Security Overview

- Background
- Web app vulnerabilities
- Securing web apps

Famous quote of the day

“Every program has at least two purposes:
the one for which it was written, and
another for which it wasn't.”

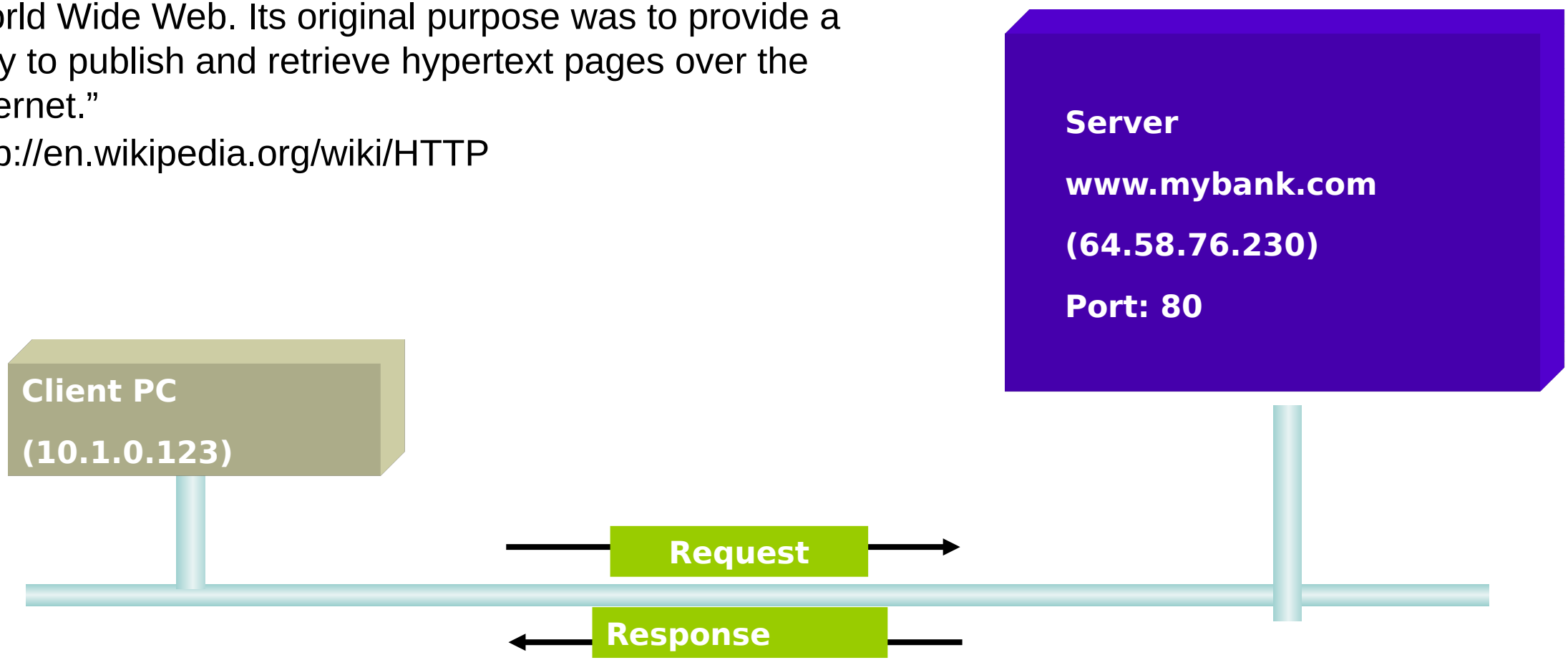
-Alan J. Perlis

HTTP

Hypertext Transfer Protocol

"Hypertext Transfer Protocol (HTTP) is a communications protocol for the transfer of information on intranets and the World Wide Web. Its original purpose was to provide a way to publish and retrieve hypertext pages over the Internet."

http://en.wikipedia.org/wiki/HTTP



Typical web app stack

- Browser (client) : DOM, tabs, Javascript, different clients
- HTTP over TCP/IP
- Server
  - ✔ Operating system
  - ✔ Web server
  - ✔ Scripting language
  - ✔ Database or persistence layer

LAMP

- Looking only at the server:
  - **Linux**
  - **Apache**
  - **MySQL**
  - **PHP/Python/Perl**
- Not only is this model common, it is easy to set up in a lab environment
- Security exists at every level of the stack
- Sadly, so does vulnerability

Fertile Ground for security breach

- Web application security is massively complex in reality
- Security researchers specialize in specific portions of the stack
    - Difficult for specialist security practitioners
    - Impossible for developers
- Protocols and specs exist but aren't implemented uniformly
- Constantly evolving field; Even platforms are changing
    - Think mobile, tablets, embedded systems, etc.
    - HTML 5, Web 2.0, AJAX, etc., etc.

Fertile Ground for security breach

- Complexity is the enemy of security
- Given all this complexity it is reasonable to assume

"Security vulnerabilities are rampant"

HTTP Request - GET

- Form data encoded in the URL
- Most common HTTP method used on the web
- Should be used to retrieve information, not for actions that have side-effects

HTTP Requests - POST

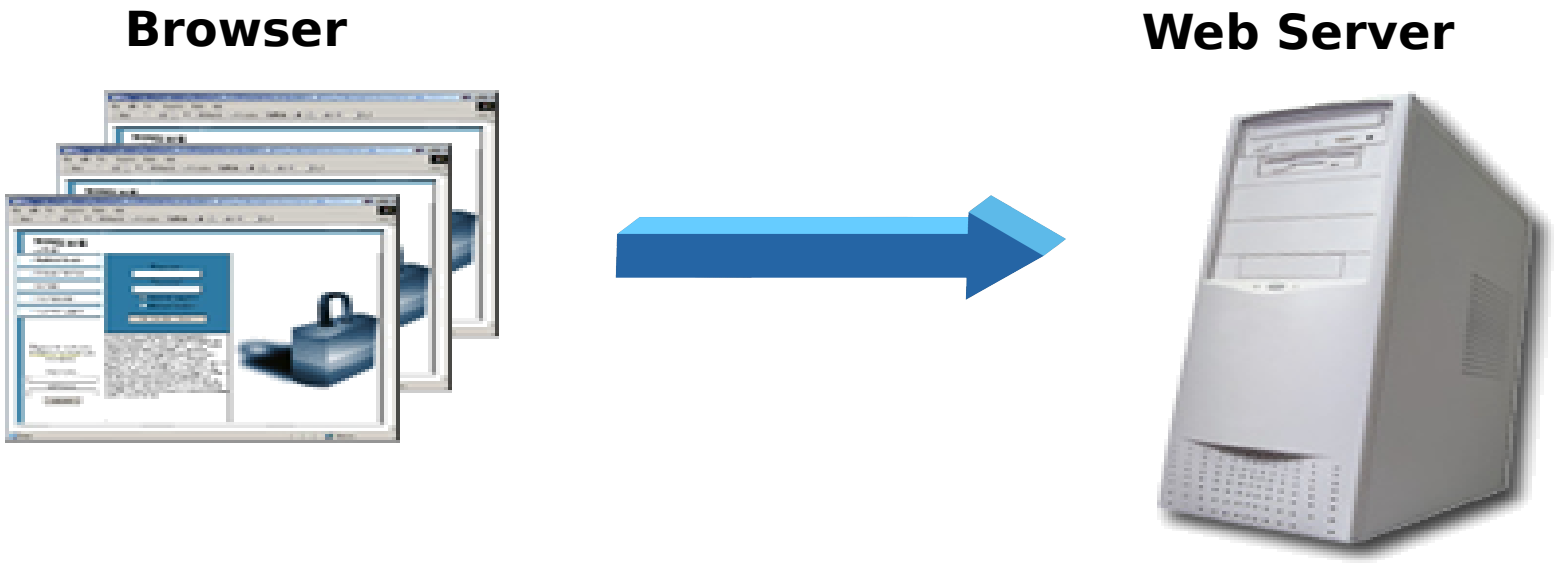
- Data is included in the body of the request.
- Should be used for any action that has side-effects
  - Storing/updating data, ordering a product, etc…

GET v. POST Security

- There information contained in parameters can tell a user a lot about how your application works
- **GET** parameters are easily visible in the address bar
- **POST** parameters are hidden from the *average* user
  - Users can still view source code
  - Users can still view the packets
  - Users can still intercept & modify web requests

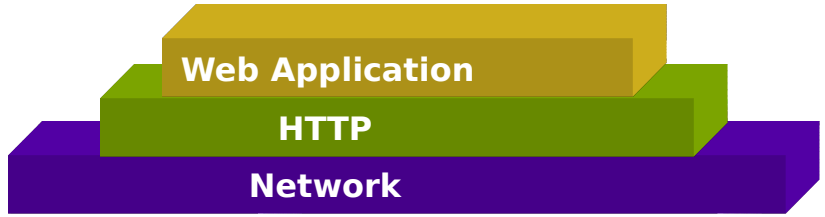
Web Sites

- **No applications**
- **Static pages**
- **Hard coded links**



Web Applications

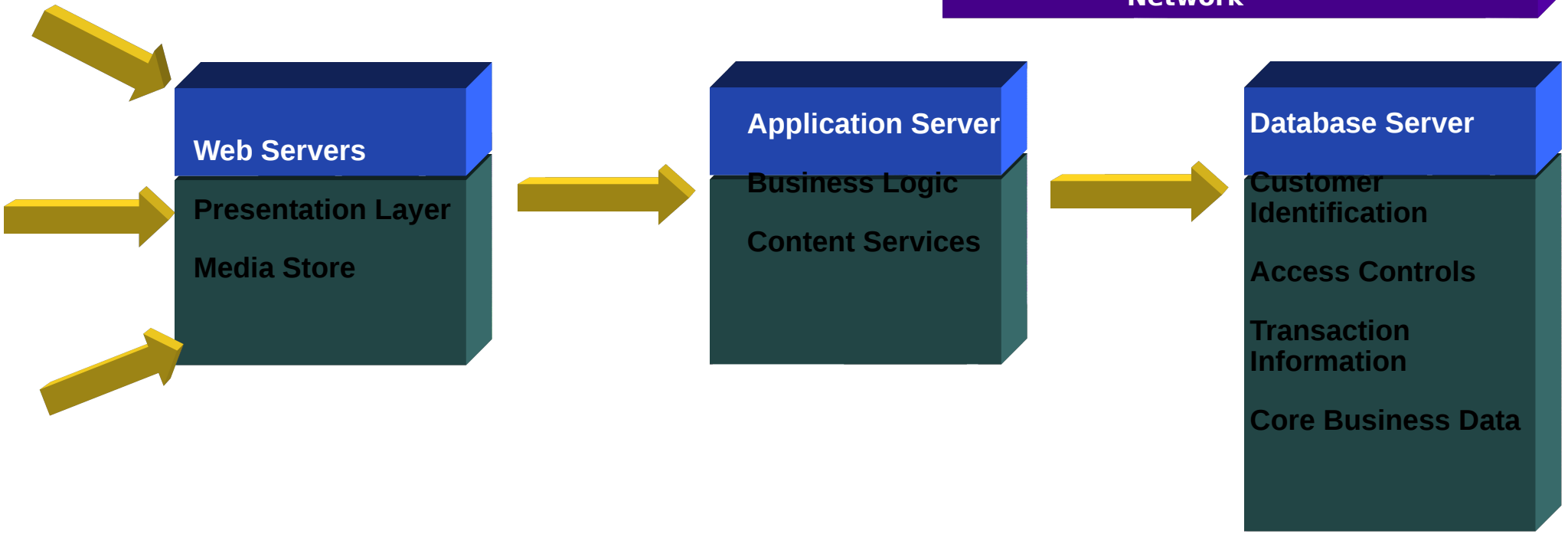
Very complex architectures, multiple platforms, multiple protocols



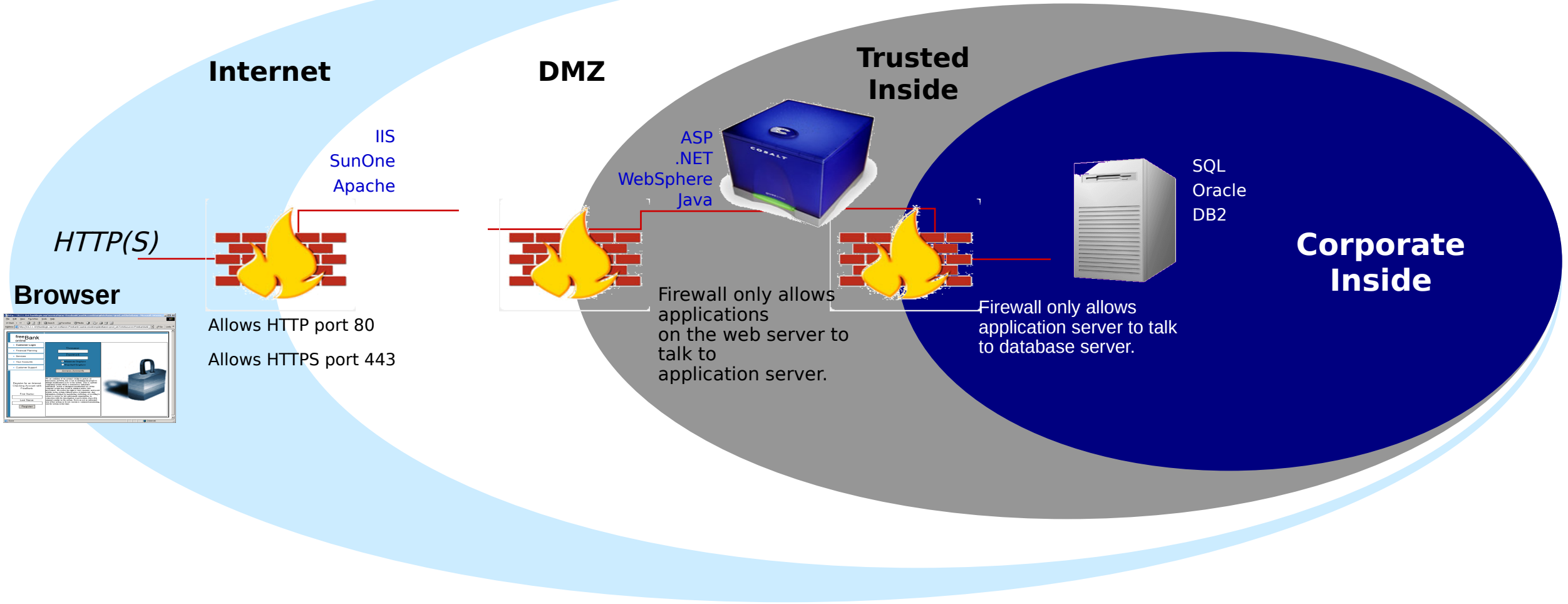
Web Services

Wireless

Browser



Web Applications Breach the Perimeter



Why Web Application Vulnerabilities Occur

Security Professionals Don't Know The Applications

"As a Network Security Professional, I don't know how my companies web applications are supposed to work so I deploy a protective solution…but don't know if it's protecting what it's supposed to."

Application Developers and QA Professionals Don't Know Security

"As an Application Developer, I can build great features and functions while meeting deadlines, but I don't know how to develop my web application with security as a feature."

The Web Application Security Gap

Web Application Vulnerabilities

“If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization.”

-Weinberg's Second Law

Security-CIA

What is it?

