The University Of Queensland
AUSTRALIA
CREATE CHANGE

Business Information Security

- Week 07: Cryptography Part 2 (Ch. 8)

Dr Alex Pudmenzky

Semester 2, 2024

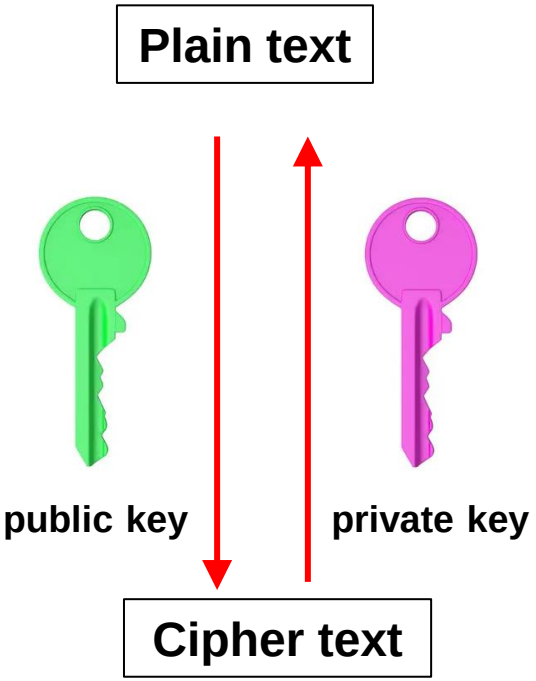
Overview

- Certificate Authority (CA) - distribution of public keys with trust

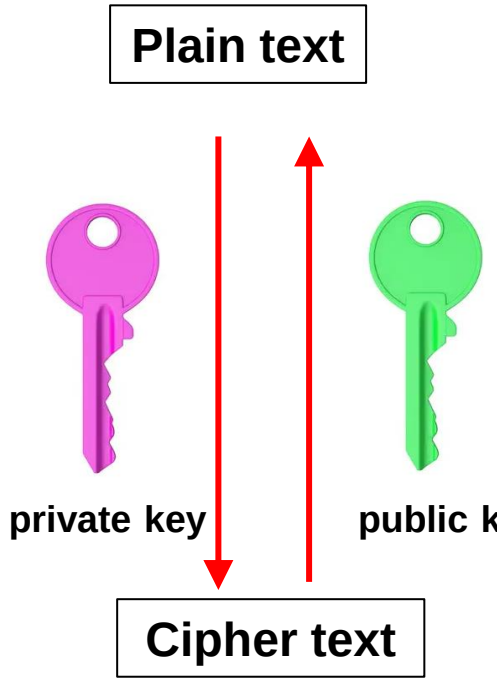
- Two important protocols in business
  - SSL/TLS (for the web)
  - S/MIME (for email)

Cryptography

Two main paradigms (both heavily used): **Private key (symmetric)** and **Public key (asymmetric)**.



Confidentiality because only the holder of the private key (the intended recipient) can decrypt the message

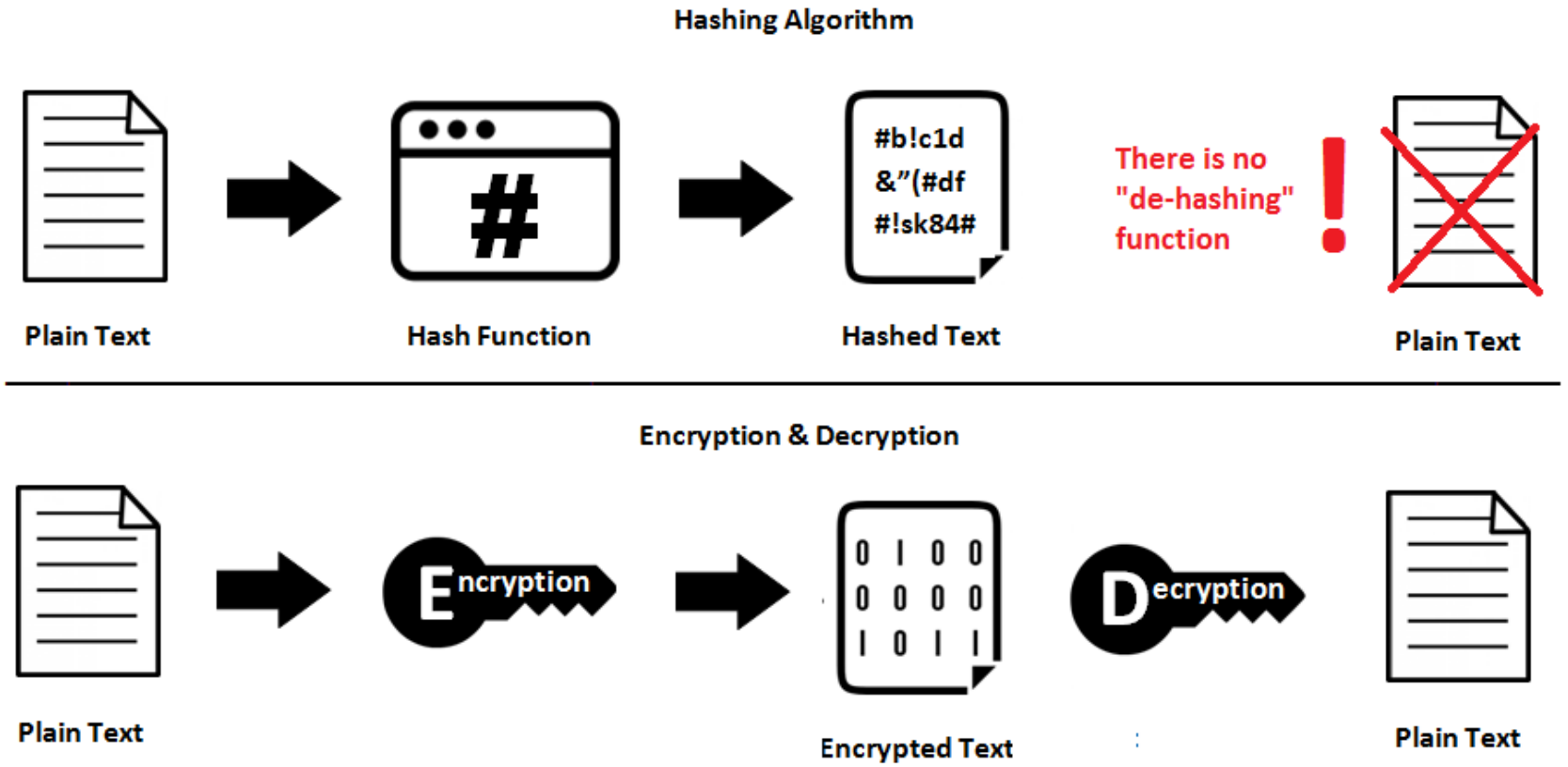


Authentication and **Integrity** because it confirms that the message was signed by the holder of the private key and that the message has not been altered.

Symmetric key
Has been around a long time (Caesar cipher).

Asymmetric key
Theory evolved in late 70s, the first major change in cryptography in a long time.

Difference Between Hashing and Encryption



- **Hashing** is a one-way function that changes a plain text to a unique digest† that is irreversible.
- **Encryption** is a two-way function that includes encryption and decryption.

† A compilation or summary of material or information.

https://www.ssl2buy.com/wiki/difference-between-hashing-and-encryption

Apply RSA Cryptography

1. Goto https://tinyurl.com/UQEncryptionKey (https://www.devglan.com/online-tools/rsa-encryption-decryption)
2. Create a public/private key pair† with the default number of bits.
3. Have a look at n, e of your public key https://report-uri.com/home/pem_decoder
   How many hex digits in n? How many bits in n?
4. Create a secret message using your private key (keep this secret!)
5. Goto https://tinyurl.com/UQCyberSecurity (https://docs.google.com/document/d/1zqlYZrOUyh-oYaE2cMQFGRf9U5rxMxy8ZND4ytsvGIM/edit?usp=sharing)
6. Enter your secret message and your public key into the table
7. Decode the message of another student using their public key and enter the decoded message into the table
8. Try decoding a message with a different public key
9. **Confidentiality** or **Authentication** and **Integrity?**

†RSA (Rivest-Shamir-Adleman):

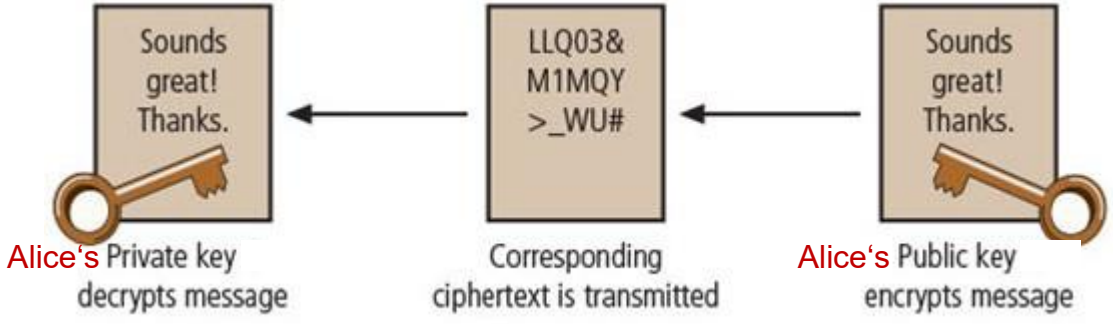
Based on: The difficulty of factoring large numbers.

Process

1.Generate two large prime numbers, p and q.
2.Calculate the product n = p * q.
3.Choose an integer e that is relatively prime to (p-1)(q-1).
4.Calculate d such that (e * d) mod (p-1)(q-1) = 1.
The public key is (n, e), and the private key is (n, d).

