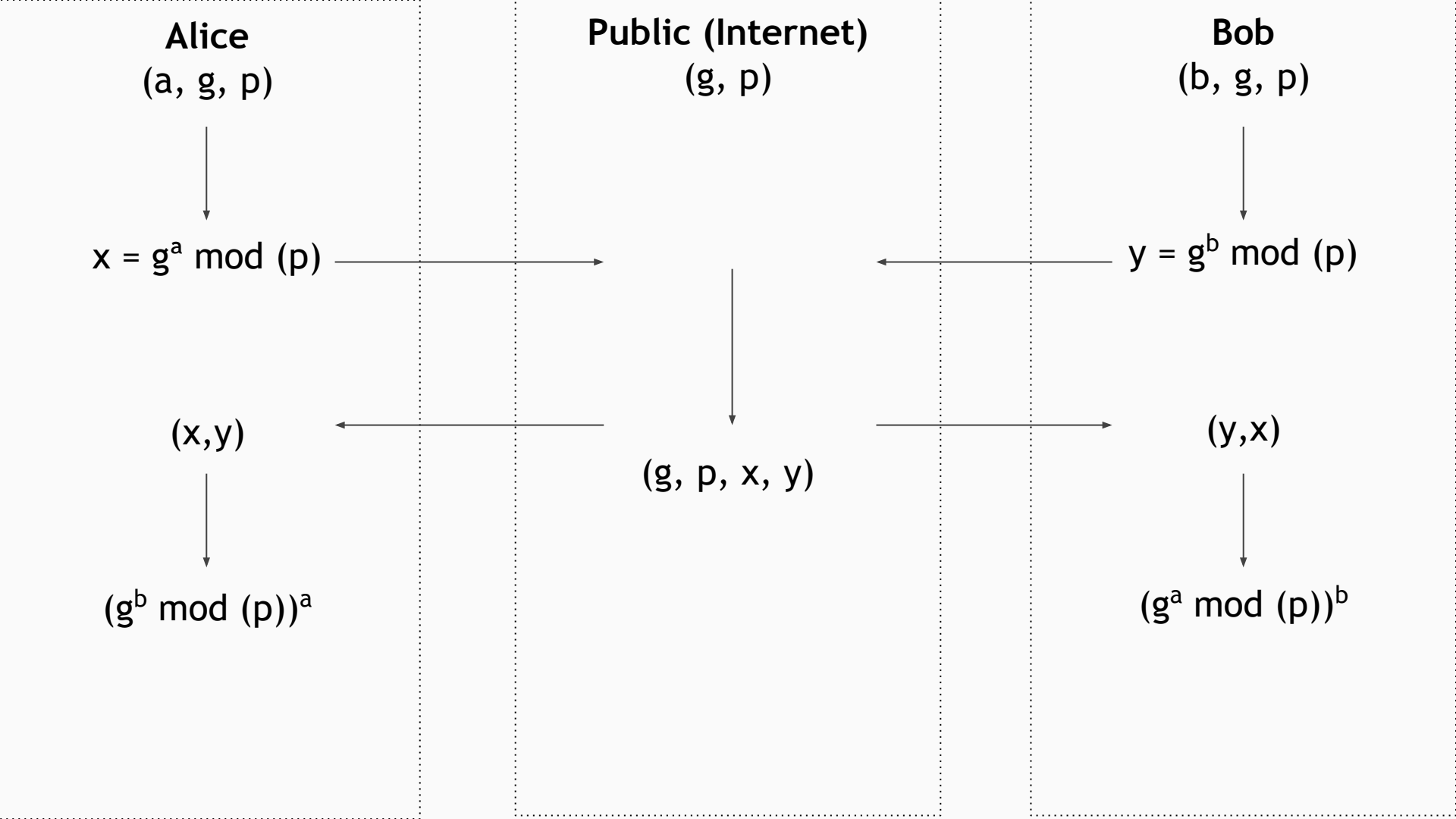
$g^a \bmod (p)$

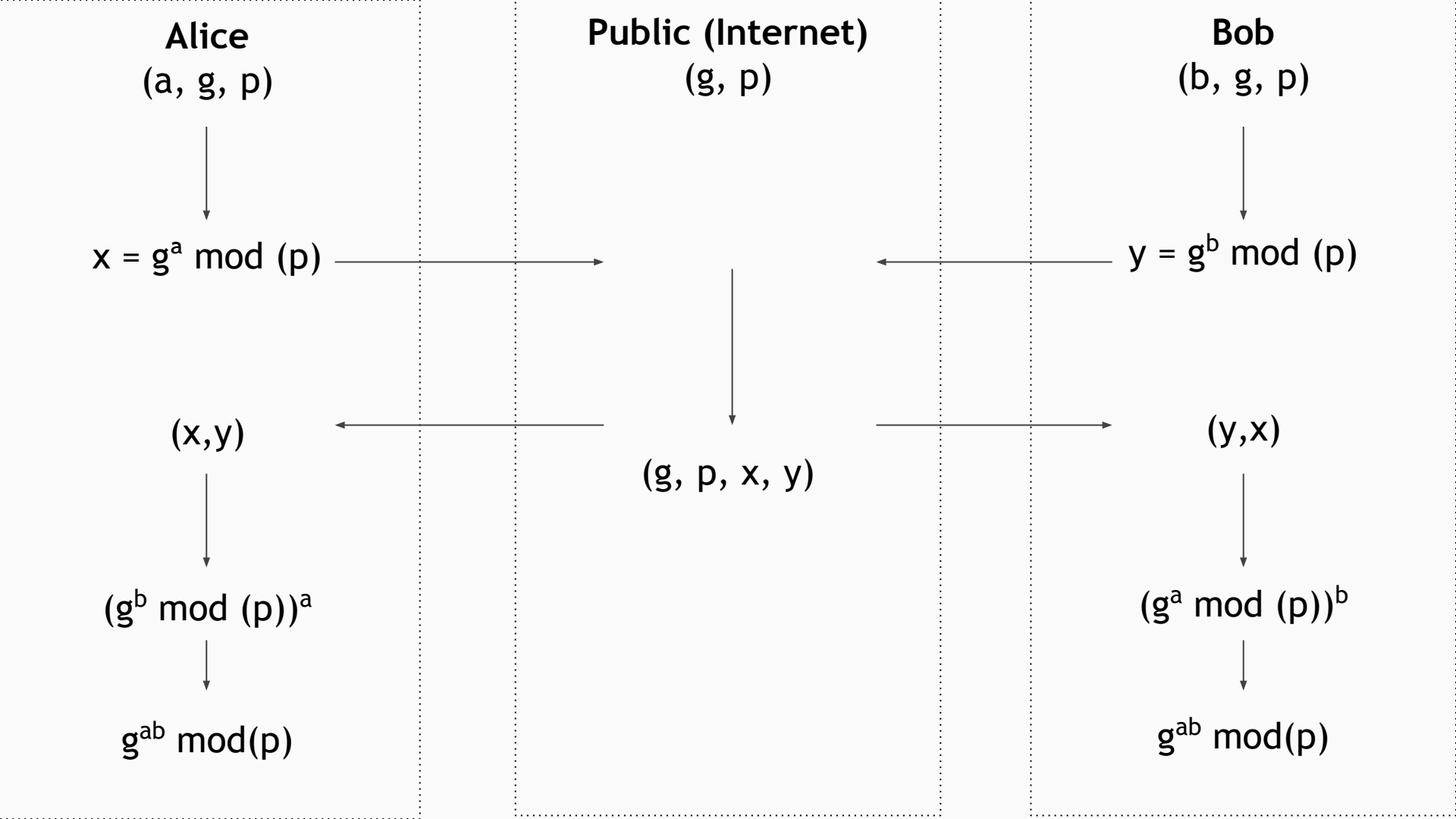
Public (Internet)
(g, p)

Bob
(b, g, p)

$g^b \bmod (p)$


Alice
(a, g, p)
Public (Internet)
(g, p)
Bob
(b, g, p)
x = g^a mod (p)
y = g^b mod (p)
(g, p, x, y)


Alice
(a, g, p)
Public (Internet)
(g, p)
Bob
(b, g, p)
x = g^a mod (p)
y = g^b mod (p)
(x,y)
(g, p, x, y)
(y,x)
(g^b mod (p))^a
(g^a mod (p))^b


Alice
(a, g, p)
Public (Internet)
(g, p)
Bob
(b, g, p)
x = g^a mod (p)
y = g^b mod (p)
(x,y)
(g, p, x, y)
(y,x)
(g^b mod (p))^a
(g^a mod (p))^b
g^ab mod(p)
g^ab mod(p)

Diffie–Hellman

Discovering ‘a’ from g^a mod (p) and ‘b’ from g^b mod (p) will take longer than the lifetime of the universe, using the best known algorithm. This is called the **Discrete Logarithm problem**.

That is the reason brute-forcing Diffie–Hellman for a large ‘p’ is just unimaginable!

Diffie–Hellman

How can two parties agree on a secret value when all of their messages might be overheard by an eavesdropper?

So can we break it using eavesdropping ? Let’s eavesdrop!

Alice
(a, g, p)

Attacker (Internet)
(m, g, p)

Bob
(b, g, p)

Alice
(a, g, p)