- CIA = confidentiality, integrity, and availability
  - Considered most crucial components of security
- Guide policies within an organization
- Confidentiality – limits access to information
- Integrity – assurance that the information is trustworthy and accurate
- Availability – definite and reliable access to the information by authorized people

Computer and Network Assets, with Examples of Threats

	Availability	Confidentiality	Integrity
Hardware	Equipment is stolen or disabled, thus denying service.		
Software	Programs are deleted, denying access to users.	An unauthorized copy of software is made.	A working program is modified, either to cause it to fail during execution or to cause it to do some unintended task.
Data	Files are deleted, denying access to users.	An unauthorized read of data is performed. An analysis of statistical data reveals underlying data.	Existing files are modified or new files are fabricated.
Communication Lines	Messages are destroyed or deleted. Communication lines or networks are rendered unavailable.	Messages are read. The traffic pattern of messages is observed.	Messages are modified, delayed, reordered, or duplicated. False messages are fabricated.

Security CIA

Security-AAA

AAA Model—Network Security Architecture

- **Authentication**
  - **Who are you?**
  - **"I am user student and my password validateme proves it."**
- **Authorization**
  - **What can you do? What can you access?**
  - **"User student can access host serverXYZ using Telnet."**
- **Accounting**
  - **What did you do? How long did you do it? How often did you do it?**
  - **"User student accessed host serverXYZ using Telnet for 15 minutes."**

AAA

stands for

Authentication, Authorization, and Accountability

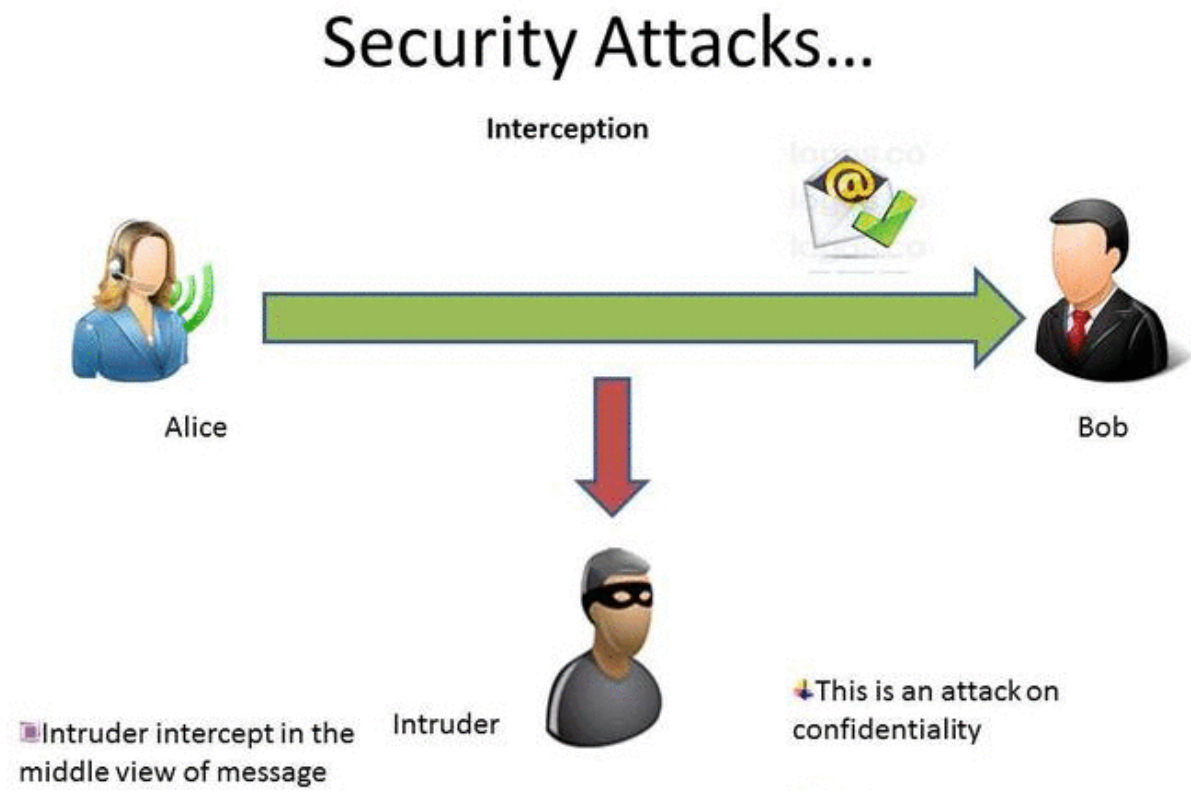
Abbreviations.com

Confidentiality

(make sure information is not accessed by unauthorized person)

- Authentication – process of verifying who you are
- Authorization -process of verifying what you have access to
- Secure communication

Man-in-the-Middle (MITM) attacks



MITM ATTACK

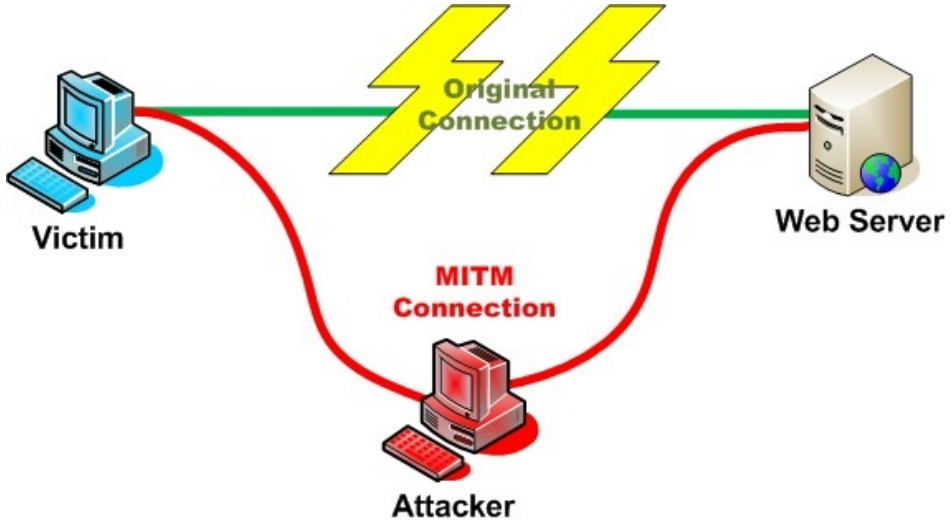
1. Fabrication: an attacker intercepts, impersonating one of the end points

- Can impersonate client: Bob thinks he is talking to Alice but it MITM
- can impersonate server : Alice thinks she is talking to Bob but it is MITM

2. Interception: an attacker can sniff the data (confidentiality)

3. Modification: an attacker can modify the data (integrity)

4. Interruption: an attacker attack to make server inaccessible (availability)



MITM ATTACK(HOW?)

Hack the wifi connection or break into the network of your Internet provider.

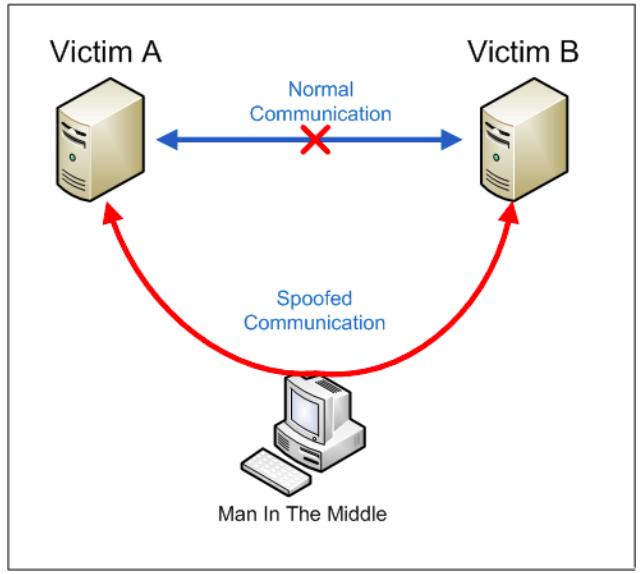
A. DNS spoofing : redirect to his malicious page by Hacking the router and changing the dns

B. ARP spoofing/ARP injection : Spoofs the router and makes you connect directly to him.(Address Resolution Protocol (ARP) is used for mapping a network address to a physical address)

https://www.quora.com/What-is-ARP-and-ARP-spoofing

C. Hack the Wifi connection

D. MITM device:Physically unplug a cable somewhere and insert a device in the middle



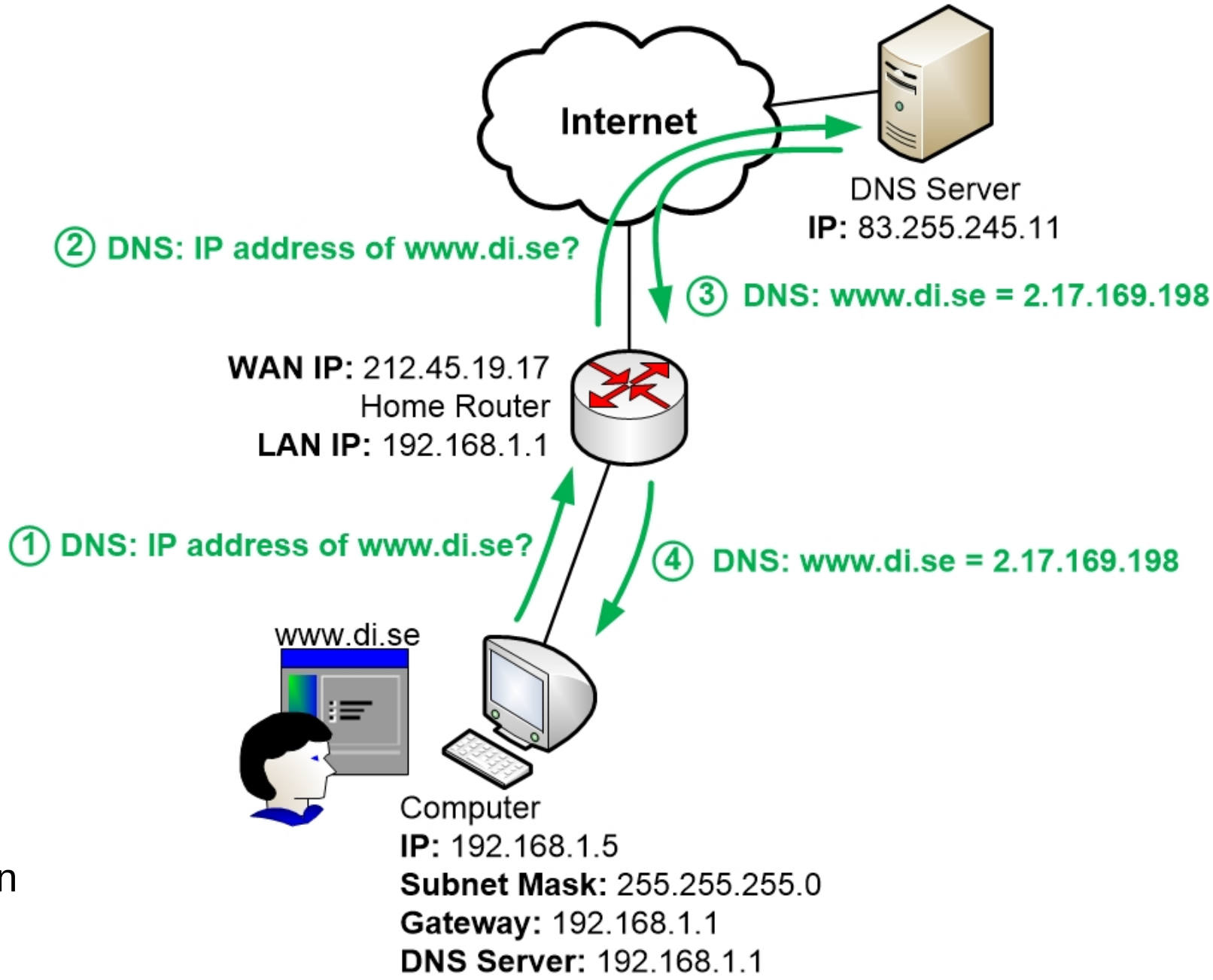
Router And DNS

1. IP address of www.di.se?
From client to router through LAN/WAN

2. From router to ISP's DNS server through Internet

3. reply from ISP's DNS server to router trough Internet

4. local router responds to client machine and caches the information to speed things up in future.



Major security issues

- To prevent unauthorized users from accessing sensitive data,
  - Authenticate on both endpoints of communication
- To prevent attackers from stealing data from network during transmission, use
  - Encryption (usually by SSL - Secure Sockets Layer)

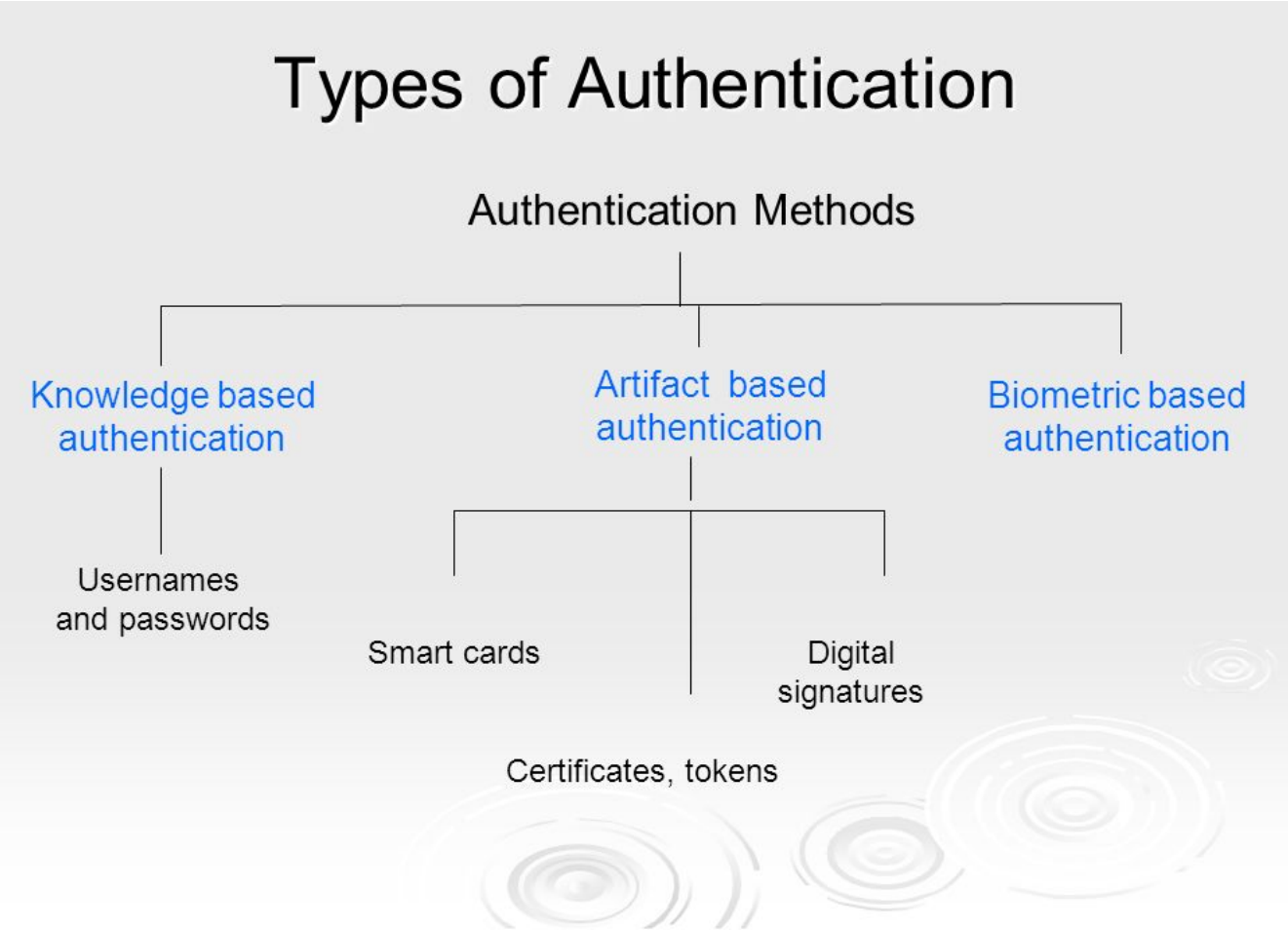
Authentication

- **Identity verification: How can Alice be sure that she is talking to Bob?**

- **1. <u>What-you-know</u>**
  passwords, PIN, photo

- **2. <u>What-you-have</u>**
  token, OTP cards, ATM card

- **3. <u>Who-you-are</u> (e.g. Biometrics)**
  - physiological:
  face, finger, iris
  - behavioral:
  handwriting, gait, speech,
  signature, digital signature



Authentication (knowledge based authentication: client side)

- Collect user ID and password information from end users (“logging in”)
  - usually by means of browser dialog / interface
  - user ID information normally refers to username and password
- Securely transport collected user ID information to the web server
  - unsecurely (HTTP) or securely (HTTPS = HTTP over SSL)
- Verify ID and password with backend Realms (“security database”)
  - Realms maintain username, password, roles, etc.
  - Validation: the web server checks if the collected user ID & passwd match with these in the realms.
- Keep track of previously authenticated users for further HTTP operations

Authorization(ACL Models)

Mandatory AC

- System decides exactly who has access to which resources.
- Centrally managed by the security policy administrator

Discretionary AC

- Users can grant/deny other users privileges to the resources that they create/use/own (e.g. UNIX).