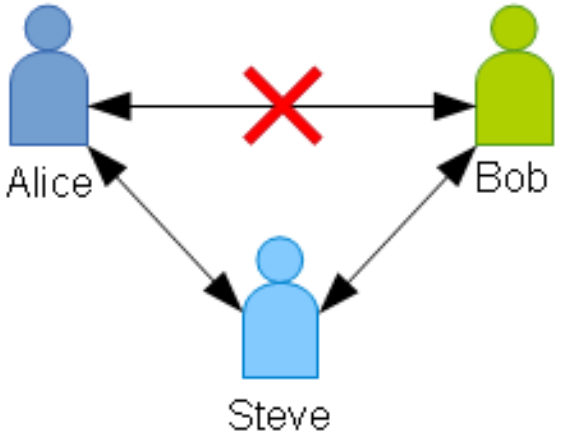
Where is the public key stored?

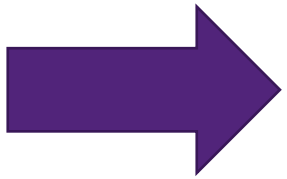
Alice at XYZ corp. wants to send a message to Bob at ABC corp. Bob stores his public key where it can be accessed by anyone. Alice retrieves Bob‘s public key and uses it to create ciphertext that can be decrypted only by Bob’s private key, which only he has. To respond, Bob gets Alice‘s public key to encrypt his message. Alice then decrypts the message with her private key.

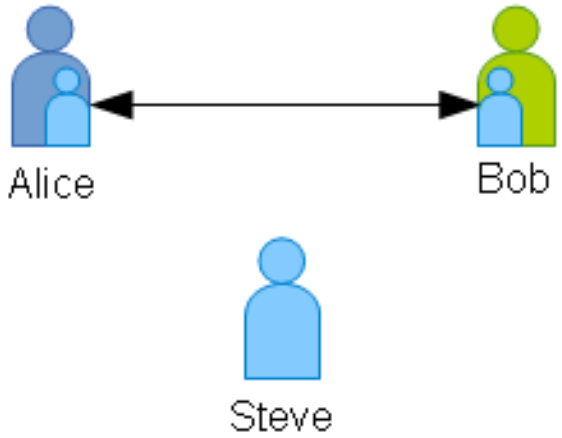


This is the response from Bob (read from right to left).

Third-Party Trust (Simplified)







Certification authority

http://justinparrtech.com/JustinParr-Tech/public-key-infrastructure-pki-and-encryption-simplified/

Public key infrastructures (PKI) #1

- The **use of public keys requires a framework of trust** (*why?*)
- Trust defined as the firm belief in the reliability, truth, or ability of someone or something
- This is what a PKI is – it is a framework for the practical operation of public key cryptography – more specifically, a **framework for the trusted (i.e. authenticated) distribution of public keys**.
- A PKI contains/provides:
  - **Standards** *(important to us)*
  - **Certification Authorities** *(important to us)*
  - **Certificates** *(important to us)*
  - **Operational protocols** *(important to us)*
  - Interoperable tools
  - Legislation (e.g. are digital signatures legally enforceable?) – we do not treat this.

Public key infrastructures (PKI) #2

- Integrated system of software, encryption methodologies, protocols, legal agreements, and third- party services enabling users to communicate securely

- PKI systems are based on public-key cryptosystems

- PKI protects information assets in several ways:
  - **Authentication:** Digital certificates in a PKI system permit individuals, organizations, and Web servers to authenticate the identity of each of the parties in an Internet transaction
  - **Integrity:** A digital certificate demonstrates that the content signed by the certificate has not been altered while in transit
  - **Confidentiality:** PKI keeps information confidential by ensuring that it is not intercepted during transmission over the Internet
  - **Authorization:** Digital certificates issued in a PKI environment can replace user IDs and passwords, enhance security, and reduce some of the overhead required for authorization processes and controlling access privileges for specific transactions
  - **Nonrepudiation:** Digital certificates can validate actions, making it less likely that customers or partners can later repudiate a digitally signed transaction, such as an online purchase.

Certificate authorities (CA's) #1

- Certificate Authority – a business – central within PKIs
- Trusted organization (a business, or a section within a corporation – e.g. the UQ CA) that:
  - Accepts applications for certificates (an entity for the distribution of public keys – we talk about these in detail shortly)
  - Authenticates applications – trust is the issue
  - Issues certificates
  - Maintains status information about certificates
  - Revokes certificates
  - Provides an historical audit trail of certificates
  - On some scale of operation (e.g. global)

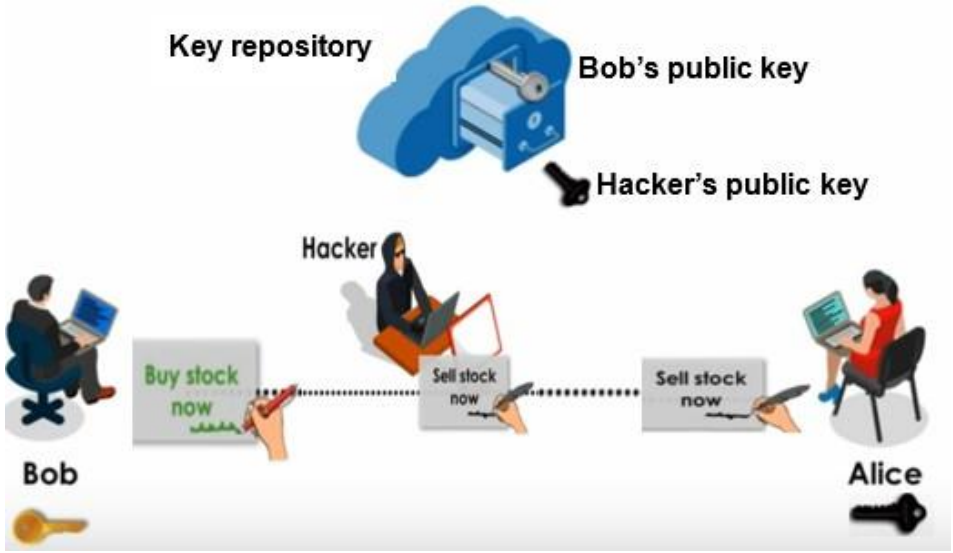
Certificate authorities (CA's) #2

- **CA must confirm the information the subject has provided.**
  - Remember TRUST is the essential goal.
  - This level of information is therefore variable and depends on the level of trust indicated by the certificate.
- Level of trust is related to the level of assurance sought by possession (and use of) the certificate
- A Local Registration Authority (RA) may be involved – an agent acting on behalf of the CA.
  - **A CA is a business model!**
- The CA issues a Digital Certificate to a suitably assessed client (i.e. individual or business)
- The issued certificate (aka as a digital ID by Verisign) can be for a variety of purposes:
  - an TLS digital certificate (used by just about all businesses for secure Web transactions),
  - an email certificate,
  - a code-signing certificate,
  - a VPN certificate
  - *(most of these further discussed next week)*

Digital certificates #1

- A certificate is a binding between an entity's public key and identifying characteristics of the entity. That is, a certificate contains the public key and other identifying information for some entity.
- Similar to a '***digital passport***' or 'digital driver's license.
- Also Known As: digital ID, certificate.
- Issued to an entity (person, software, hardware item, etc) For example – issued to a person for securing email, or a web server program to secure ecommerce transactions.
- Are used for authentication, key exchange, non-repudiation
- Aim to achieve TRUST

Digital certificates #2 – Why we need them



Public Key Repository – Not good

- We need strong authentication – we must further support the digital signature model (public key itself does not verify the ID of sender)
- The solution is a digital certificate – it verifies the owner of the **certificate** AND verifies the owner of the **public key**
- Digital certificates must be readily available and readily exchangeable (without heavy security restricting exchange)

Digital certificates #3

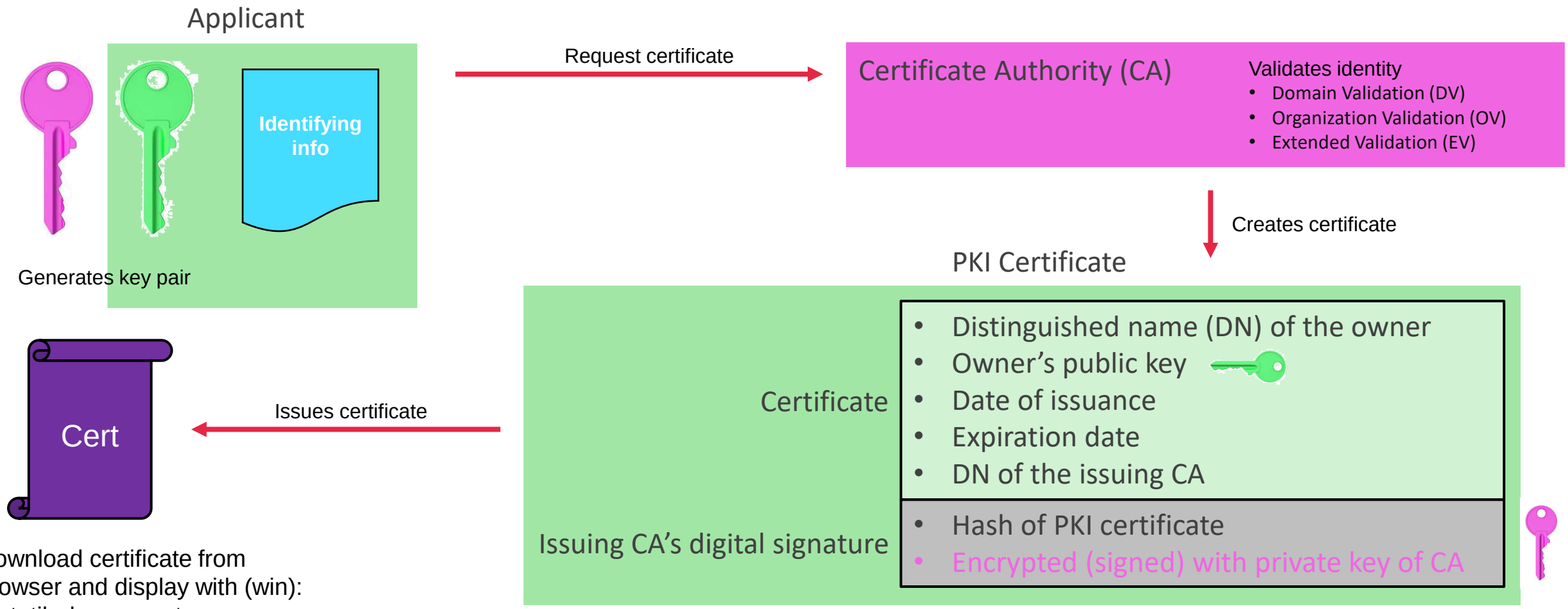
Digital Certificate - Good

- Bob obtains a digital certificate – which also contains his public key – from a trusted organization (the Certificate Authority or CA)
- Bob creates the message, signs it (using hashing and his private key) – then sends this with a copy of his digital certificate to Alice
- How have we controlled this message (integrity, authentication, non-repudiation) – explain your reasoning. Is Bob's key authentic? Has it been changed?