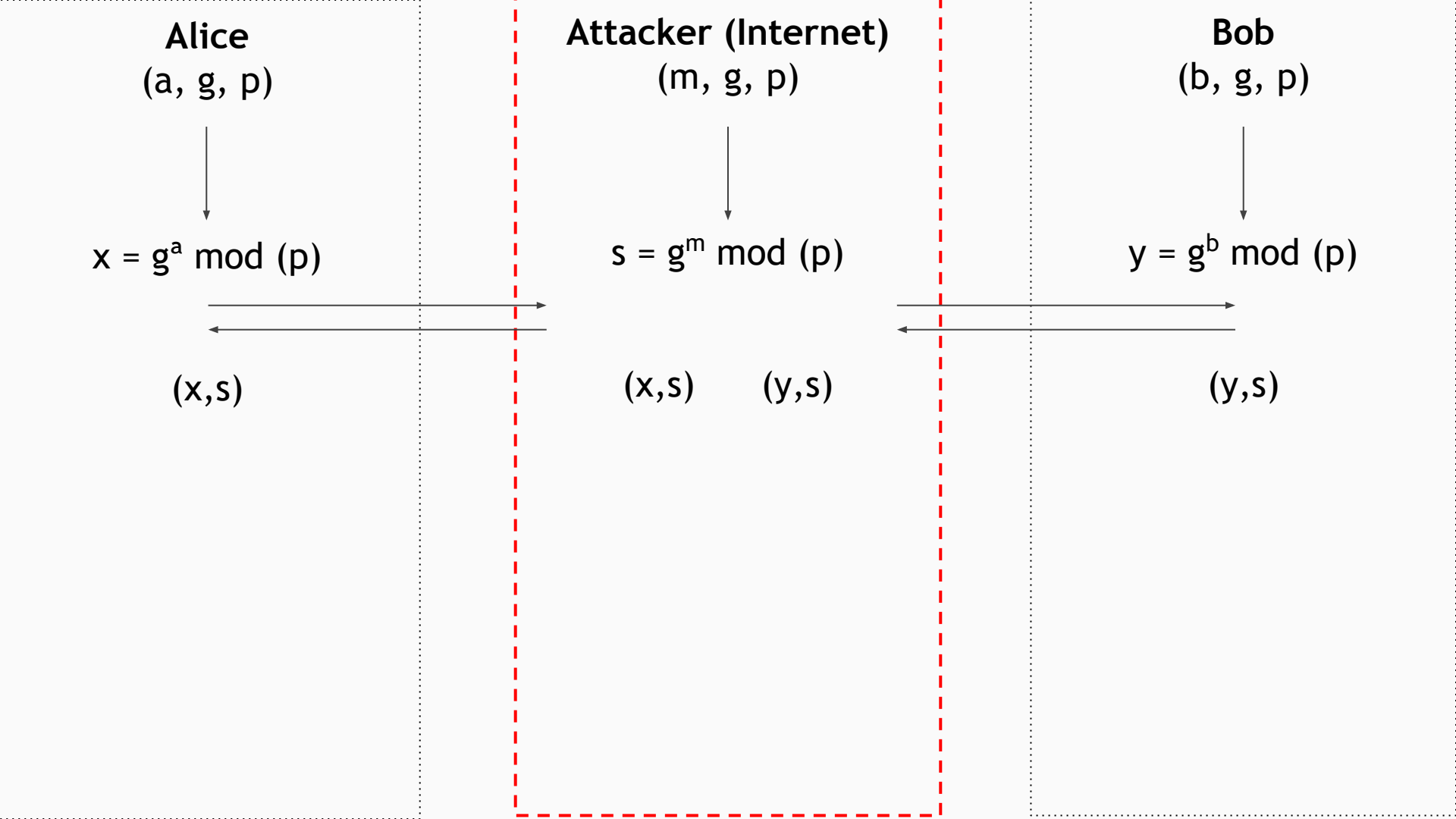
$x = g^a \bmod (p)$

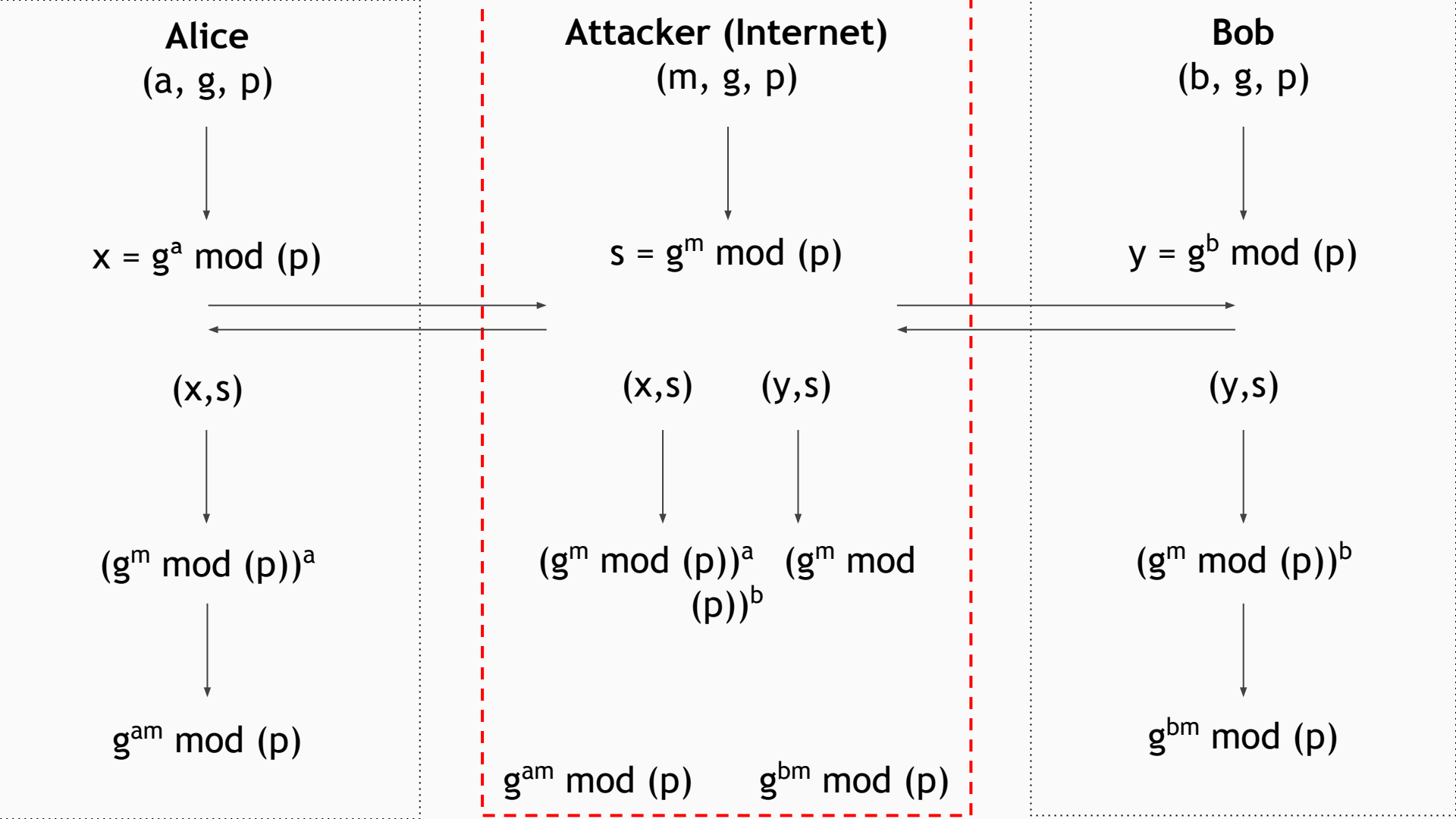
Attacker (Internet)
(m, g, p)

$s = g^m \bmod (p)$

Bob
(b, g, p)

$y = g^b \bmod (p)$


Alice
(a, g, p)
x = g^a mod (p)
(x,s)
Attacker (Internet)
(m, g, p)
s = g^m mod (p)
(x,s)
(y,s)
Bob
(b, g, p)
y = g^b mod (p)
(y,s)


Alice
(a, g, p)
x = g^a mod (p)
(x,s)
(g^m mod (p))^a
g^am mod (p)
Attacker (Internet)
(m, g, p)
s = g^m mod (p)
(x,s)
(y,s)
(g^m mod (p))^a
(g^m mod (p))^b
g^am mod (p)
g^bm mod (p)
Bob
(b, g, p)
y = g^b mod (p)
(y,s)
(g^m mod (p))^b
g^bm mod (p)

Limitations of Diffie-Hellman

Diffie-Hellman provides key exchange mechanism, but it doesn’t provide authentication!

How can two parties agree on a secret value when all of their messages might be overheard by an eavesdropper?