Role-Based AC / a.k.a. Non-Discretionary

- Privileges are determined by a user's role (e.g. admin, manager)
- Assigns permissions to operations with meaning in the organization.

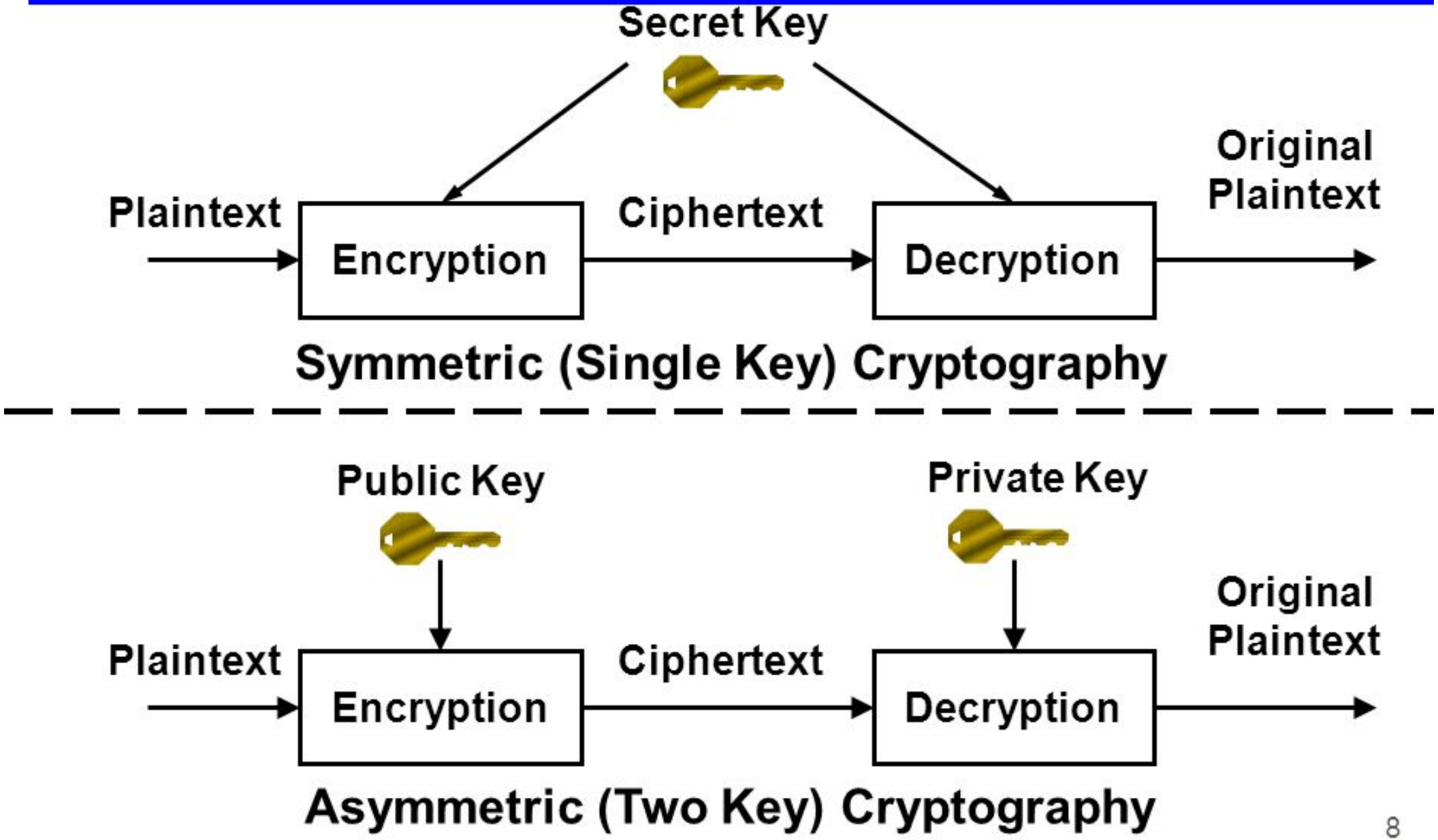
Secure Net: What is Secure Sockets Layer (SSL)?

- A protocol developed in 1996 by Netscape for securely transmitting private web documents over the Internet.
- It transmits sensitive data online, such as credit card information, ID #, login info(username and password) etc.
- It employs private and public key to encrypt data that's transmitted over the SSL connection.
- By convention, URLs that require SSL connection start with *https:* (port 443) instead of *http:* (port 80).

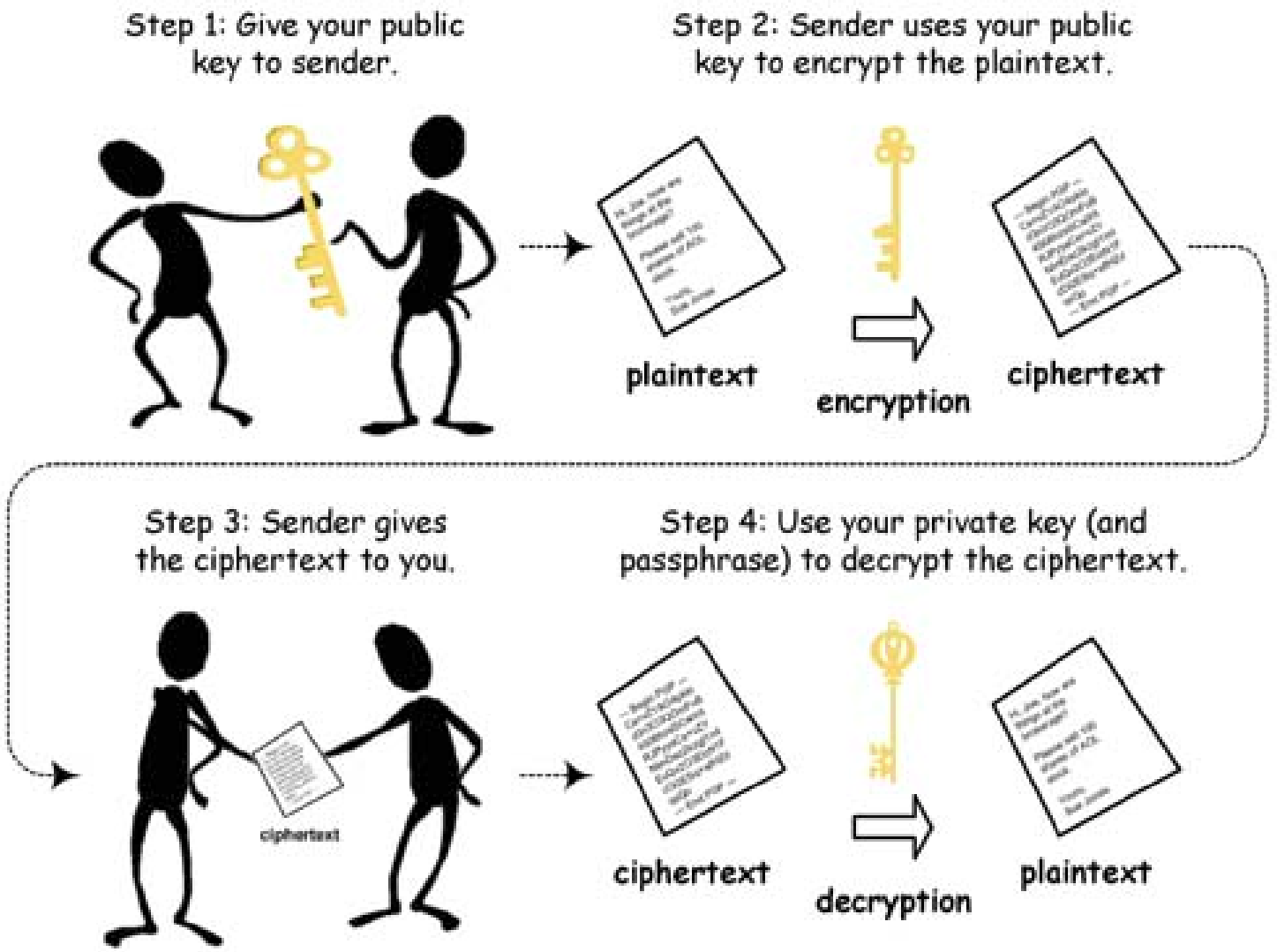
How does HTTPS work: SSL explained

This presumes that SSL has already been issued by SSL issuing authority.

Encryption (single vrs. Double keys)

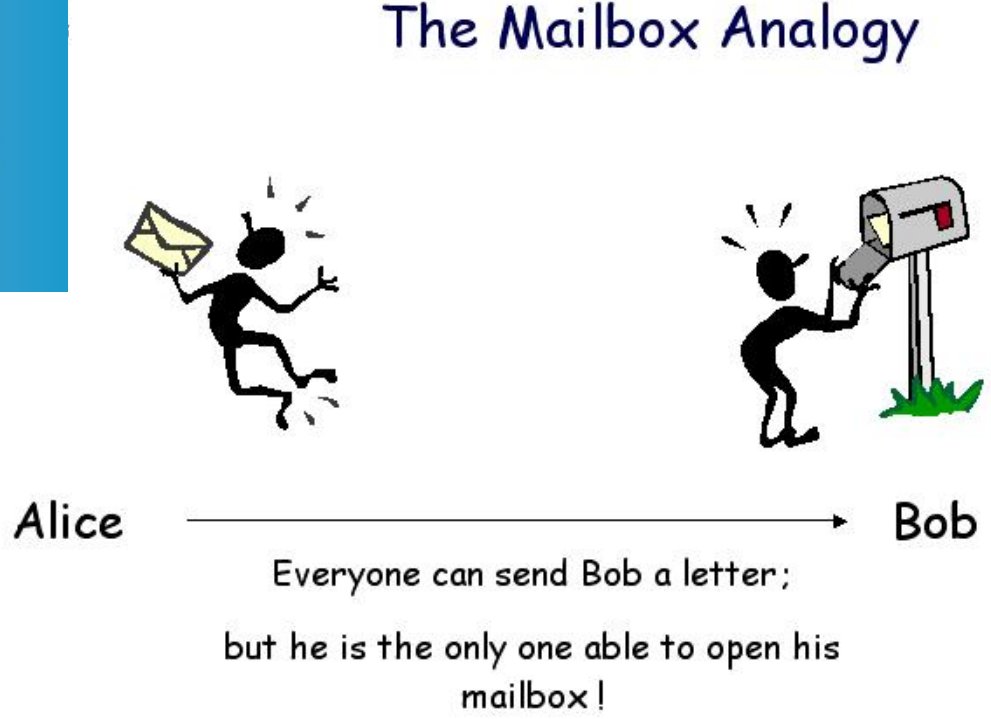


Encryption (between sender and receiver)