Digital certificates #4

- Not very useful if individuals issue their own certificate (why? It must be based on trust)
- Certificates must be verifiable – forgeries must be readily detected. Therefore we must control for authentication/integrity
- Certificates must be readily available and readily exchangeable (without heavy security restricting exchange)
- **Must be scalable** – **grow with demand** (number of users and the evolution of the types of use)
- Must conform to an open standard (for future focus and individual domain needs)
- We need a Certificate Authority (CA) for certificate distribution. We also need an international standard for the content/design of a digital certificate

1.Root Certificates: CAs have root certificates that contain their public keys. These root certificates are self-signed and serve as the foundation of trust for the CA's entire certificate hierarchy.
2.Trust Stores: Operating systems, browsers, and other software maintain a list of trusted root certificates from well-known CAs. These trust stores are pre-installed and regularly updated to include only reputable CAs.



Download certificate from browser and display with (win): certutil -dump <cert>

Digital certificates #5

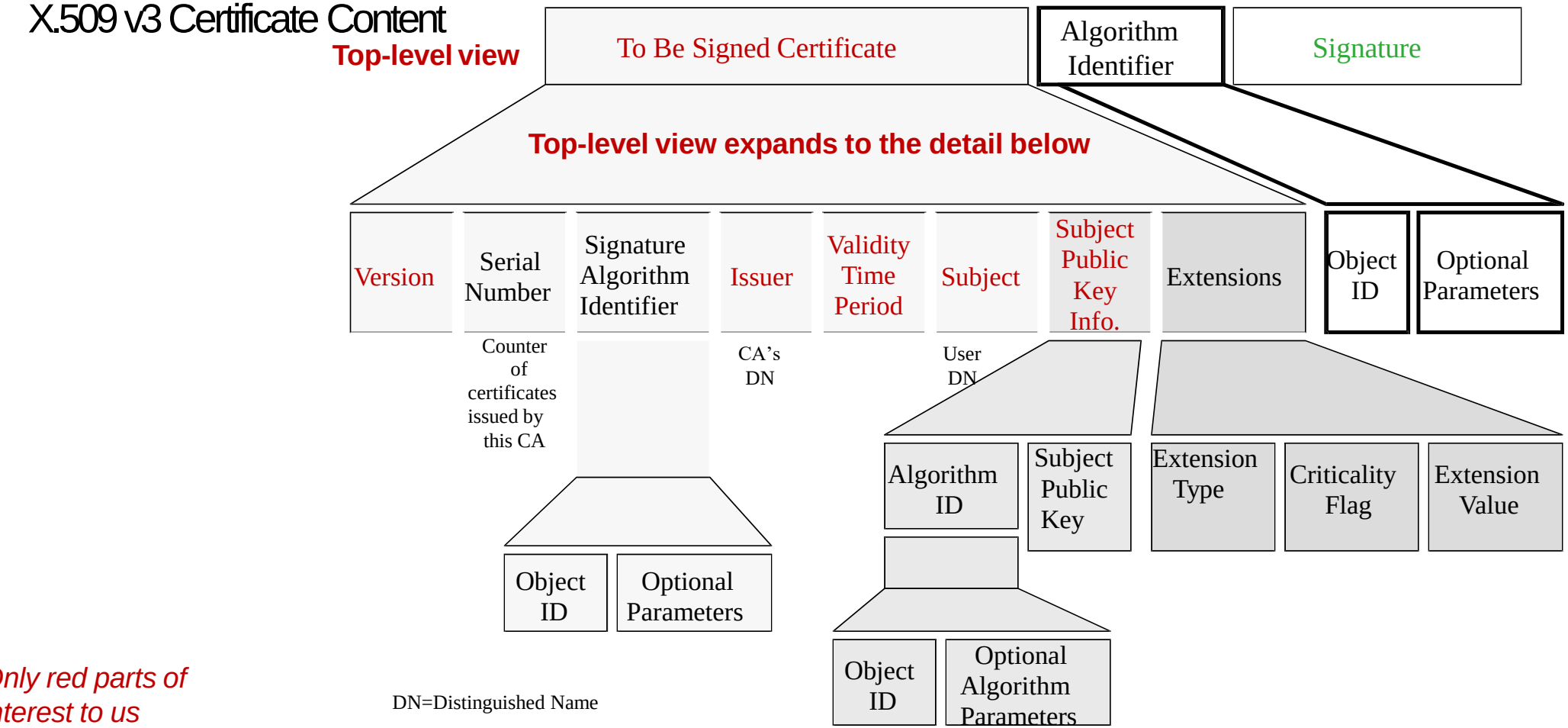
- X.509 is an ITU and ISO certificate format standard (a framework or high level standard)
  - First published in 1988 – part of the X.500 Directory recommendations. Latest version is Version 3 issued in 1996.
  - An X.509v3 certificate contains a set of basic, predefined fields and zero or more extension fields. This is the standard that business should use when a digital certificate is required (and it frequently is) – it is very widely recognised.

- Each certificate contains the **public key of a user** and is **signed with the private key of a CA**
  - *(does the certificate contain the PRIVATE KEY?*)

- **Is used in S/MIME, IP Security, SSL/TLS and other protocols**

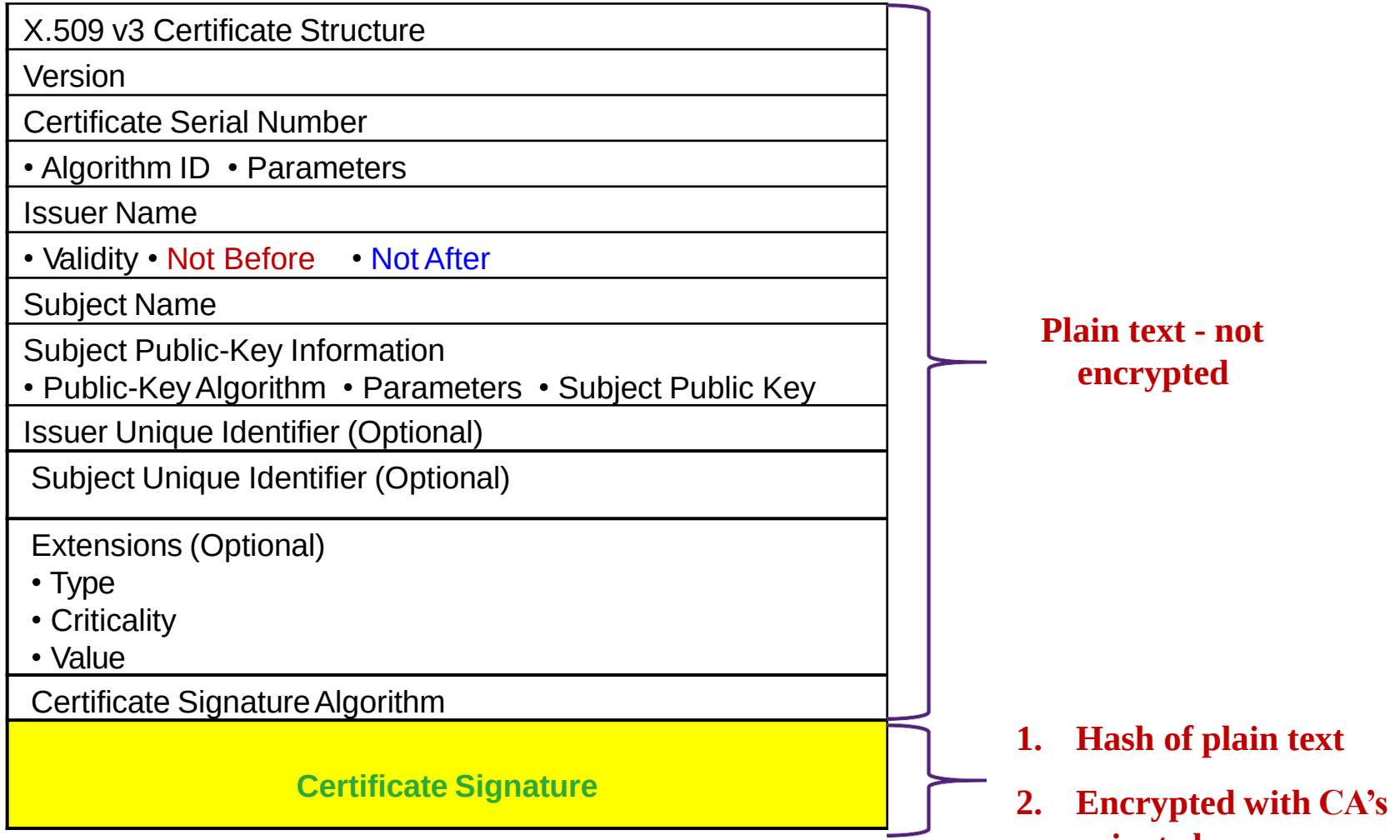
Digital certificate content (view #1)

A digital signature is always a hash of the data to be signed and the digest that is produced is then encrypted with the private key of the CA.