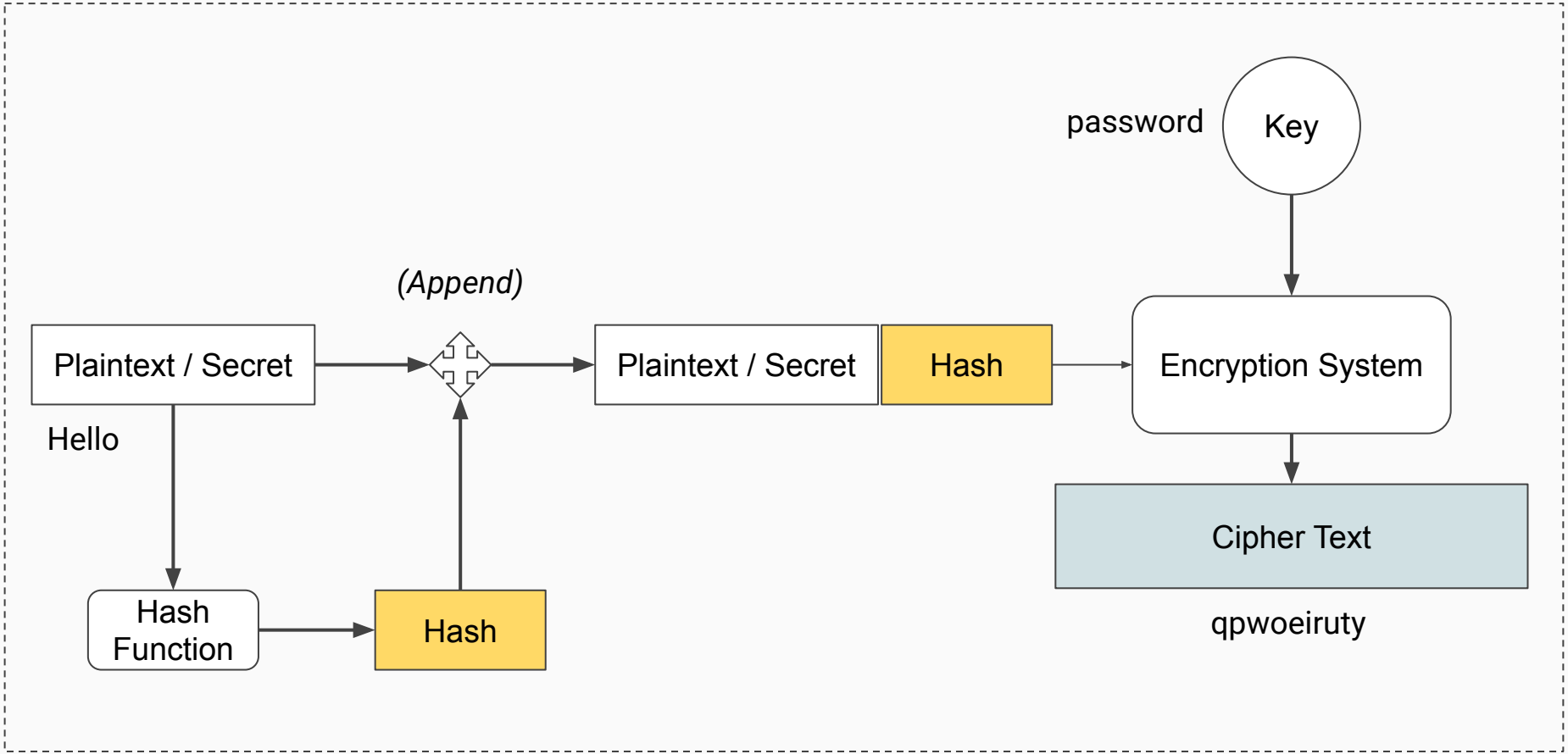
How to secure Diffie–Hellman

How can two parties agree on a secret value when all of their messages might be overheard by an eavesdropper?

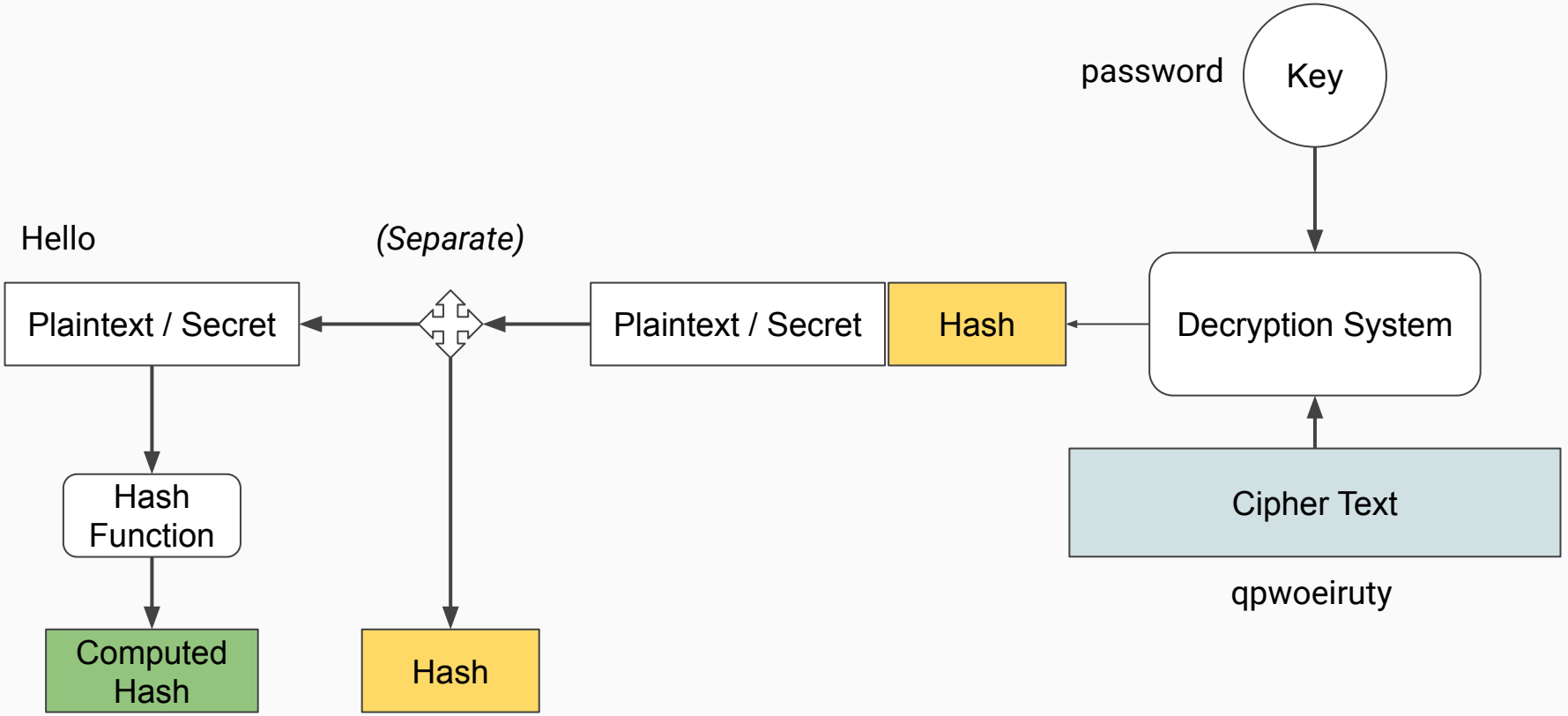
PKI (Public Key Infrastructure) is based on the concept of asymmetric encryption! Use RSA