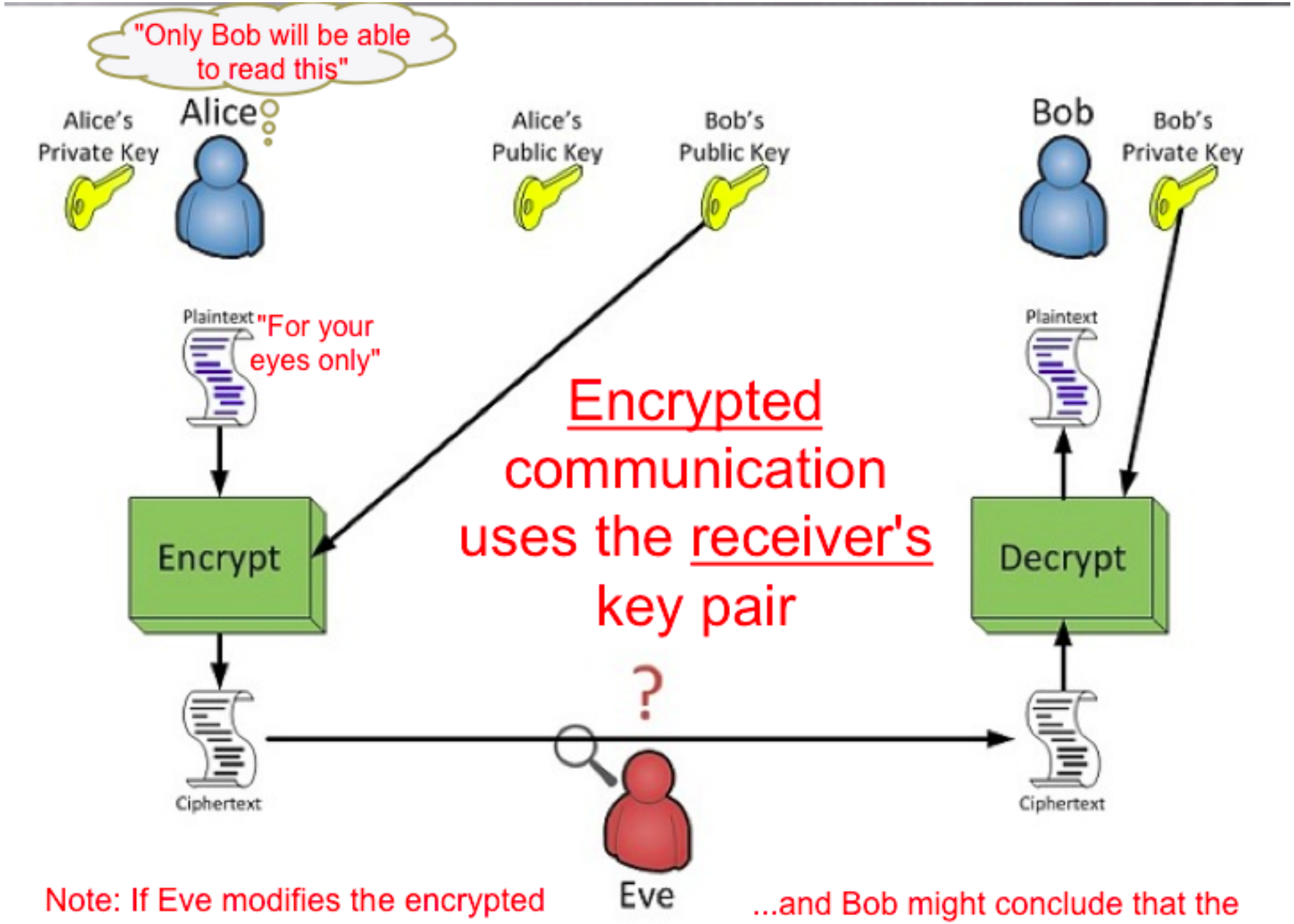


Encryption for Confidentiality and Integrity

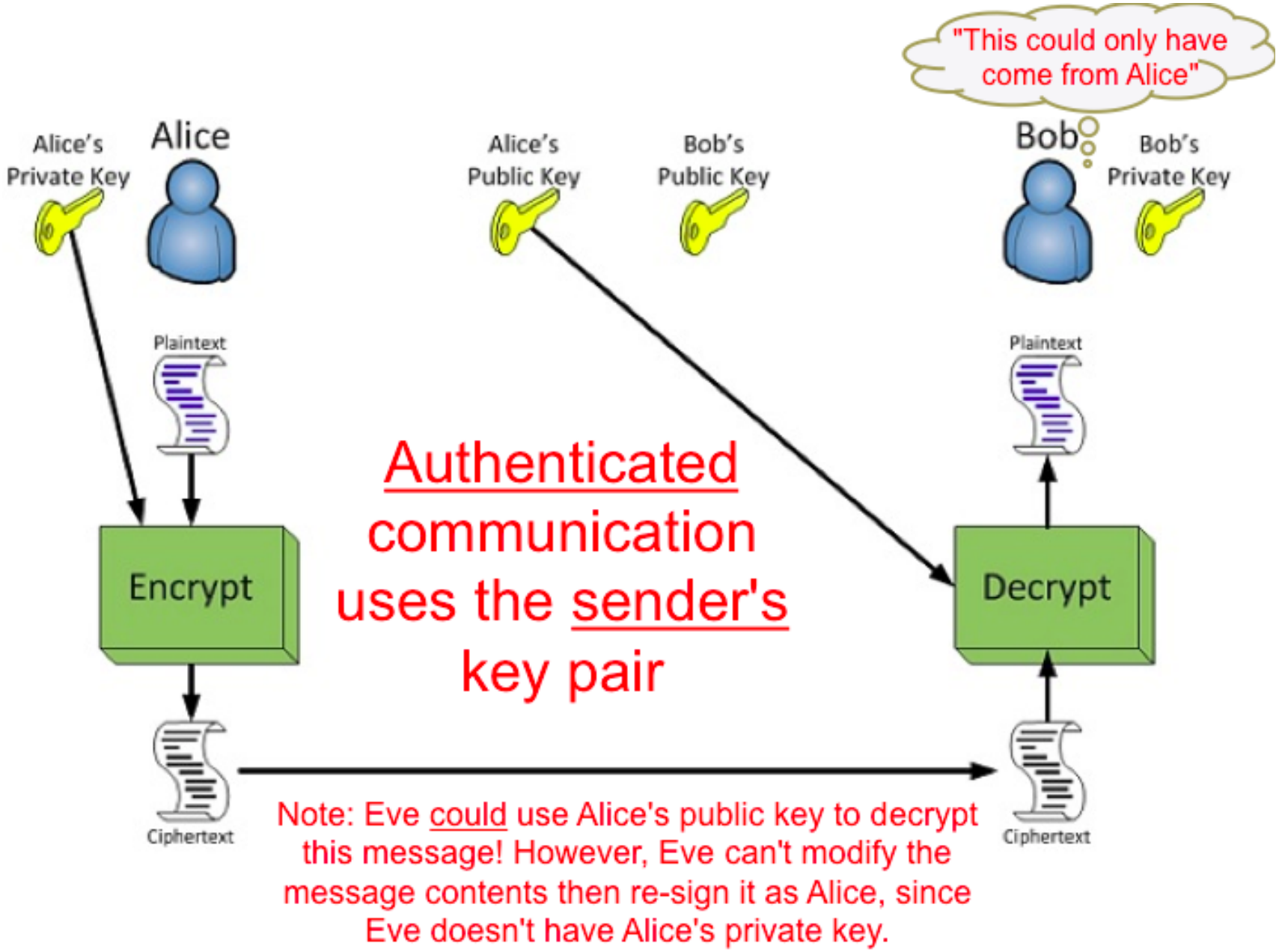
- Asymmetric encryption use two keys:
  - Public Key - to encrypt the data
  - Private Key - to decrypt the data
- These keys are generated together.
- The Public key(s) is distributed freely between the sender and receiver.
- The other is named as Private Key and it is kept hidden.
- The Private Key is only used for Decryption and will not be shared between the sender and receiver.



Encryption (Message from Alice to Bob)



Encryption **(Messages from Alice being authenticated)**



Encryption (signature creation and verification

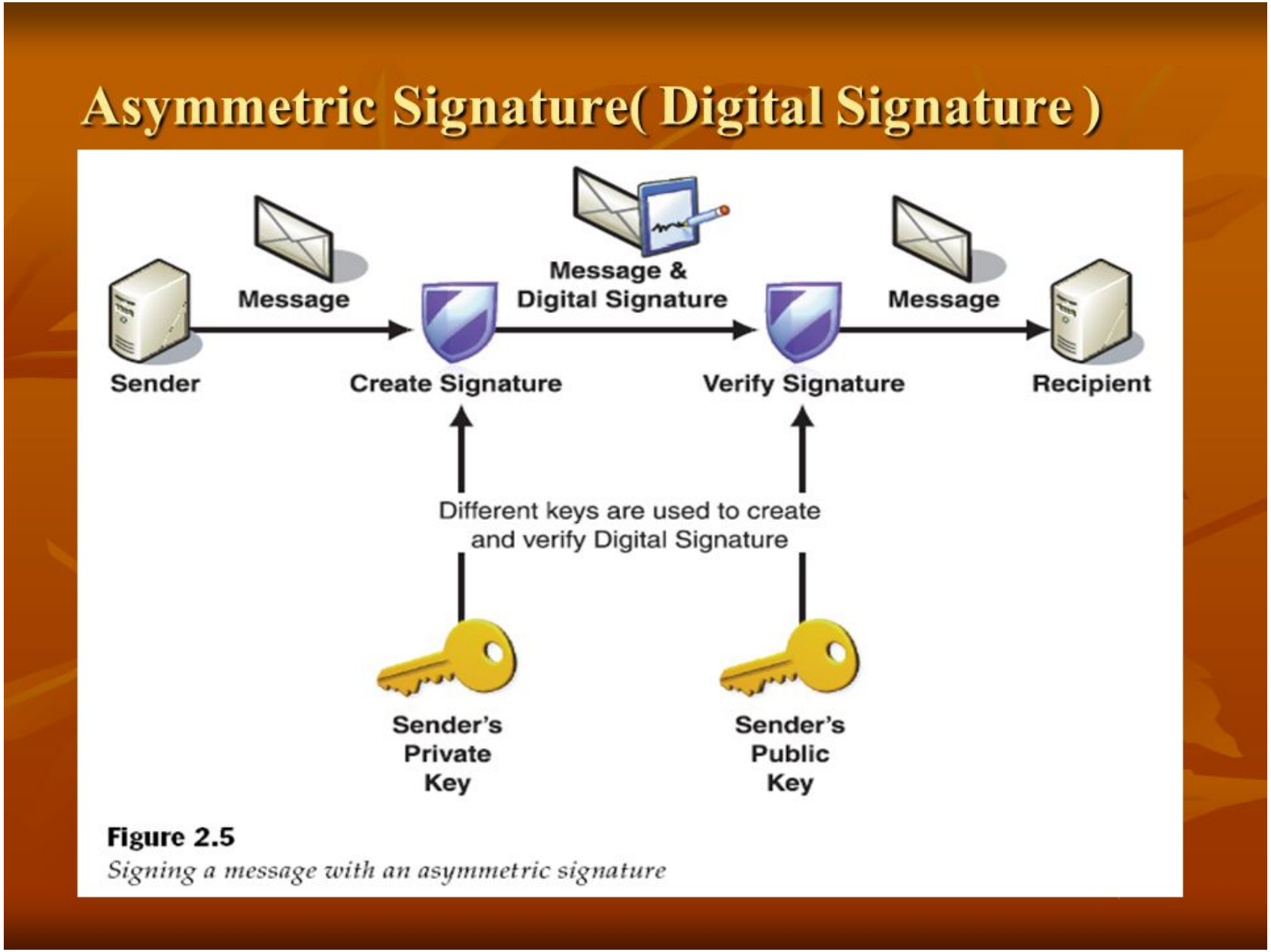
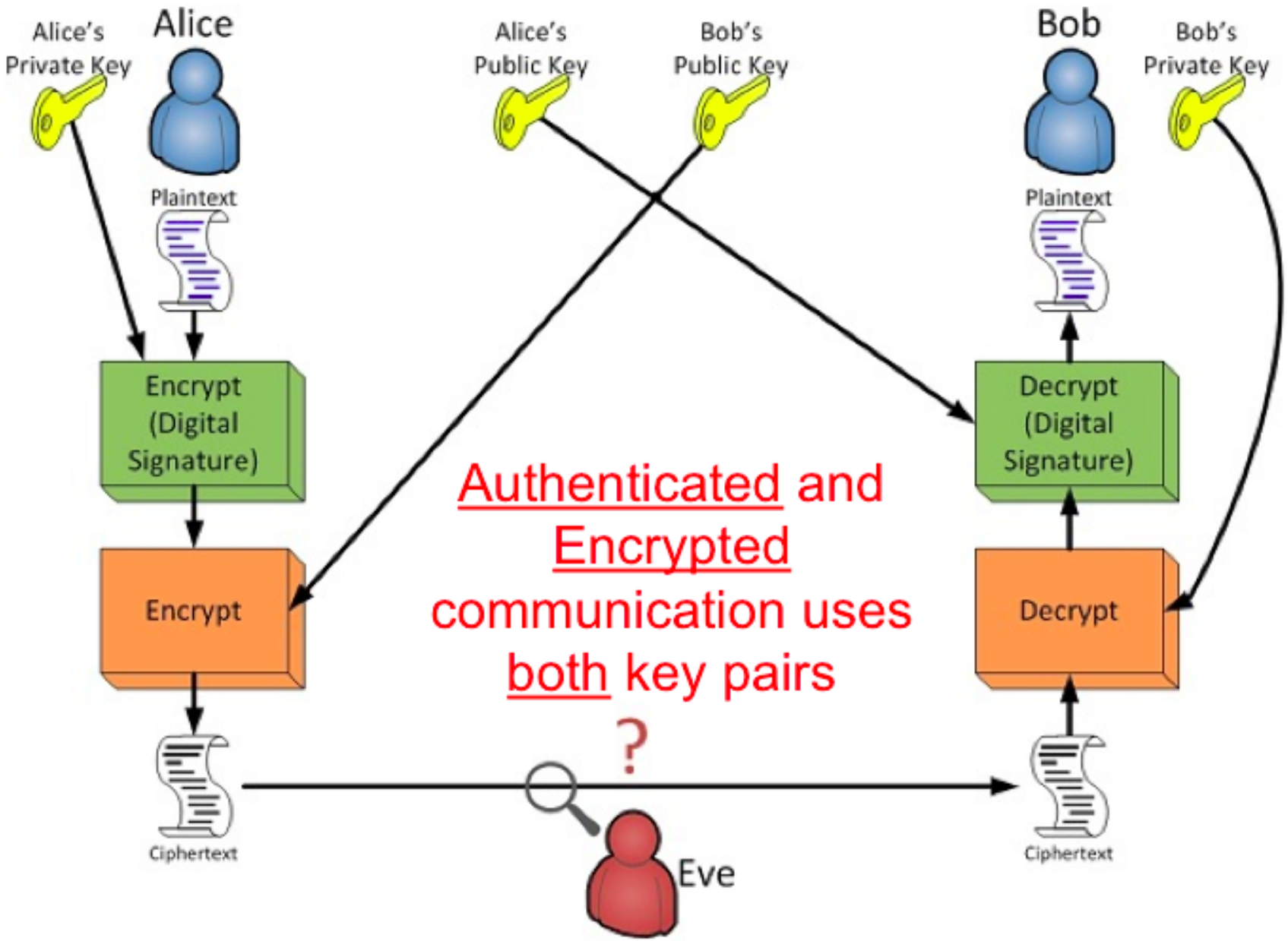


Figure 2.5

Signing a message with an asymmetric signature

Encryption (**(Messages from Alice being authenticated and encrypted)**



Encryption(summary)

- Alice produces key pair
  - private and public
  - private key is only known to Alice.

- Alice uses his private key to
  1. create his digital signature on a message
  2. decrypt messages coming from others

- Man-in-middle does not know about any private key, so
  1. can’t tamper and then spoof signatures on messages
  2. can’t decrypt messages received by others

- Bob uses Alice’s public keys
  to encrypt messages to be sent to Alice and
  to verify Alice’s signature in received message

Encryption(summary)

For authentication, sender’s key pair is used.

For secure message communication, receiver’s key pair is used.

- Sender

  uses *sender’s private* key for signing the message to be sent

  uses *receiver’s public* key to encrypt the message to be sent

Receiver

uses *sender’s public* key to verify the signature and authenticate the sender.

uses *receiver’s private* key for decrypting the message just received

Digital signature

Encryption on hash of the original data using private key of the signer

Digital signature ensures:

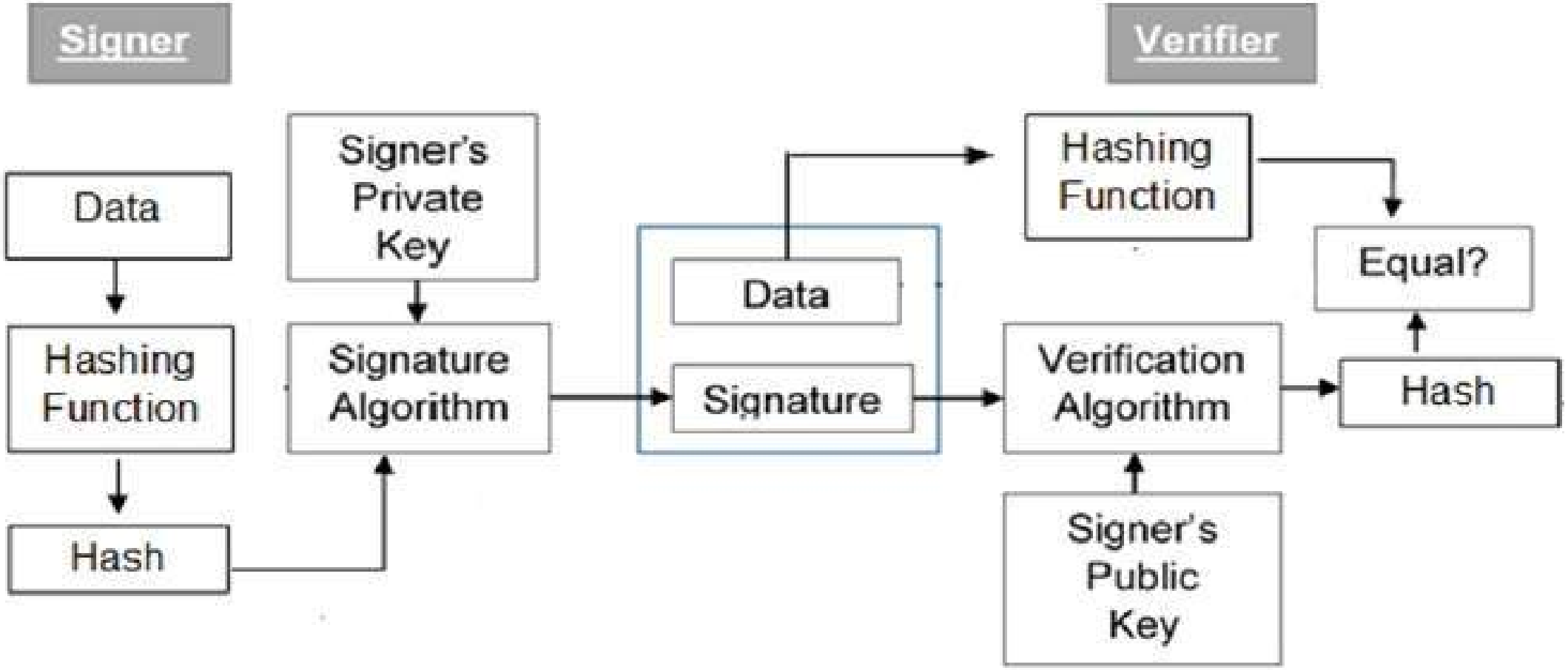
1. Data integrity (hash will change if original data is tampered with)
2. Message authentication (only signer has knowledge of signature key)
3. Nonrepudiation (since only signer has knowledge of signature key, signed document is the valid evidence in case of any dispute)

Nonrepudiation refers to the ability to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated.