Only red parts of interest to us

Digital certificates – the contents (view #2)



Root and user-level certificates

- Characteristics of certificates generated by CA:
  - Any user with access to the public key of the CA can verify the user public key that was certified – this introduces the concept of a root certificate – **the trusted distribution of the CA's public key**
  - No party other than the CA can modify the certificate without this modification being detected – this is because of the inbuilt digital signature control (i.e. contained within the digital certificate)

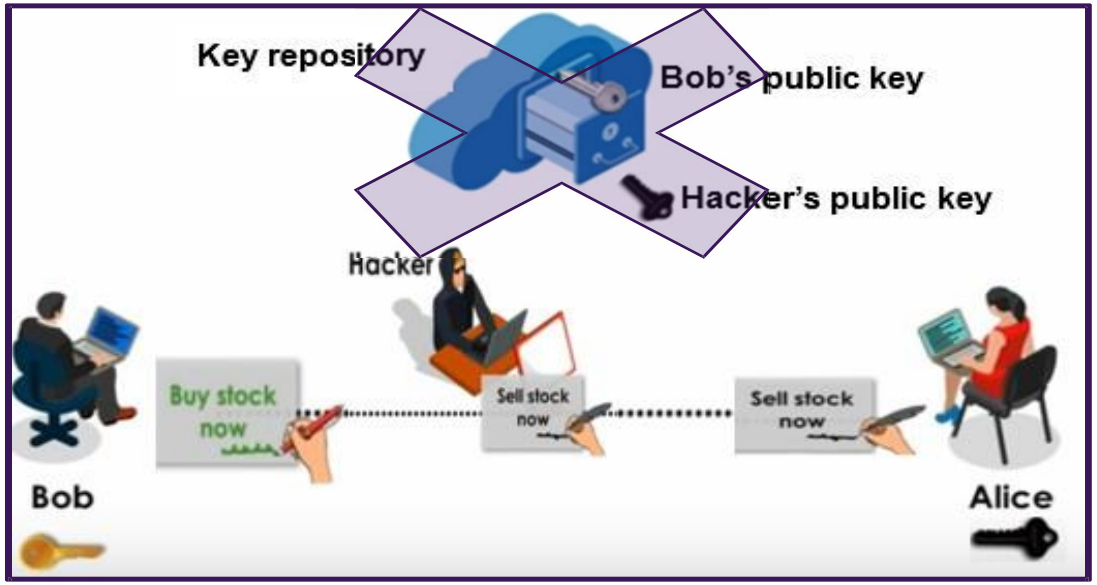
- Therefore we end up with (at least) two levels of certificates:
  - (1) **root certificates** (i.e. the certificate of a CA containing the public key of the CA), and
  - (2) **user-level certificates** (containing the public key of the user and issued by the CA using its public key)

Digital certificates (continued)

- **A primary responsibility of a CA is to revoke certificates** (they have an expiry date – but sometimes it is necessary to 'remove' the certificate from general circulation before this date is reached)
  - Certificate Revocation Lists (CRL)
  - Revocation status is the most prominent information a CA maintains about the status of a certificate
  - Revocation reasons: key compromise, change in the subject's affiliation, CAs certificate has been compromised
  - CRLs must be publicized by CAs
  - Distribution method – polled or pushed by certificate using software (e.g., web browser – certificates are used heavily by Web browsers – e.g. Firefox, IE, etc. – more on this next week)
  - X.509 specifies a CRL format – the format is not examinable, included here for completeness.

Conclusion: The Need for Digital Certificates

Not this system



Yes – this system

Security Topic – Cryptography (2)

SSL/TLS (web), S/MIME (for email)

Now we can start our work on ‘hybrid’ security!

Firstly, communication protocols – some important characteristics

- Know what a protocol is …
- Know basic telecommunications terminology.
- Understand the nature of processing in a layered communications protocol – how the protocols combine to get the whole job done.
- Know the purpose of the five layers of the TCP/IP-OSI protocol/model/framework
- Appreciate that protocols construct the content of network packets (which are comprised of control headers and data)

Communication *protocols* – consider '*snail-mail*' *addressing*

Mr Harry Clark
16 Jones Road
Maleny 4552

I've found XYZ as a great and influential company, so I would like to use a chance to present myself to be a part of your design team.

For the past four years, I've worked on various design projects including websites, outdoor and print advertisements, brand logos, and book illustrations. Using mostly Adobe Creative Suite together with a great team, we have implemented successful design solutions for more than 70 websites, including brands such as Max Tire, East-West Airlines, and Rockstar Mag. I have an end-to-end understanding of project development from concept to solution, and from pitch to discussed implementation. XYZ company boasts famous work such as Dove Milk Bird and ABC Color Balls, and they would be the perfect place for me to work.

I would like to meet in person to share more of my knowledge, learn more about your graphic design needs and discuss ways I can help to attain them. Please feel free to call or email me. I look forward to this opportunity.

Thank you for your time,
Gabriel Rees

The letter

Maleny

The envelope

Mr Harry Clark
16 Jones Road
Maleny 4552

Several 'layers' of addressing

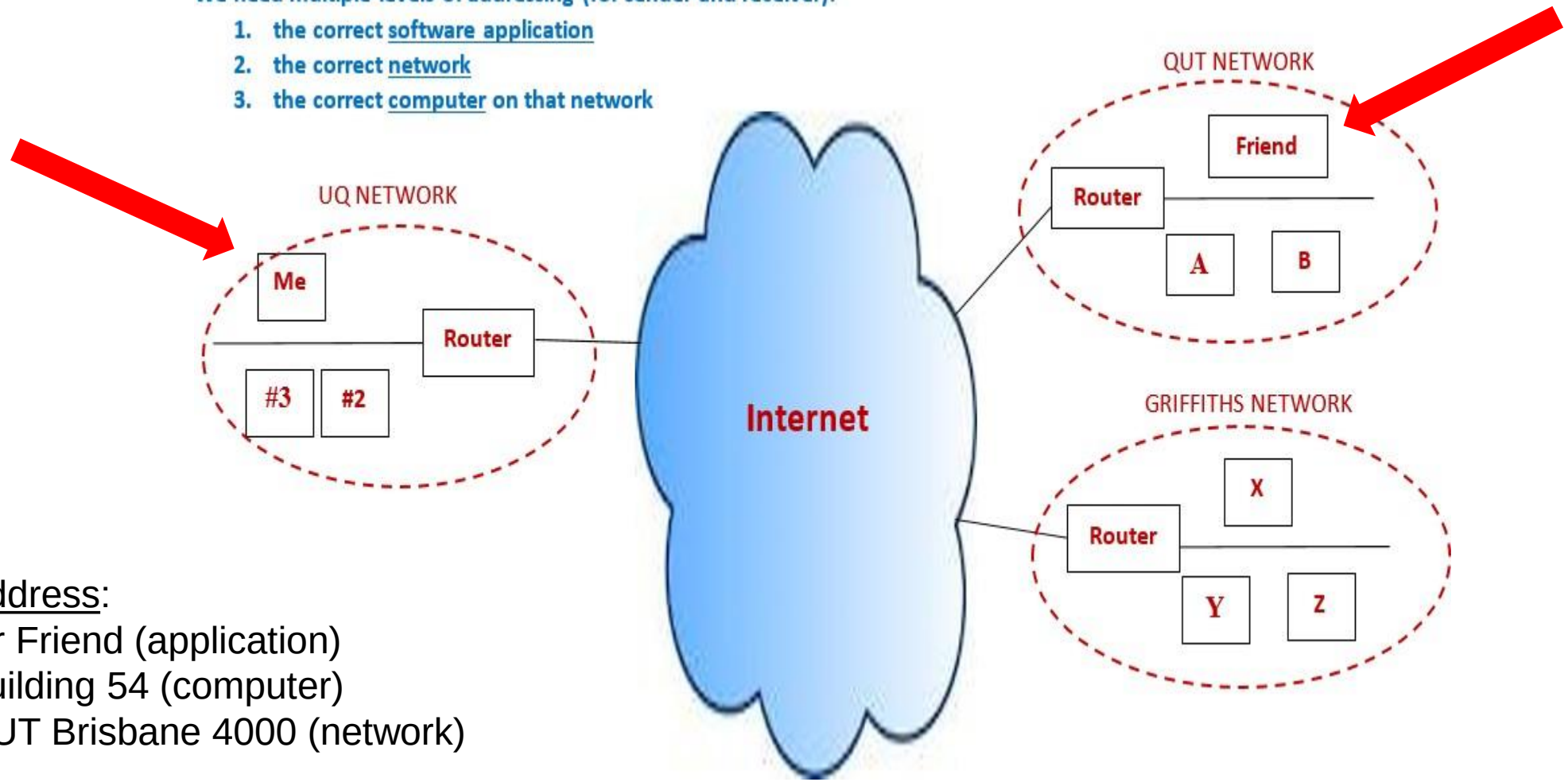
- Business/person
- Property number, street name
- Suburb/town/postcode

This is 'hierarchical' NOT 'flat'