How to Achieve

- Confidentiality
- Integrity



Process of Cryptography - Encryption (Confidentiality and Integrity)



“Check if both these hashes are the same”

RSA

- Basics of RSA

Intro to RSA

Let’s talk about ‘Alice’ first!

Two prime numbers ‘**p**’ and ‘**q**’

n = p*q

ϕ(n) = (p-1)*(q-1)

Select ‘e’ such that ‘**e**’ is **co-prime** to ‘**ϕ(n)**’

Calculate ‘d’ such that **e*d mod(ϕ(n)) = 1**

Intro to RSA

Let’s talk about ‘Alice’ first!

Two prime numbers ‘p’ and ‘q’

$n = p*q$

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

Select ‘e’ such that ‘e’ is co-prime to ‘$\phi(n)$’

Calculate ‘d’ such that $e*d \bmod(\phi(n)) = 1$

Public Key = (e,n)

Private Key = (d,n)

Intro to RSA

p = 3, q = 11

n = p*q = 33

ϕ(n) = (p-1)*(q-1) = 20

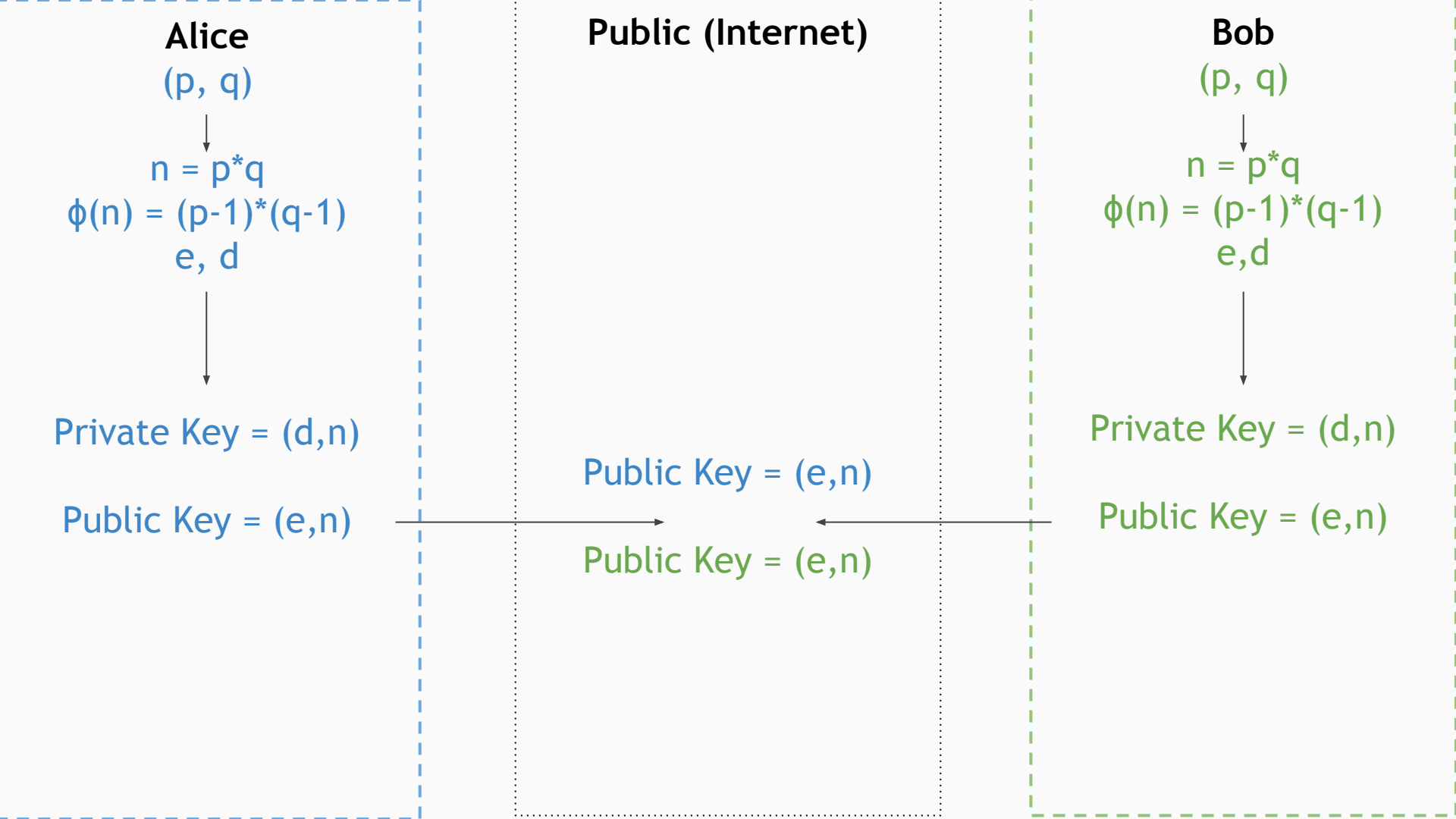
e = 3

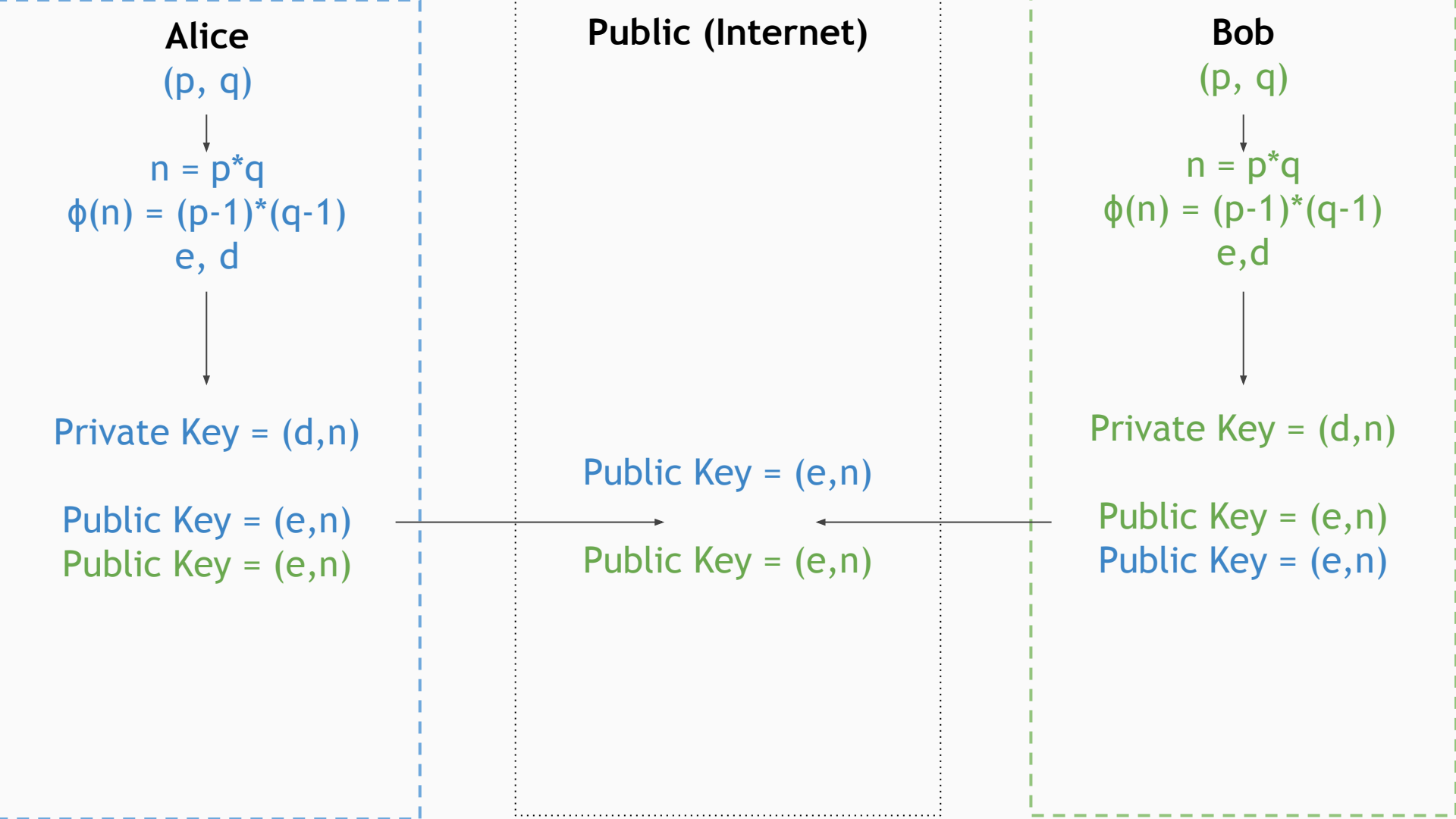
How do we calculate ‘d’ for the given ‘e’