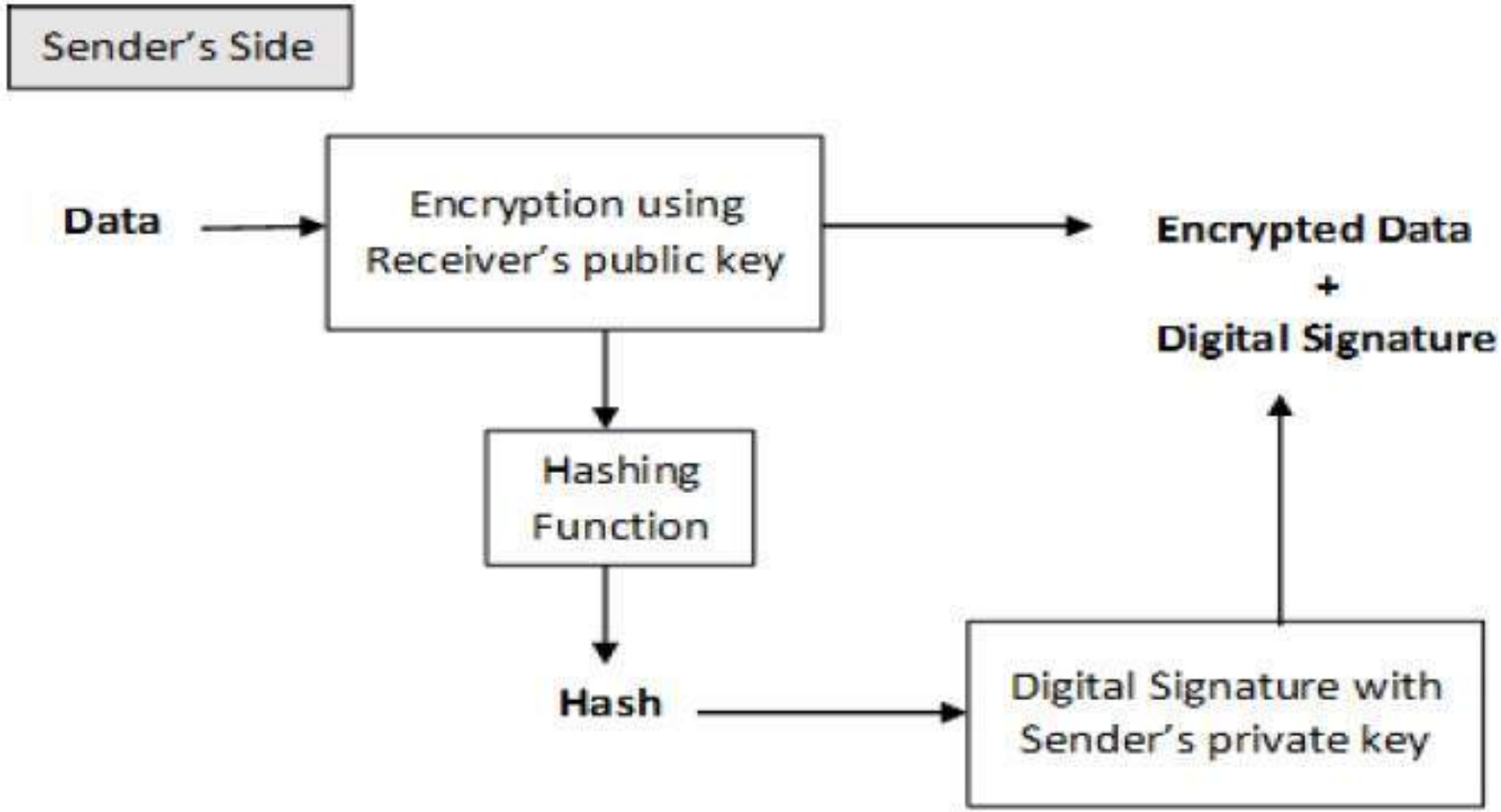
Along with the encryption scheme on data, 4^{th} property is added :

4. Confidantiality

Digital signature



Digital signature with encryption



Need for authentication at server side

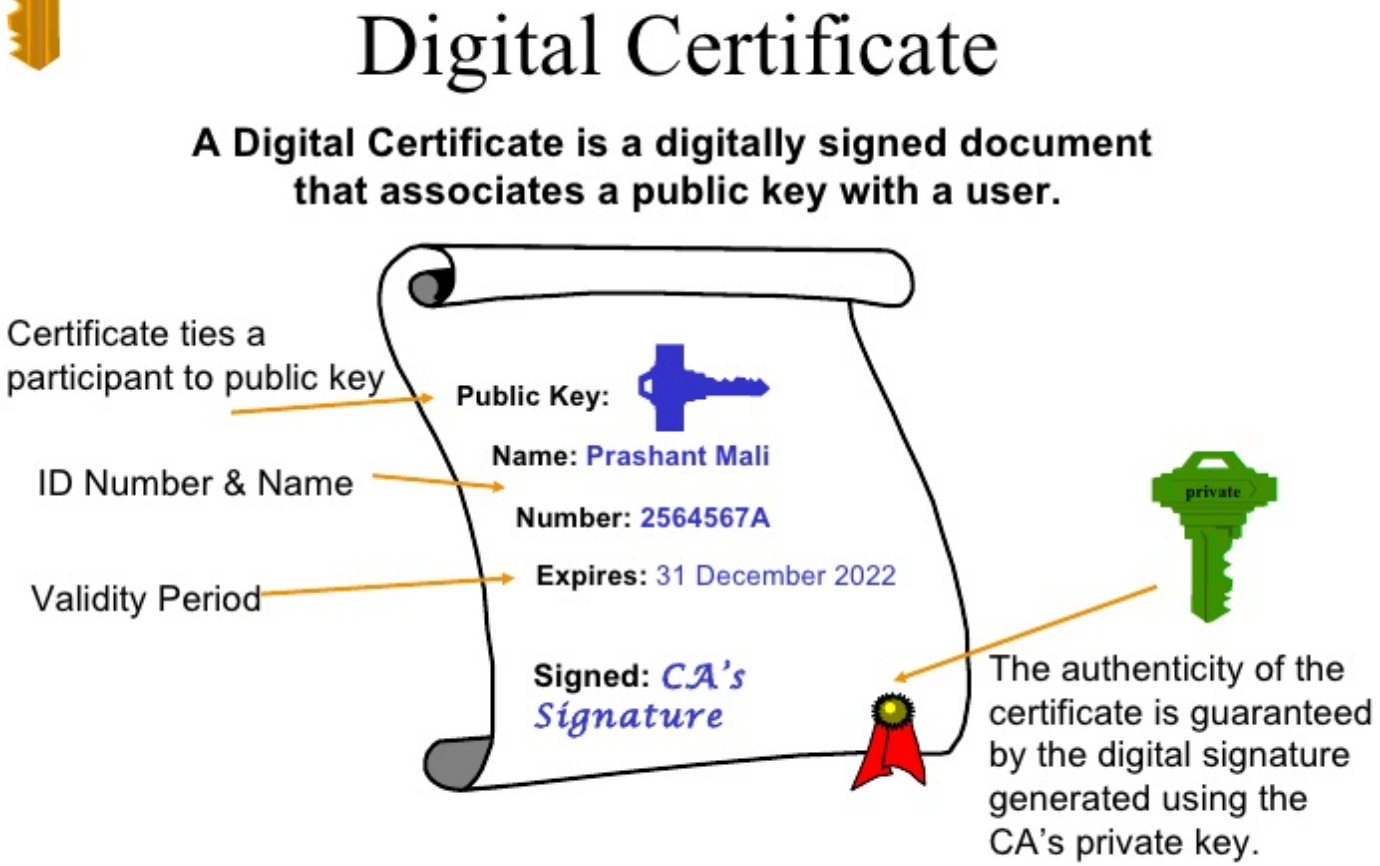
Man in Middle Attack (MIM) − The targets of this attack are mostly public key cryptosystems where key exchange is involved before communication takes place.

- Host A wants to communicate to host B, hence requests public key of B.
- An attacker intercepts this request and sends his public key instead.
- Thus, whatever host A sends to host B, the attacker is able to read.
- In order to maintain communication, the attacker re-encrypts the data after reading with his public key and sends to B.
- The attacker sends his public key as A’s public key so that B takes it as if it is taking it from A.

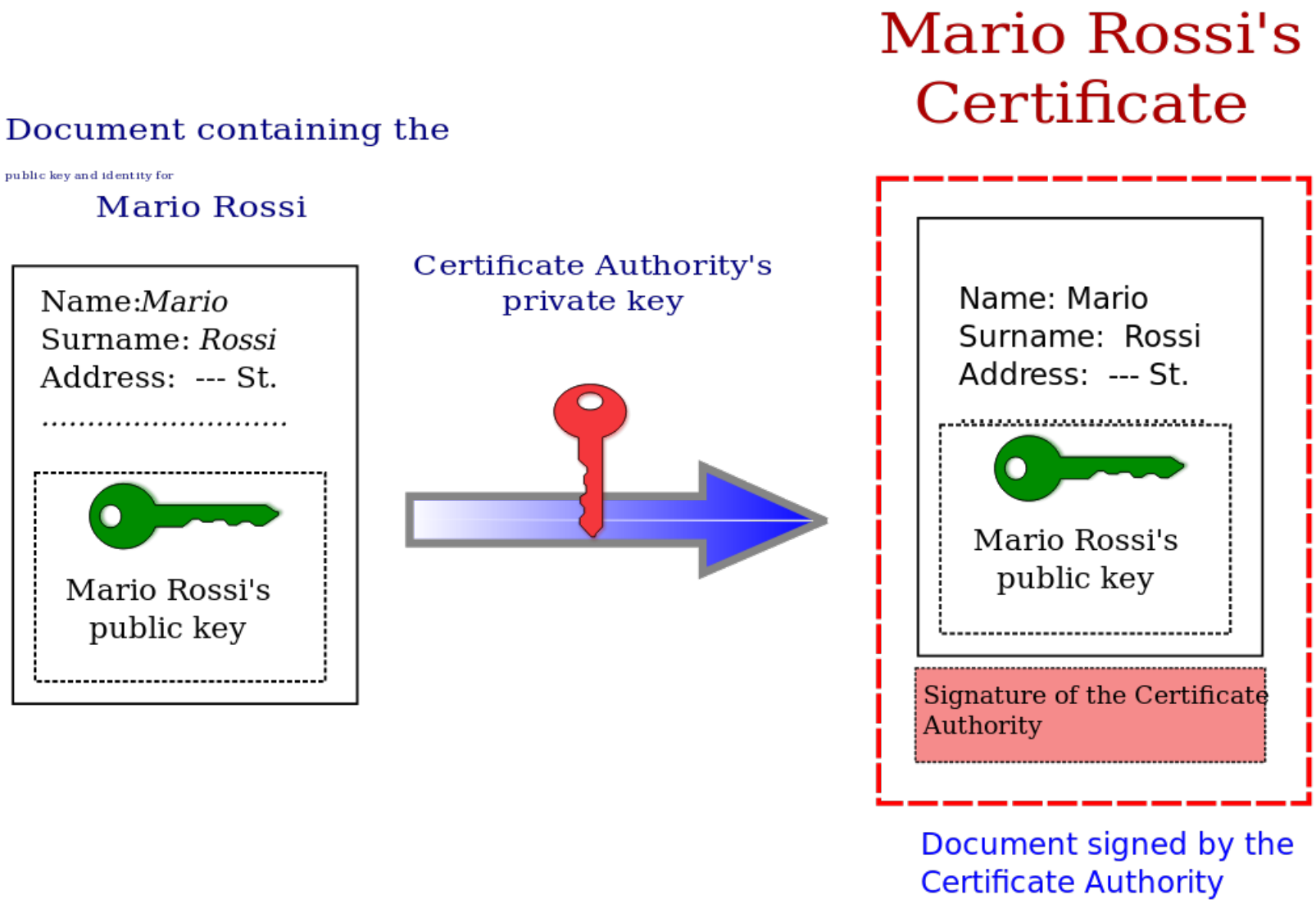
Server authentication with Digital Certificate

To enable secured SSL connections:

- the server needs an SSL certificate signed by a Certificate Authority (CA) which confirms that you are who you say you are in the Internet.
- Each SSL Certificate contains unique and authenticated information about the certificate owner, such as name, public key, and the signature of the CA.