Communication *protocols* – consider 'network' addressing

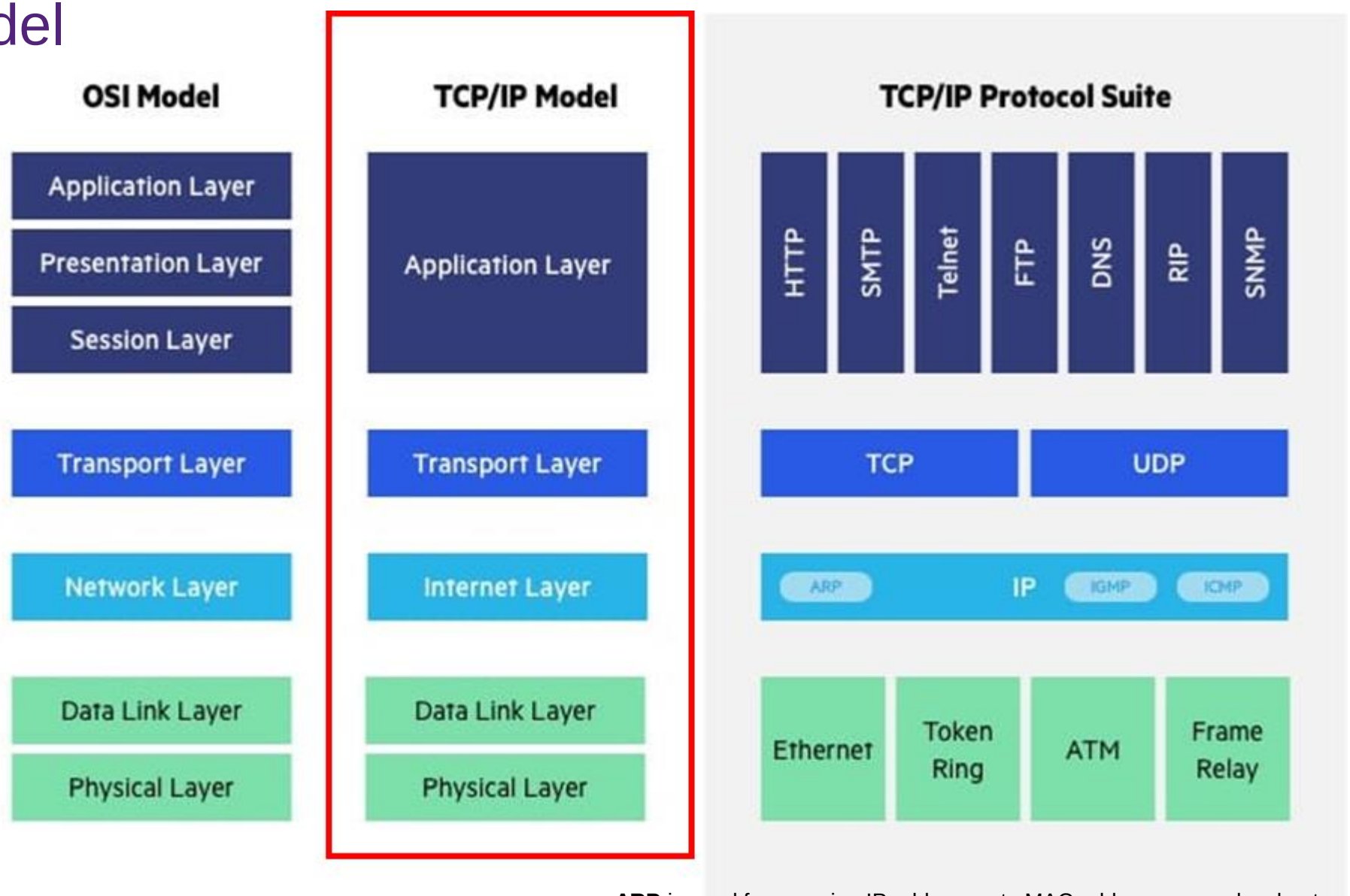
We need multiple levels of addressing (for sender and receiver):

1. the correct software application
2. the correct network
3. the correct computer on that network



Address:
Mr Friend (application)
Building 54 (computer)
QUT Brisbane 4000 (network)

OSI vs. TCP/IP Model
7 vs 5 layers

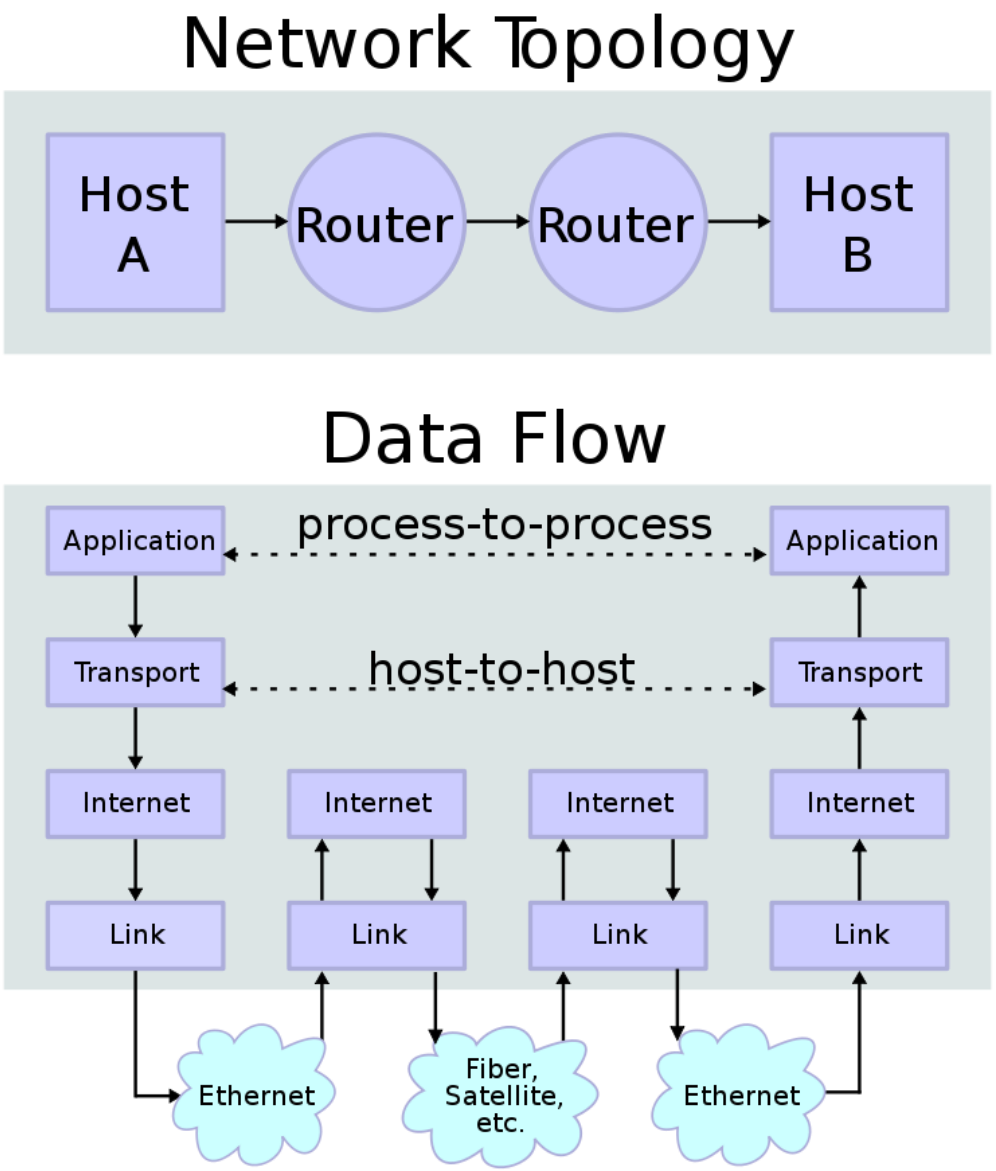


Some abbreviations:

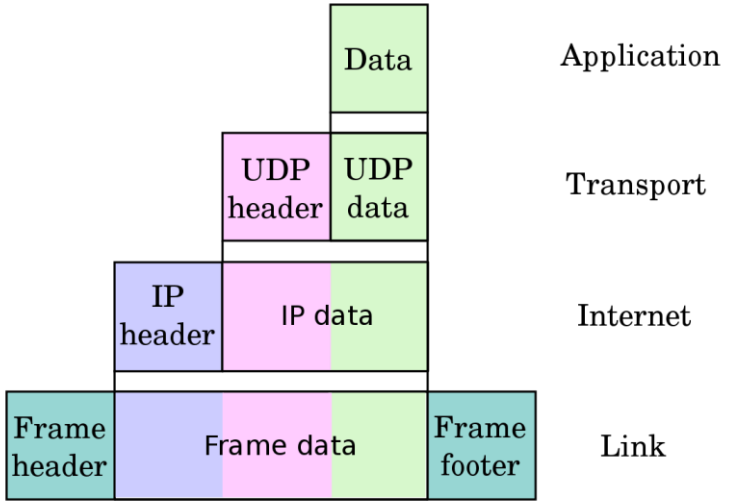
OSI: Open Systems Interconnection
TCP: Transmission Control Protocol
IP: Internet Protocol
UDP: User Datagram Protocol.

https://www.imperva.com/learn/application-security/osi-model/

- **ARP** is used for mapping IP addresses to MAC addresses on a local network.
- **ICMP** is used for error messages and network diagnostics.
- **IGMP** is used for managing multicast group memberships.



Conceptual data flow in a simple network topology of two hosts (*A* and *B*) connected by a link between their respective routers. The application on each host executes read and write operations as if the processes were directly connected to each other by some kind of data pipe. After establishment of this pipe, most details of the communication are hidden from each process, as the underlying principles of communication are implemented in the lower protocol layers. In analogy, at the transport layer the communication appears as host-to-host, without knowledge of the application data structures and the connecting routers, while at the internetworking layer, individual network boundaries are traversed at each router.



Encapsulation of application data descending through the layers described in RFC 1122.

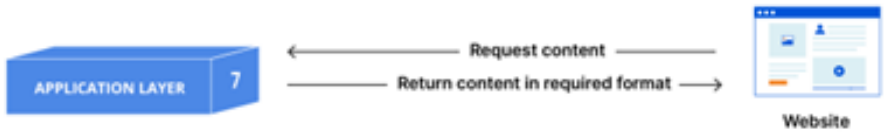
https://en.wikipedia.org/wiki/Internet_protocol_suite

NB. Link=Network

Communication *protocols* – a series of 'layers'

We have (from the ISO) a 'layered' communication model. This will be very helpful for our work with other security topics (e.g. firewalls). We show an **adapted version** of this below.