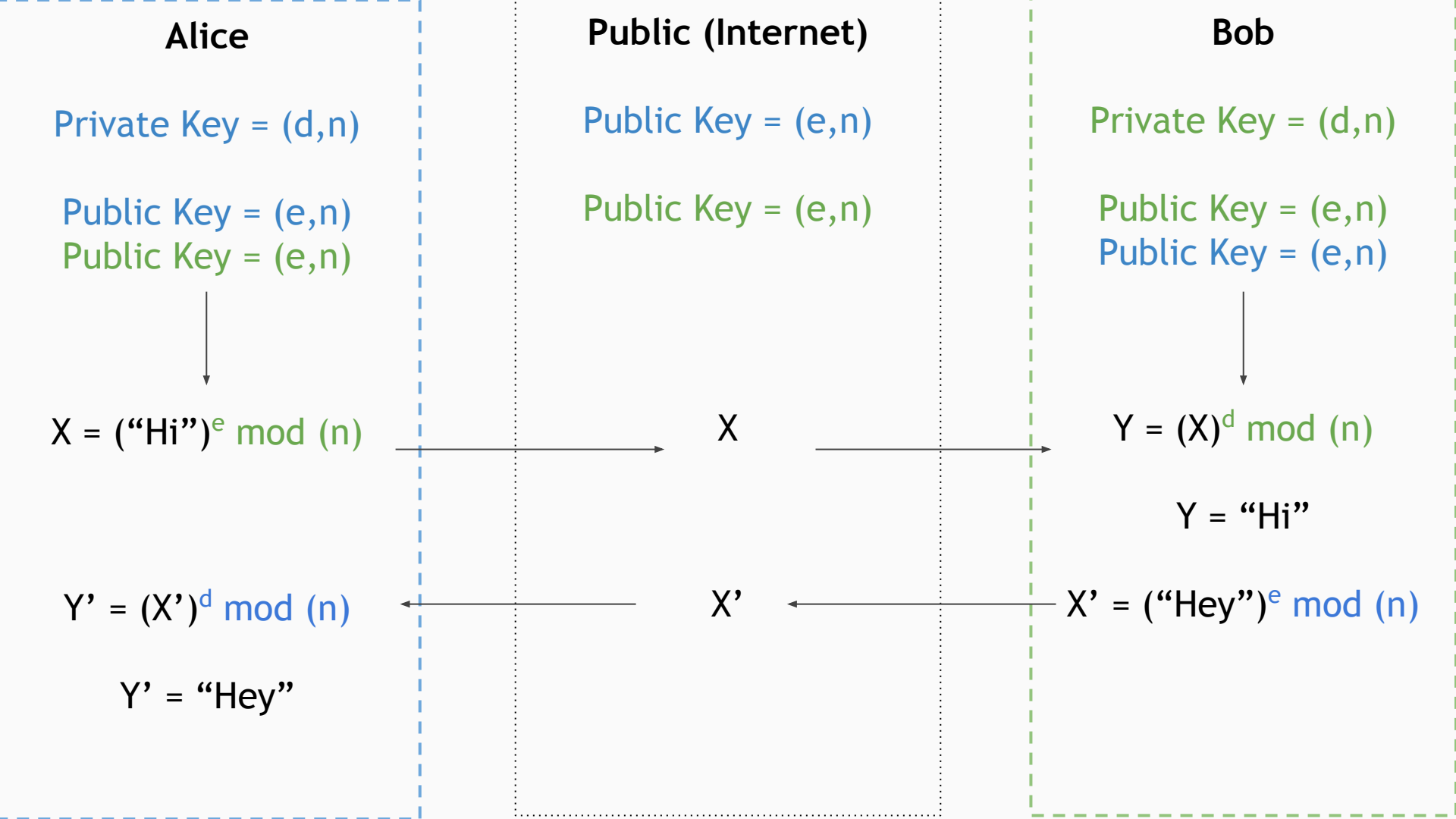
Using Extended Euclidean Algorithm d = 7

Public Key = (e,n) = (3,33)

Private Key = (d,n) = (7,33)


Alice
(p, q)
n = p*q
ϕ(n) = (p-1)*(q-1)
e, d
Private Key = (d,n)
Public Key = (e,n)
Public (Internet)
Public Key = (e,n)
Public Key = (e,n)
Bob
(p, q)
n = p*q
ϕ(n) = (p-1)*(q-1)
e,d
Private Key = (d,n)
Public Key = (e,n)


Alice
(p, q)
n = p*q
ϕ(n) = (p-1)*(q-1)
e, d
Private Key = (d,n)
Public Key = (e,n)
Public Key = (e,n)
Public (Internet)
Public Key = (e,n)
Public Key = (e,n)
Bob
(p, q)
n = p*q
ϕ(n) = (p-1)*(q-1)
e,d
Private Key = (d,n)
Public Key = (e,n)
Public Key = (e,n)


Alice
Private Key = (d,n)
Public Key = (e,n)
Public Key = (e,n)
X = (“Hi”)^e mod (n)
Y’ = (X’)^d mod (n)
Y’ = “Hey”
Public (Internet)
Public Key = (e,n)
Public Key = (e,n)
X
X’
Bob
Private Key = (d,n)
Public Key = (e,n)
Public Key = (e,n)
Y = (X)^d mod (n)
Y = “Hi”
X’ = (“Hey”)^e mod (n)

Usage of RSA

Key Exchange

Secret Messaging

Authentication

Signatures

Certificates

Thank You

Atit Gaonkar

atit-gaonkar.me/
linkedin.com/in/atit-gaonkar/
instagram.com/atit.sgaonkar/