A Sample certificate



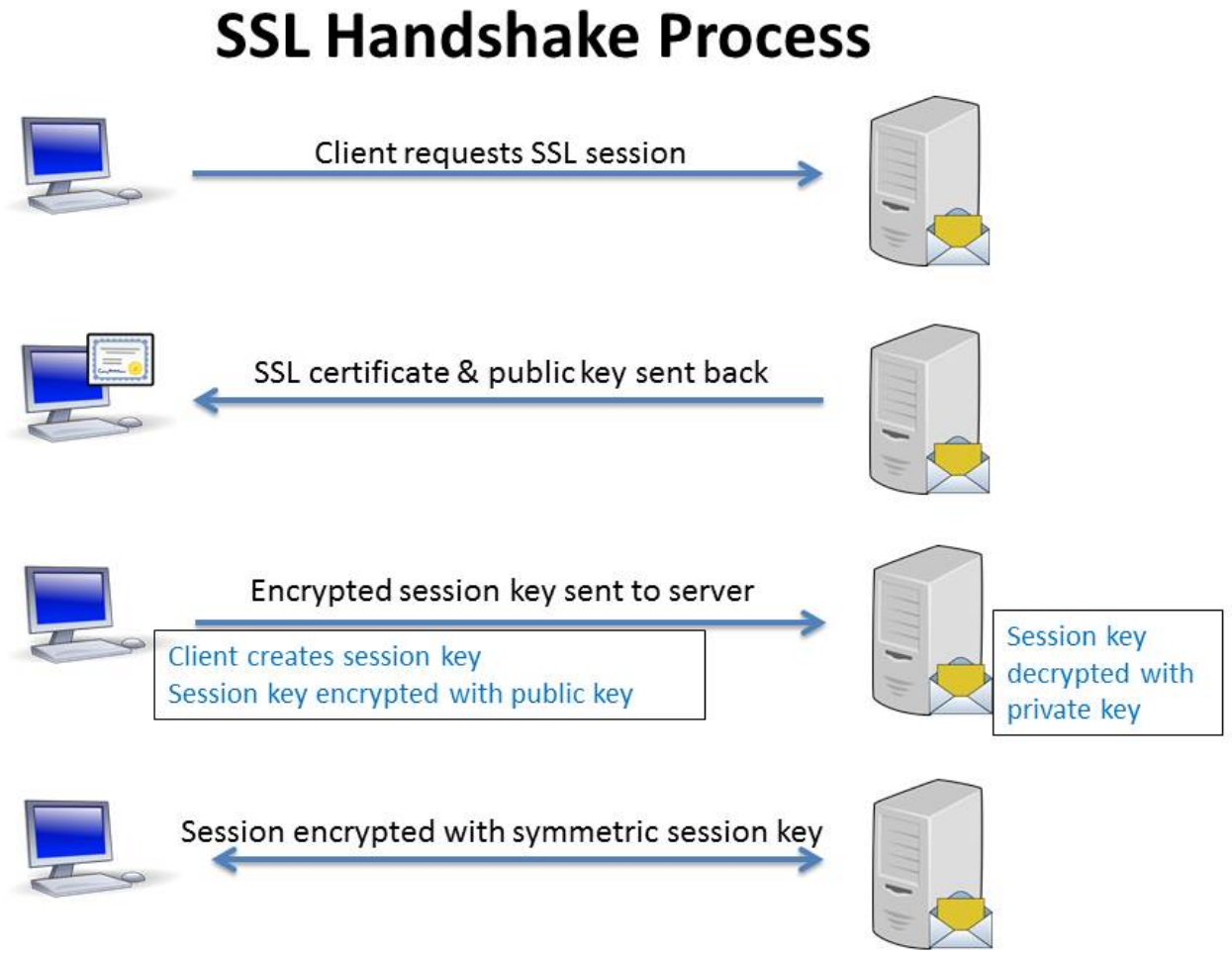
SSL - authentication (1 way)

Step1: SSL -handshake (not Diffie Hellman)

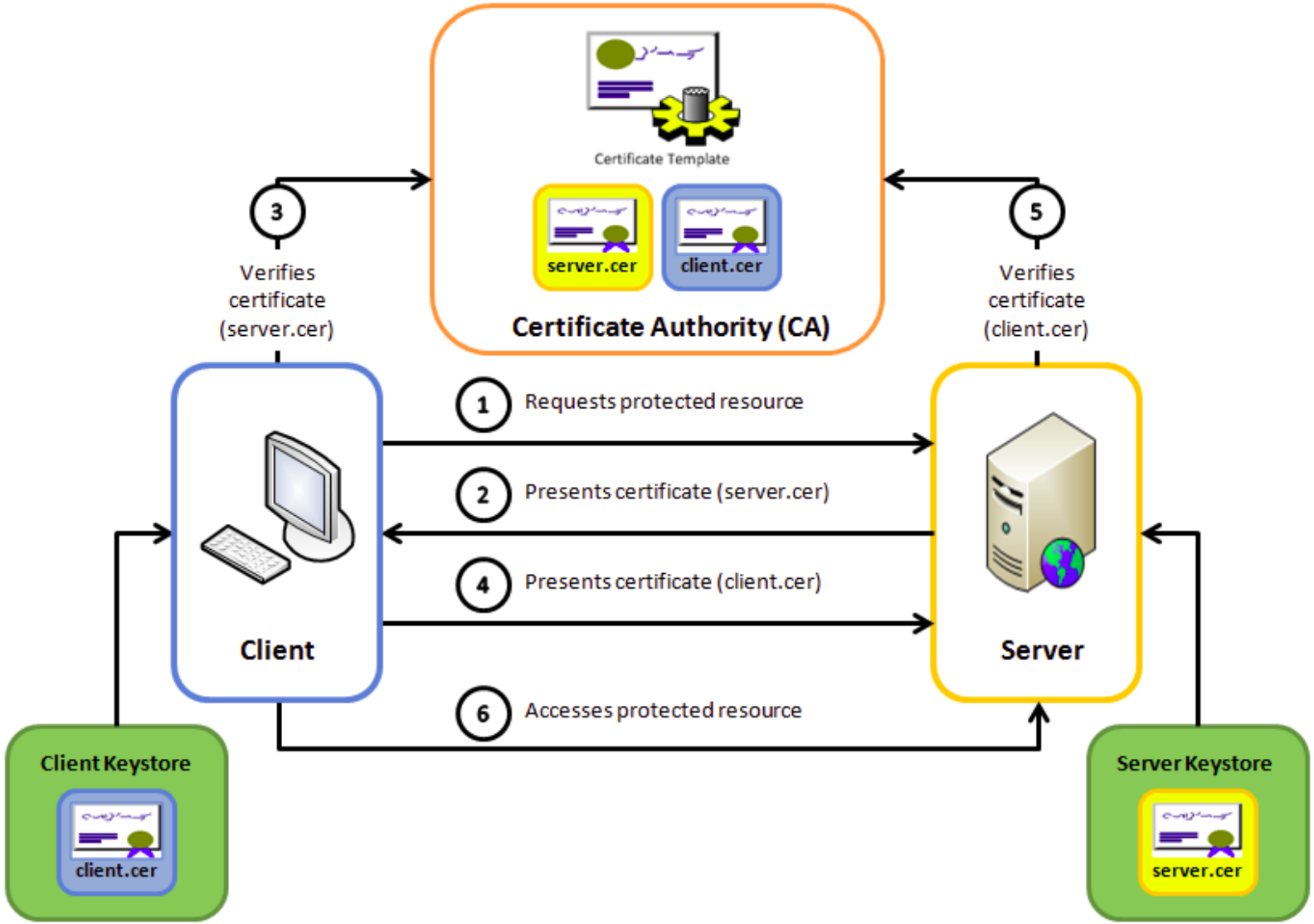
- Server sends a copy of its asymmetric public key.
- Browser creates and then encrypts the session key with the server's asymmetric public key. Then sends it to the server.
- Server decrypts the session key using its asymmetric private key to get the symmetric session key.

Step2: Secure Communication

- Server and Browser now encrypt and decrypt all transmitted data with the symmetric session key.
- This allows for a secure channel because only the browser and the server know the symmetric session key, and the session key is only used for that session.
- If the browser was to connect to the same server th next day, a new session key would be created.



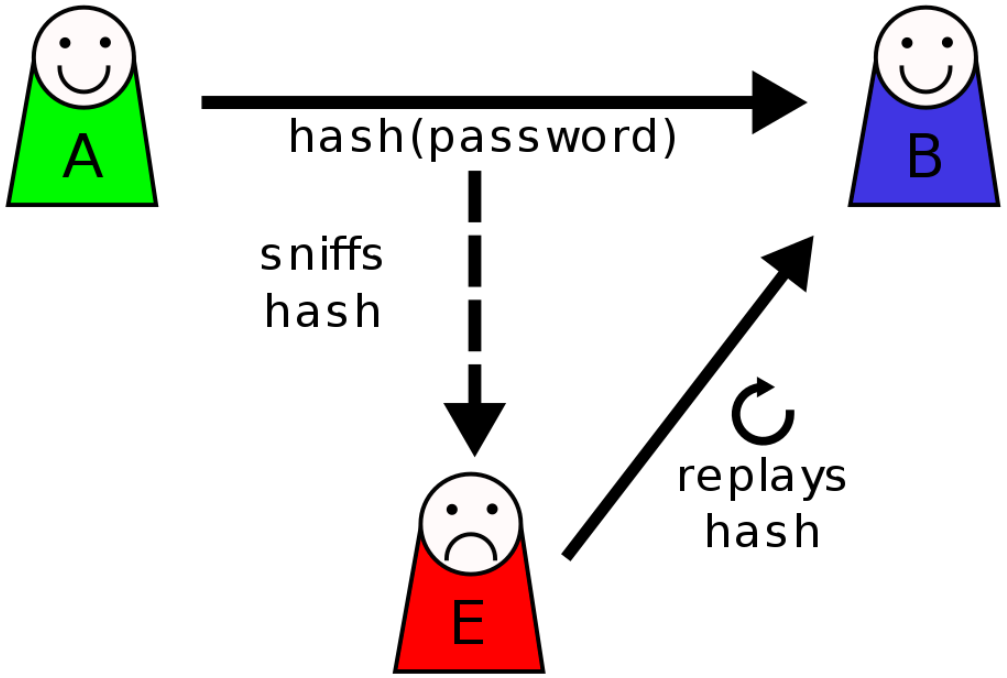
SSL -certificate based mutual authentication (2 ways)



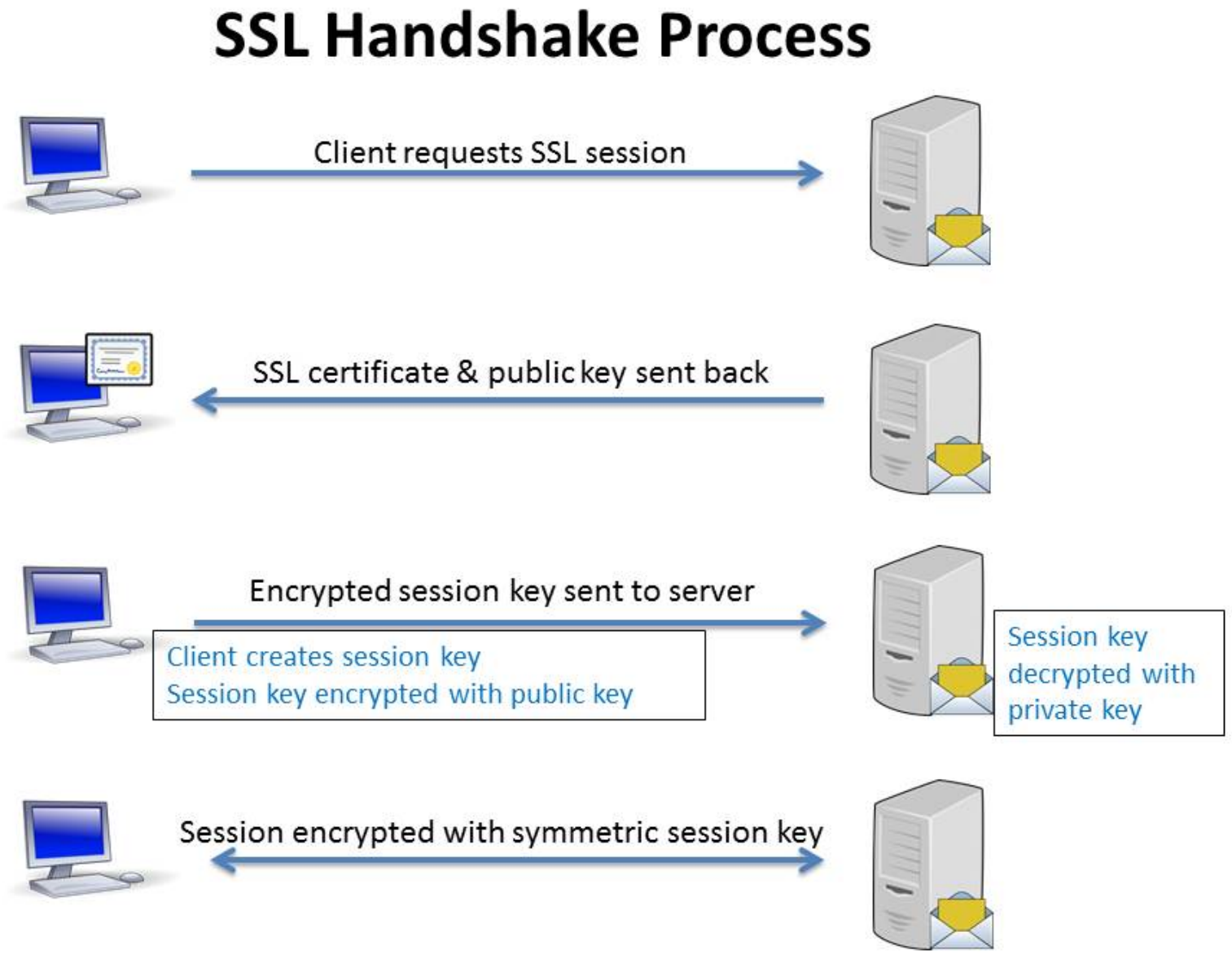
Mutual SSL authentication / Certificate based mutual authentication

MITM replay/playback attack

- Valid data transmission is maliciously or fraudulently repeated or delayed.
- Counter measures are one time expirable passwords, sequence numbers, timestamps or session id.



MITM session id:forward secrecy issues



MITM session id : forward secrecy

- Forward secrecy is designed to prevent the compromise of a long-term secret key from affecting the confidentiality of past conversations.

- The Heartbleed bug in vulnerable version of OpenSSL allows an attacker to extract data from the memory of a vulnerable server. This includes things like usernames, passwords, email addresses, session data and worst of all, the server's private key.

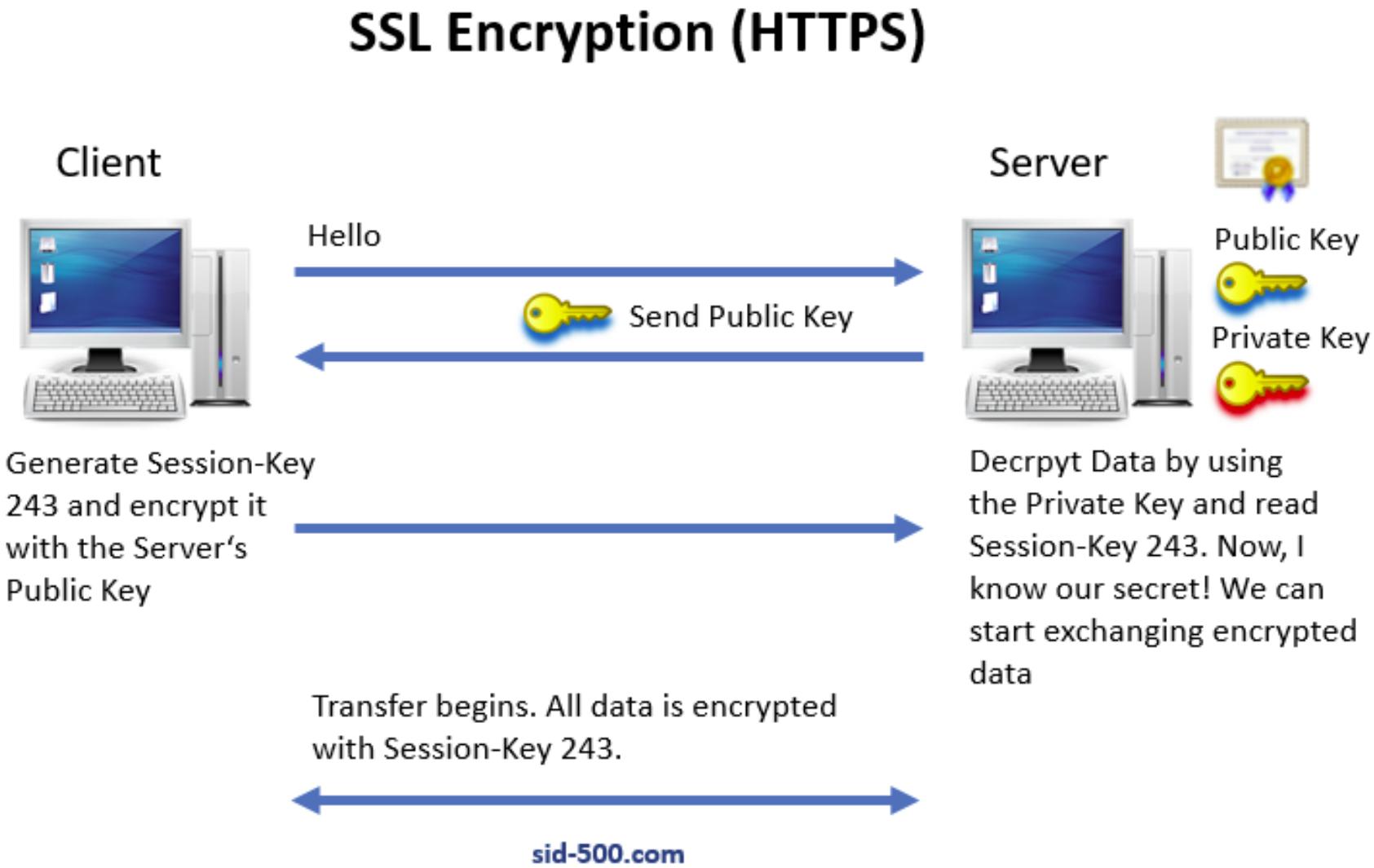
- If an attacker has been recording encrypted traffic and storing it, once they gain access to the private key, they can go back and decrypt all of the data that they have intercepted in the past.