Layer	Name	Our focus	Examples
5	Application	This layer describes how two software applications communicate with each other (e.g. a web browser communicating with a web server)	Web (HTTP); Email (SMTP)
4	Transport	This layer describes (for us) the application addressing scheme	TCP
3	Internet/Network	This layer describes (for us) the addressing of the correct network and correct computer on that network	IP
2	Data Link	(not discussed in this course)	
1	Physical	(not discussed in this course)	



7 **Application Layer**

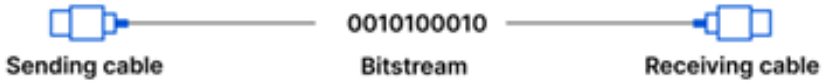
6 **The Presentation Layer**

5 **The Session Layer**

4 **Transport Layer**

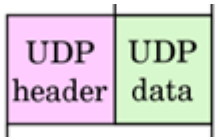
3 **The Network Layer**

2 **The Data Link Layer**

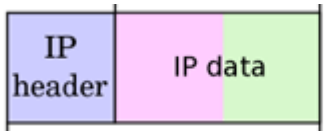


1 **The Physical Layer**

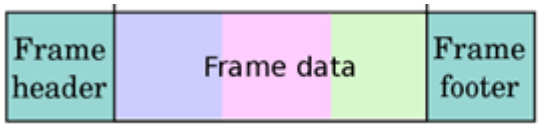
Application



Transport



Internet



Link

https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/

Communication *protocols* – addressing characteristics

Network & computer address (IP Addresses)

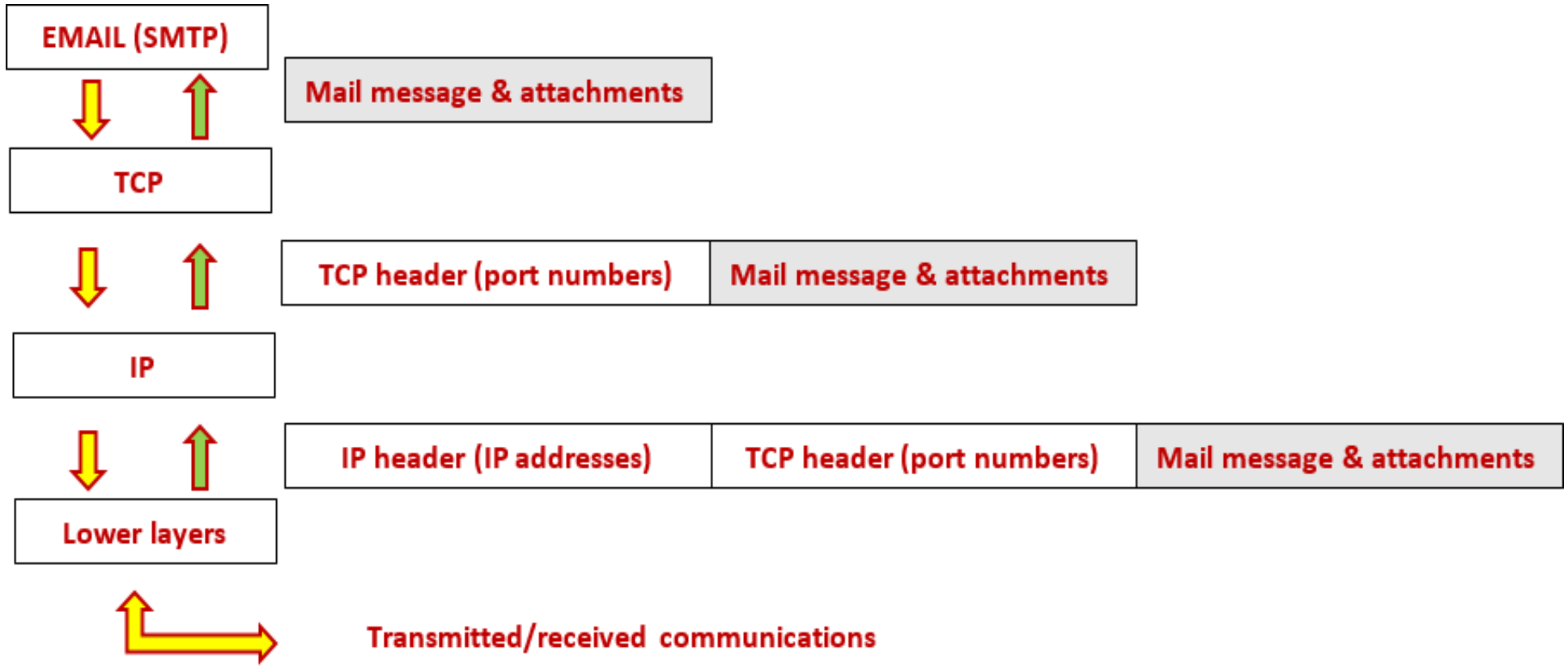
- The Internet uses addresses, which are also called IP addresses, for example, 192.168.2.28.
- An IP address comprises a network identifier component and a host identifier component.
- Layer 3 (network layer) is concerned with IP addresses.
- So 192.168.2.28 has a network identifier component (i.e. part of the address) AND a computer identifier component (i.e. the other part of the address) - learn which is which next week!

Application address (Port Address or simply: port)

- The concept of a port number (a simple integer) is used to identify the identity of the application sending the communication and also the identity of the application receiving the communication.
- Ports are extremely relevant in IS security.
- Layer 4 and 5 (transport & application) are concerned with port addresses.
- As an example, a ‘web server’ always has port 80, you write port numbers like this 192.168.2.28:80

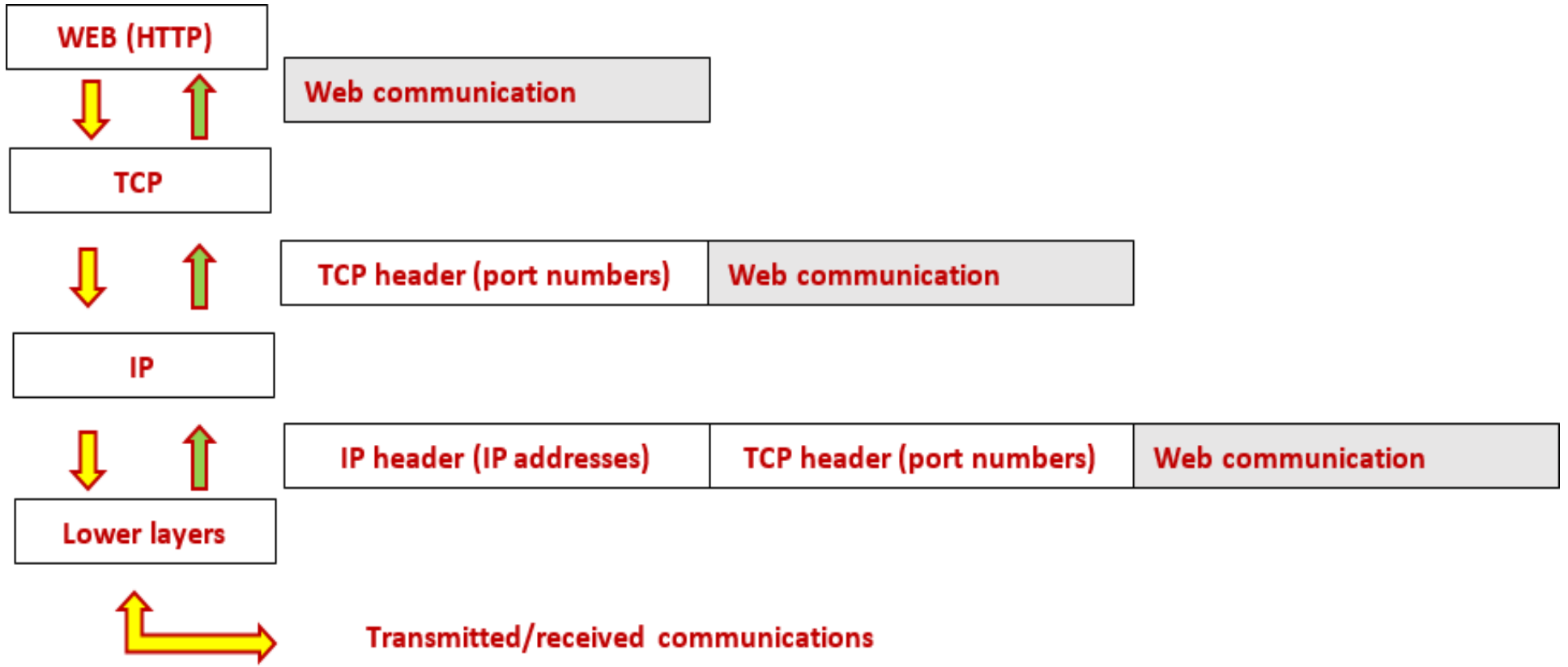
Communication *protocols* in action – email (SMTP)

Simple Mail Transfer Protocol



Communication *protocols* in action – web (HTTP)

Hypertext Transfer Protocol



No security protocols...!

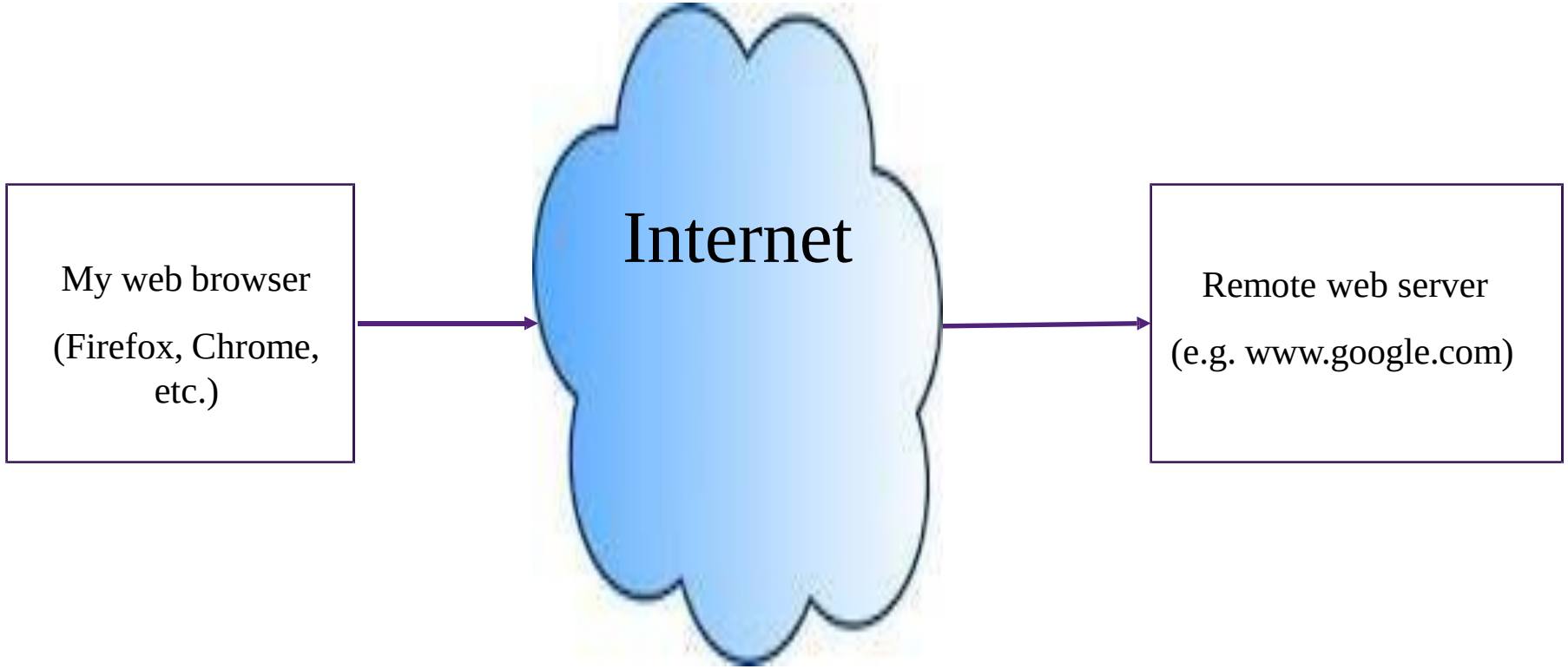
Security ***protocols*** – important characteristics