- By generating a unique session key for every session a user initiates, even the compromise of a single session key will not affect any data other than that exchanged in the specific session protected by that particular key.

- Diffie-Hellman key exchange can be used to generate session keys. When you enable Perfect Forward Secrecy (PFS), there is no link between your server’s private key and each session key.

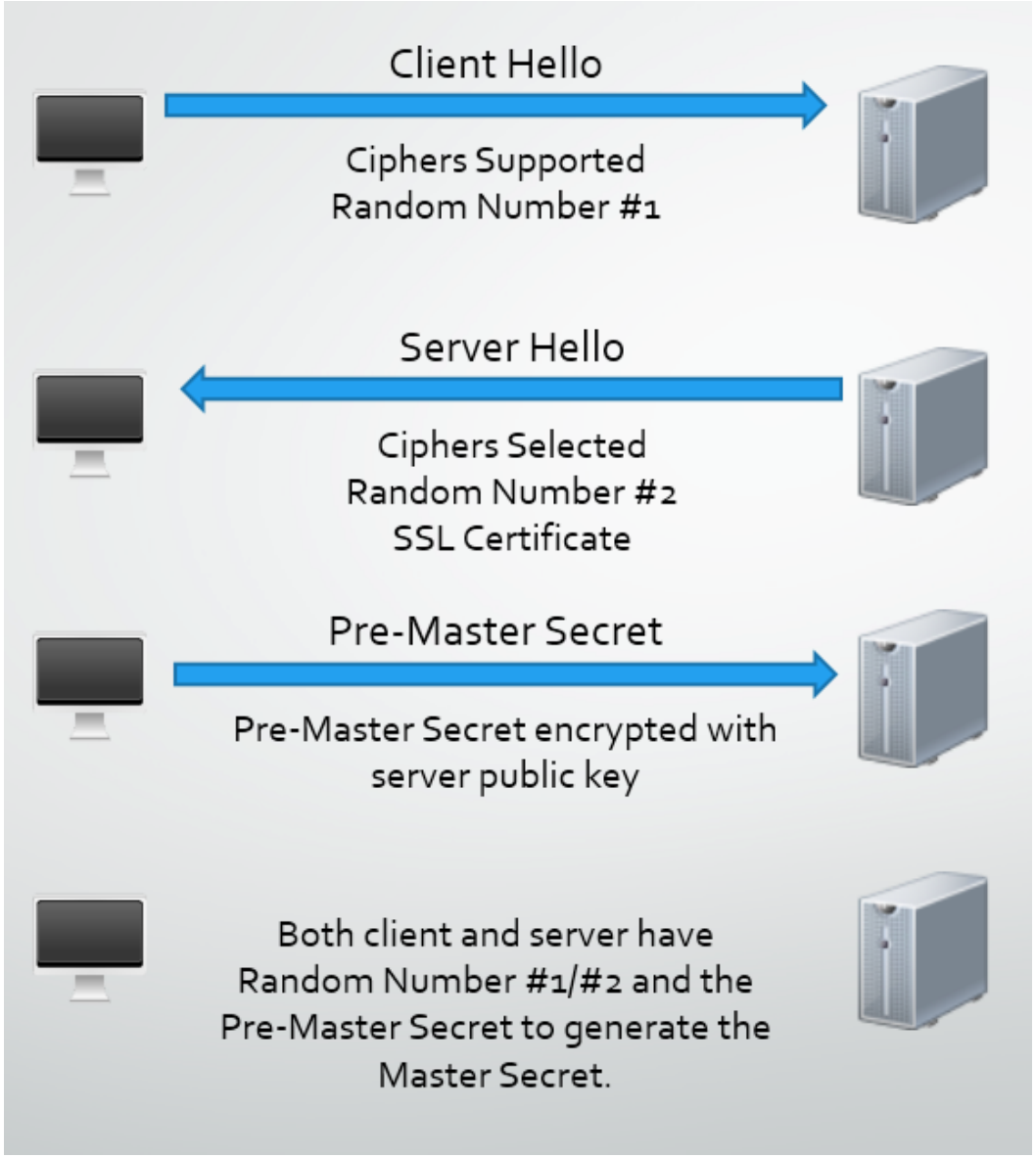
MITM session id : forward secrecy compromised!

What happends when the server’s private key is compromised.



MITM session id : forward secrecy compromised!

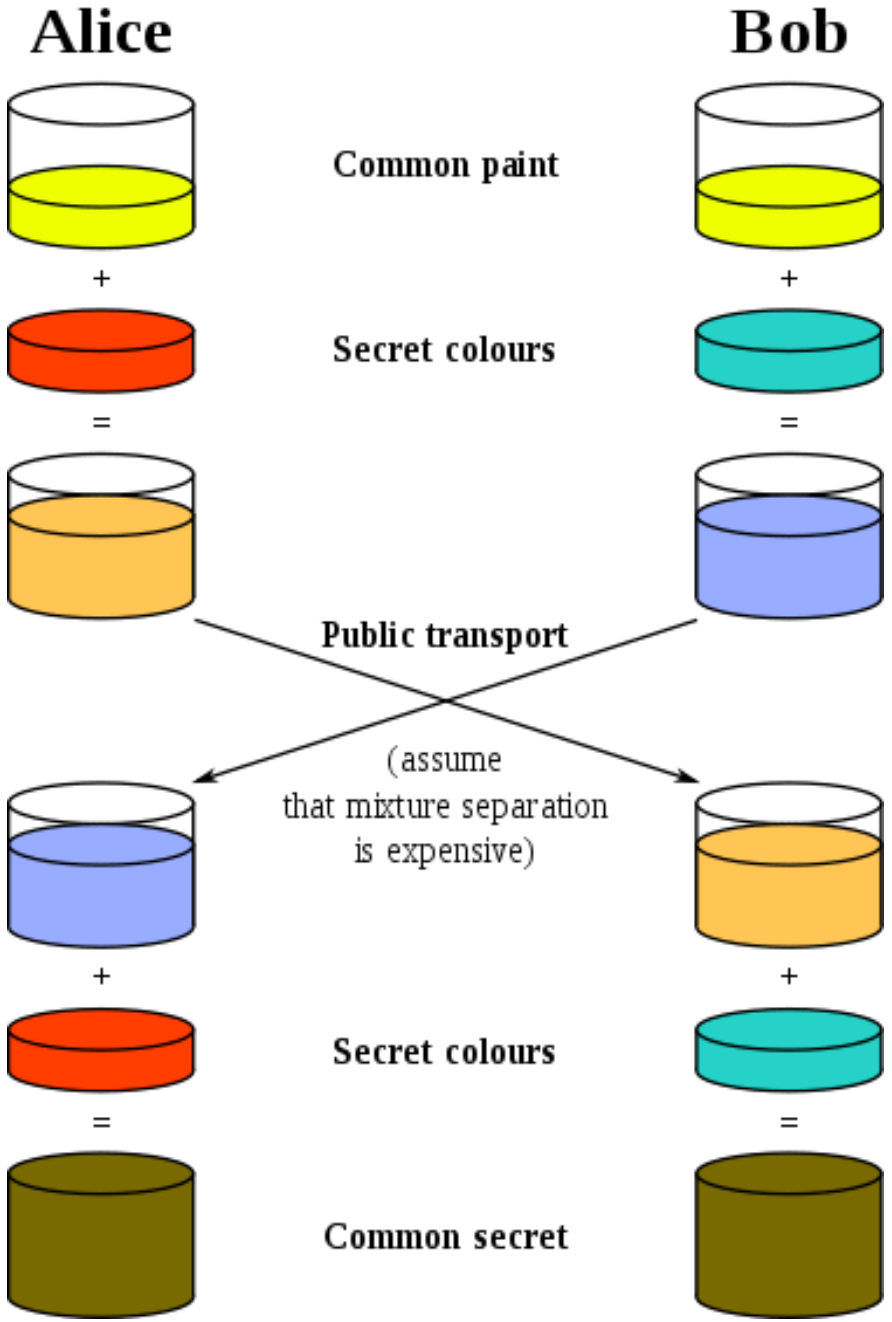
What happends when the server’s private key is compromised.



What happends when the server's private key is compromised.

MITM

Diffie-Hellman for forward secrecy



MITM session id : DHE - forward secrecy

Two parties that have no prior knowledge of each other jointly establish a shared secret key over an insecure channel.

1) Alice and Bob use pair of long-term keys to verify their public-key fingerprints and authenticate each other

2) Alice and Bob generate a ephemeral session key using a Diffie–Hellman algorithm.

3) Alice and Bob now communicate using symmetrical encryption scheme using keys.

The process repeats for each new message sent, starting from step 2 (and switching Alice and Bob's roles as sender/receiver as appropriate). Step 1 is never repeated.

Forward secrecy (achieved by generating new session keys for each message) ensures that past communications cannot be decrypted if either the session key or long term key is compromised, even though future sessions could be compromised.

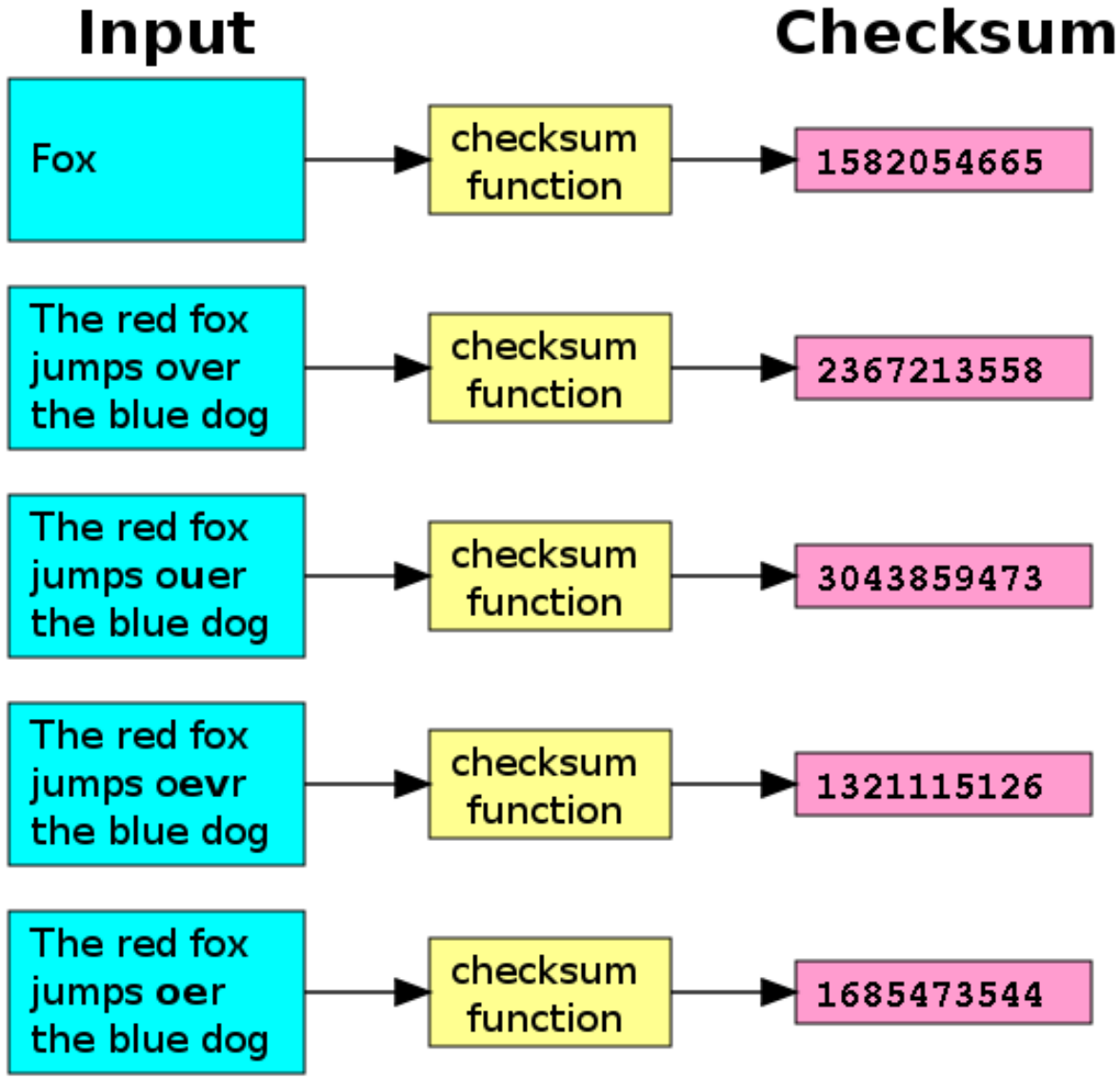
Integrity

(make sure information is trusted and not tempered by anyone.)