- We must differentiate between Offline and Online Encryption Systems
- Secure communication protocols
- Offline communication protocol (**email**) – take a message, encrypt it and either store the message or transmit it to another user (**S/MIME**)
- Online communication (**web**) – require real time interaction between participants (**TLS**)

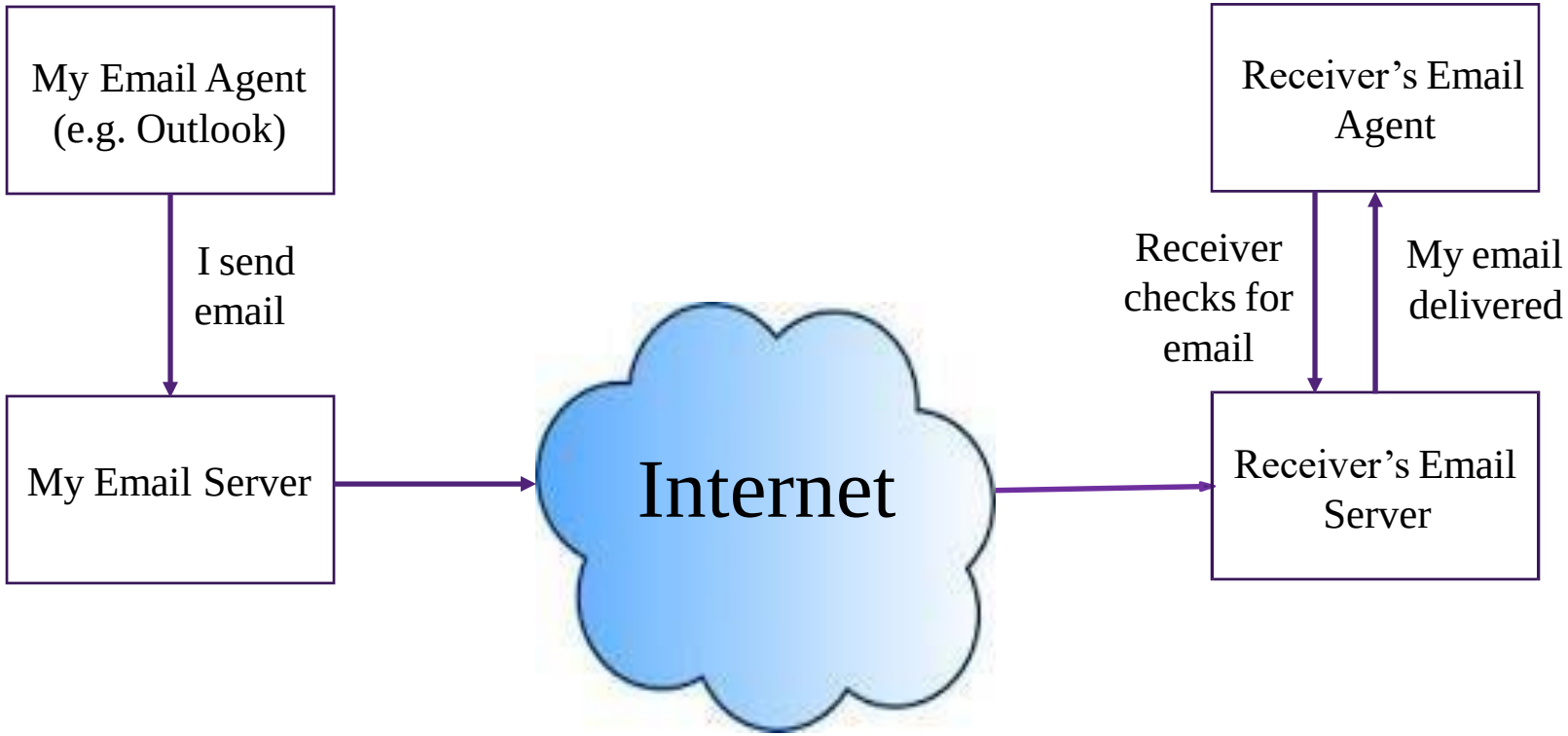
S/MIME: Secure/Multipurpose Internet Mail Extensions.

TLS: Transport Layer Security.
TLS is the successor to Secure Sockets Layer (SSL) and is often referred to as SSL/TLS.

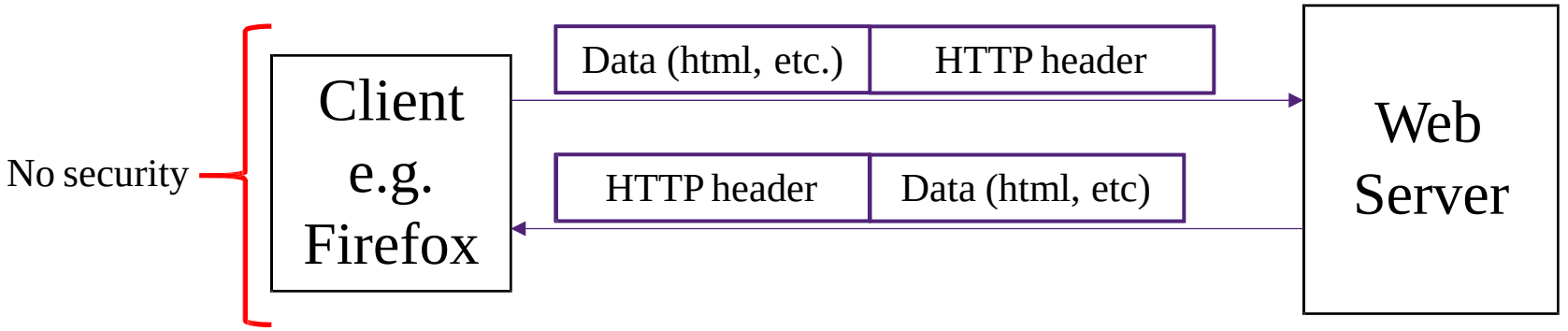
Web is online – therefore TLS is online
(communicating with each other in 'real-time')



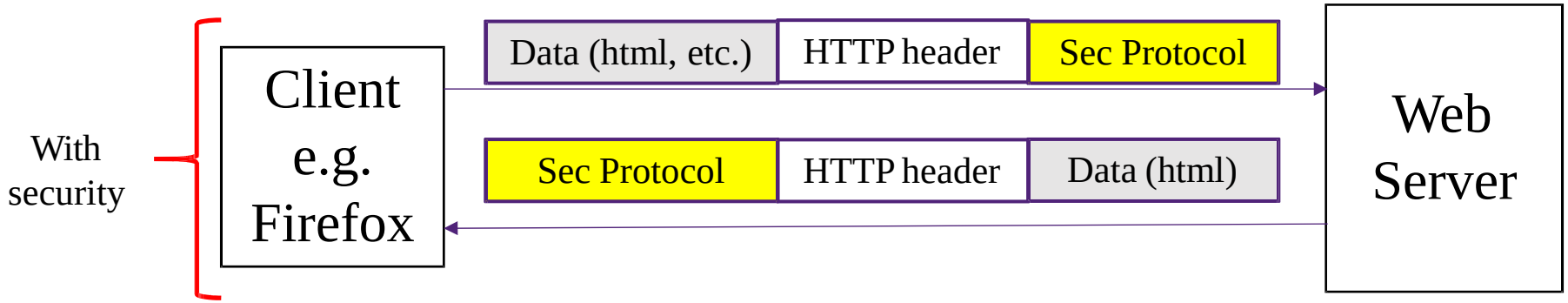
Email is offline – therefor S/MIME is offline
(not communicating in 'real-time')



The web requires an online security *protocol*

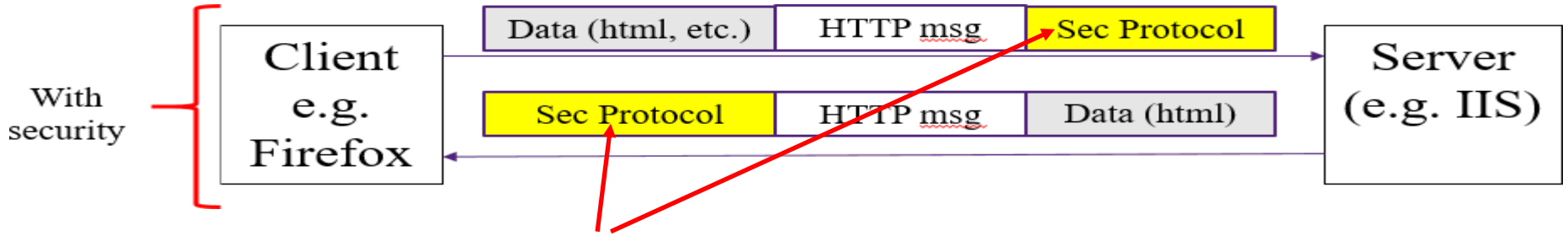


This communication exchange is **always ONLINE**



HTTP is the communication protocol between a web browser and a web server – it is a very simple 'language' designed to enable a 'discussion' between the two entities – this defines the Web. Originally, the Web had NO SECURITY – it has now been added – we shall now discuss this security protocol!