- Has Mallory tampered with Alice's/ Bob's message? Man-in-the-Middle (MITM) attacks!!!
- Safeguard techniques?
  - ✔ Checksum (e.g. CRC)
  - ✔ Hashing (e.g. MD5, SHA-1 algorithms)
  - ✔ Message Authentication codes (MACs)

CheckSum

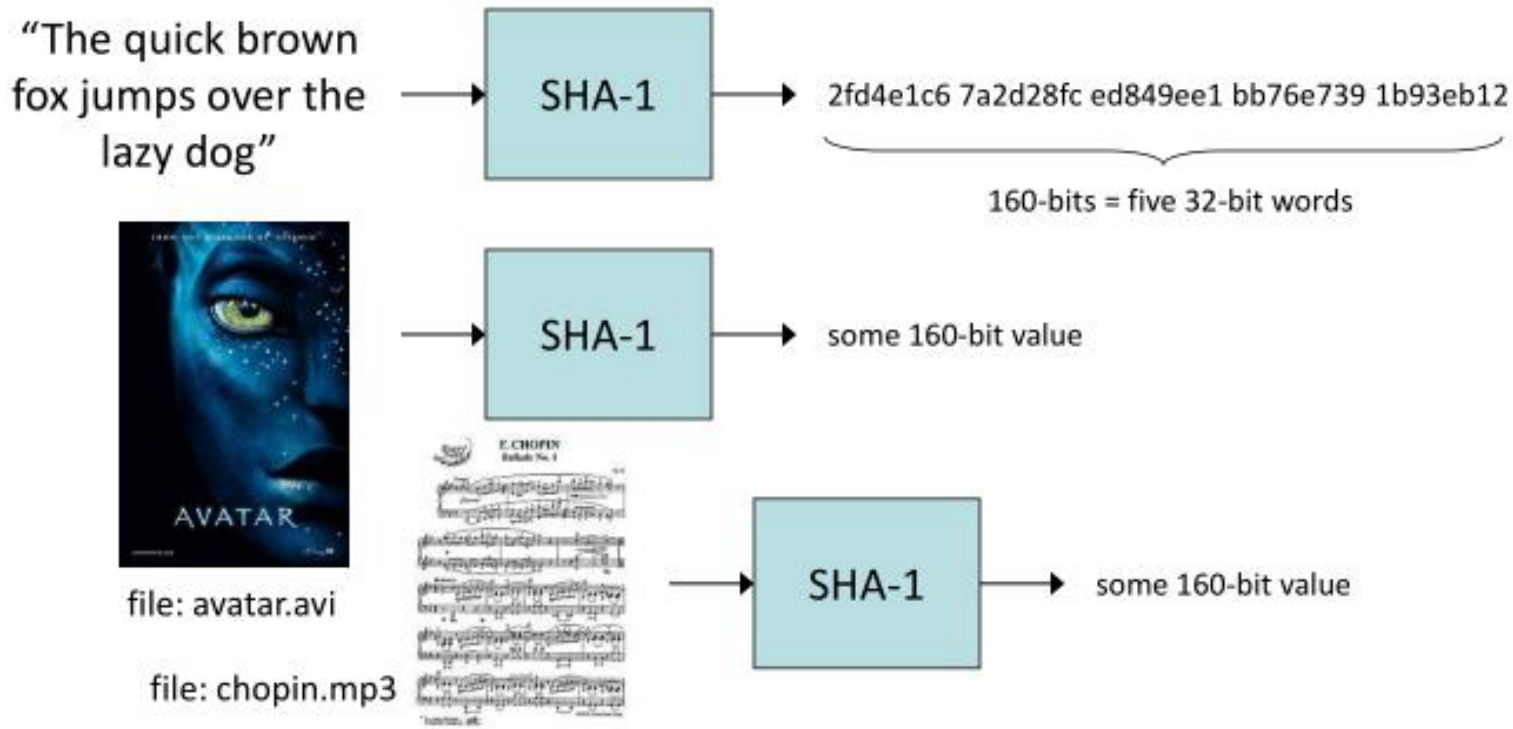
- Ensures message integrity not authentication
- A small change in input will produce new checksum
- Catches small data transmission/storage corruption errors.



Hashing for data integrity

Secure Hash Algorithm

- Goal is to compute a unique hash value for any input "message", where a "message" can be anything.
- SHA-1 (widely used) returns a 160-bit hash value (a.k.a. message digest or strong checksum)



Hashing for data confidentiality

1. Used for password etc which should not neither be stored in database nor be sent anywhere in readable form.

2. Does not need reversal (hashing to original text)

3. Is this safe?

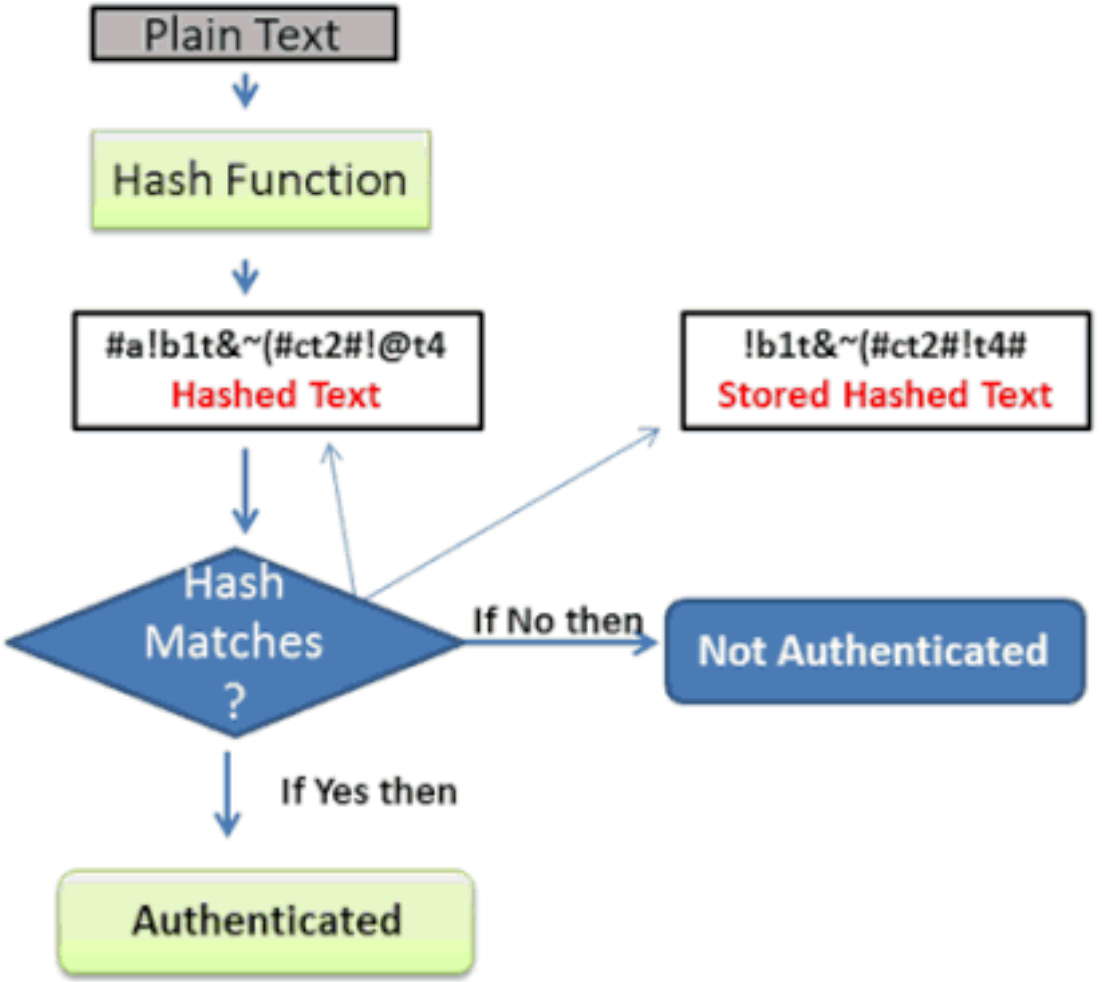
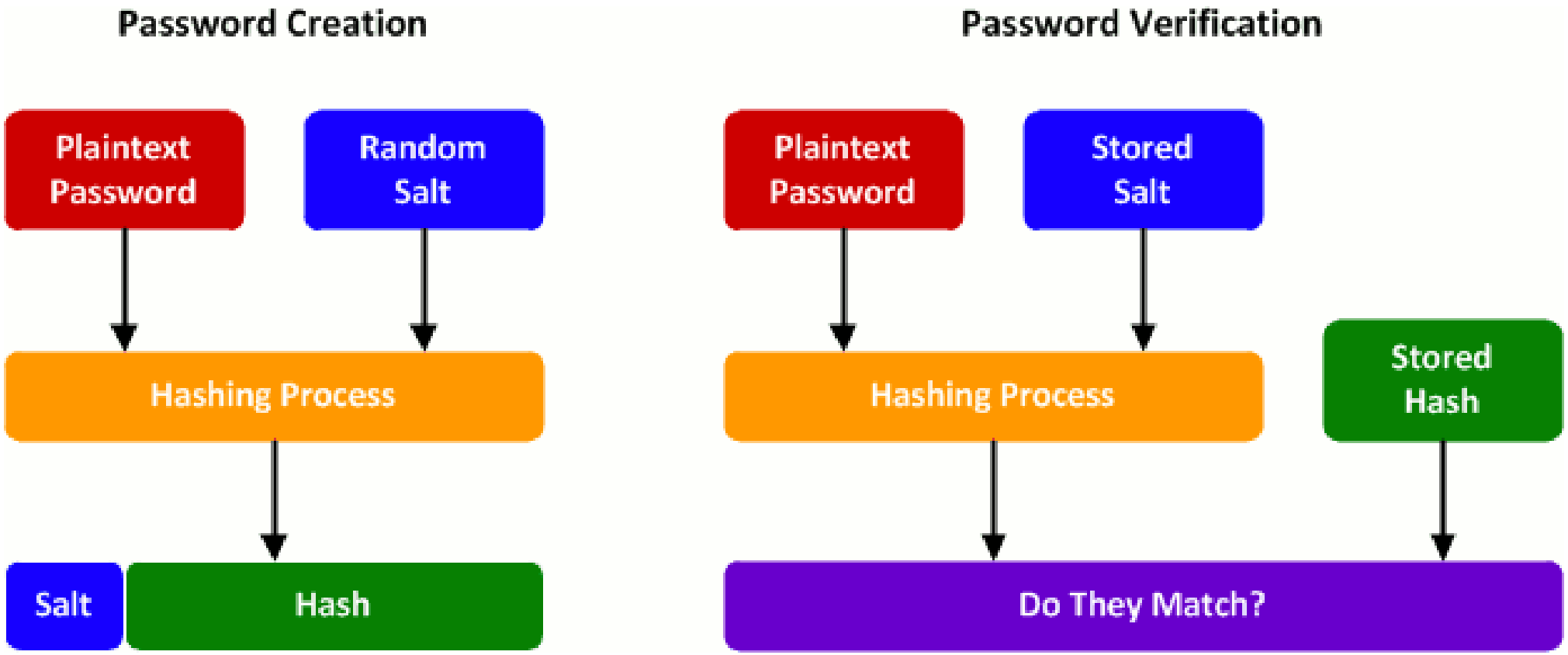


Fig : Flow Chart Example of Hash work

Hashing(**rainbow table attack**)

Password	**MD5 Hash**
123456	e10adc3949ba59abbe56e057f20f883e
password	5f4dcc3b5aa765d61d8327deb882cf99
12345	827ccb0eea8a706c4c34a16891f84e7b
12345678	25d55ad283aa400af464c76d713c07ad
qwerty	d8578edf8458ce06fbc5bb76a58c5ca4
123456789	25f9e794323b453885f5181f1b624d0b
1234	81dc9bdb52d04dc20036dbd8313ed055
baseball	276f8db0b86edaa7fc805516c852c889
dragon	8621ffdbc5698829397d97767ac13db3
football	37b4e2d82900d5e94b8da524fbeb33c0

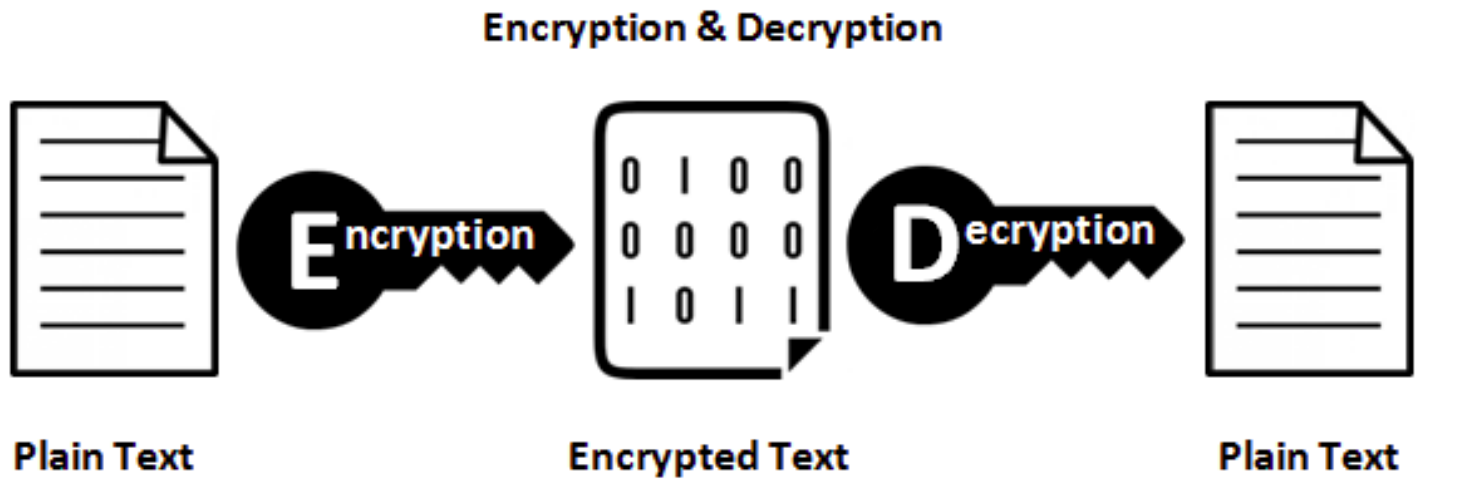
Hashing with salt **(making rainbow table useless)**

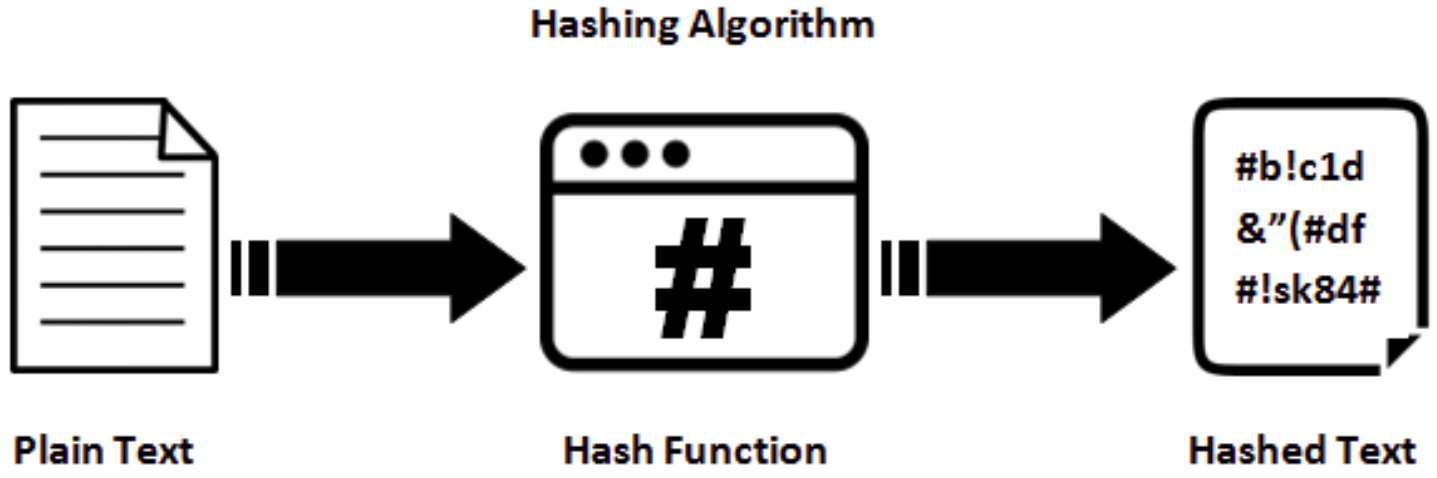


Hashing vrs. encryption

- Hashing
  - one way,
  - unkeyed
  - irreversible

- Encryption/decrption
  - 2 ways,
  - keyed
  - reversible





Message Authentication code (MACs)

- **Ensures message authentication and integrity both,**

  but not non-repudiation

- **Based on Symmetrical cryptography, secret key required at both ends.**