Web security – **historically** via Secure Sockets Layer (SSL)

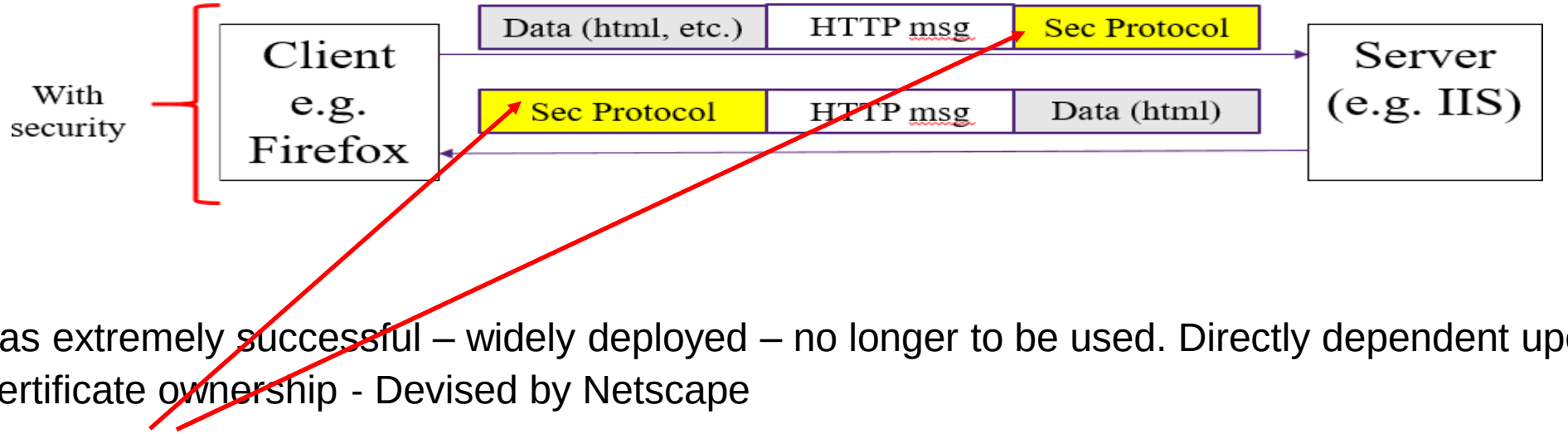


- For over 20 years Secure Sockets Layer (SSL) has been in the market as one of the most widely-used encryption protocols ever released and remains in widespread use today despite various security vulnerabilities exposed in the protocol.

- SSL v3.0 was superseded in 1999 by TLS v1.0, which has since been superseded by TLS v1.1 and v1.2. To date, SSL and early TLS no longer meet minimum security standards due to security vulnerabilities in the protocol for which there are no fixes. It is critically important that entities upgrade to a secure alternative as soon as possible and disable any fallback to both SSL and early TLS.

- The SSL protocol (all versions) cannot be fixed. SSL and early TLS (1.1) no longer meet the security needs of entities implementing strong cryptography to protect payment data over public or untrusted communications channels. Additionally, modern web browsers have begun prohibiting SSL connections, preventing users of these browsers from accessing web servers that have not migrated to a more modern protocol.

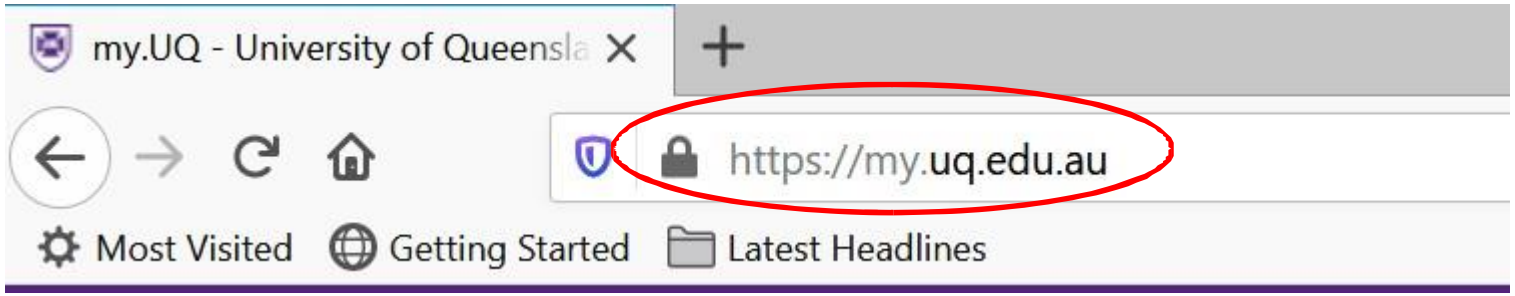
Secure Sockets Layer (SSL) and TLS



- SSL - was extremely successful – widely deployed – no longer to be used. Directly dependent upon digital certificate ownership - Devised by Netscape
- **TLS (transport layer security)** - an Internet standard (TLS) managed by the Internet Engineering Taskforce (IETF). Latest version: TLS 1.3 (August 2018) TLS now to be used.
- A general purpose security protocol – not a specialized secure payment protocol. What does this mean
  - Not just for the Web - secures any content that will be carried by TCP *(explained next week)*
  - Does not provide transaction accountability, i.e. does not check if credit card is valid
  - Secures data during transmission only – not whilst resident on client or server machines

Web secure communication: TLS

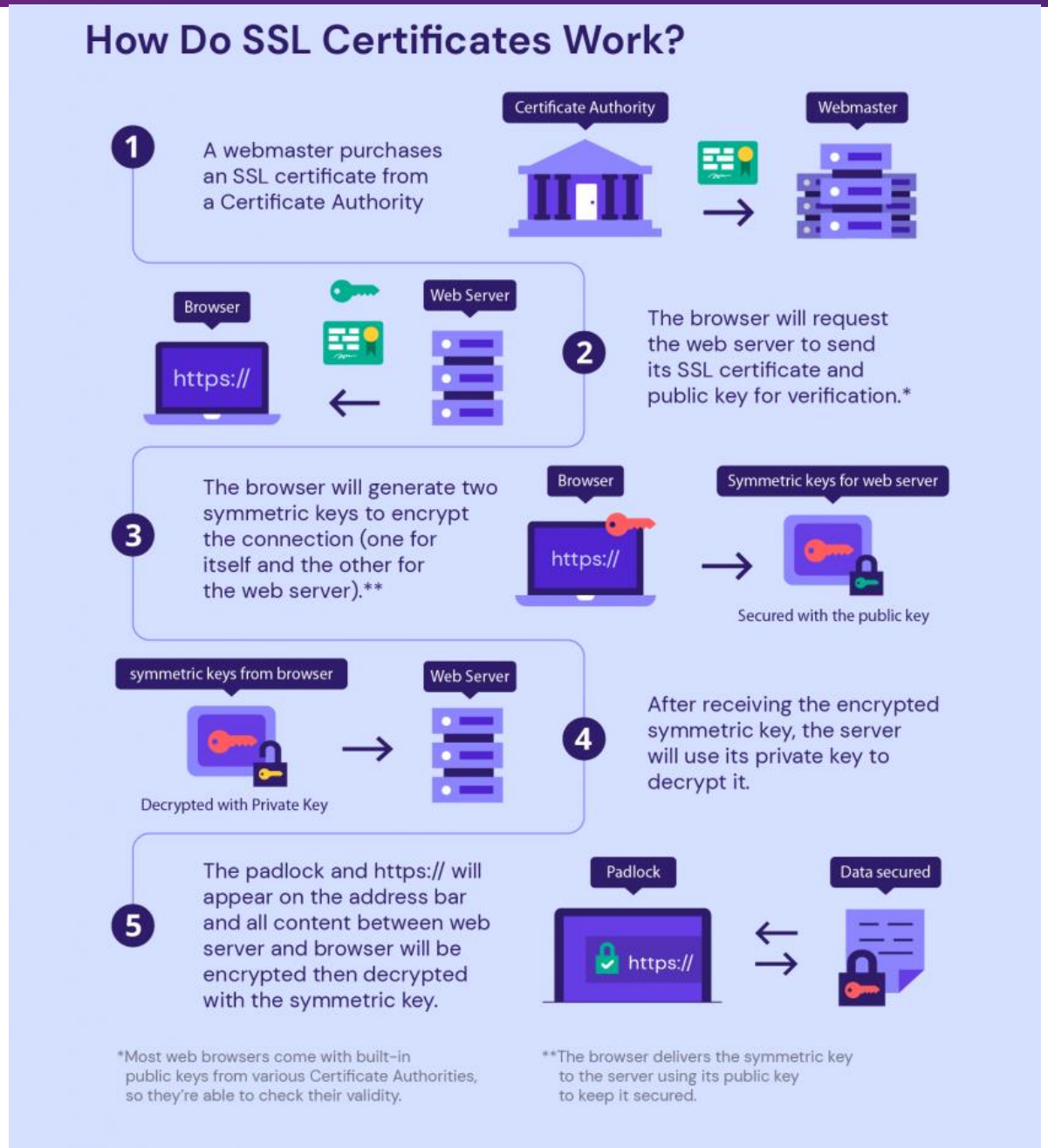
- Transport layer security (TLS) protocol: an Internet standard (TLS) managed by the IETF.
- TLS now to be used
- **HTTPS**
  - The HTTP protocol with the data encryption using TLS



Transport Layer Security (TLS)

- TLS working group was formed within IETF
- First version of TLS can be viewed as an updated version of SSLv3.1

- We must consider:
  - Levels of authentication (partial or full – implications)
  - Paradigms of security (it is hybrid, symmetric key and public key)
  - Key distribution problem
  - Certificate usage – private key usage to confirm ownership of certificate.
  - Root certificates – where must they be located - implications



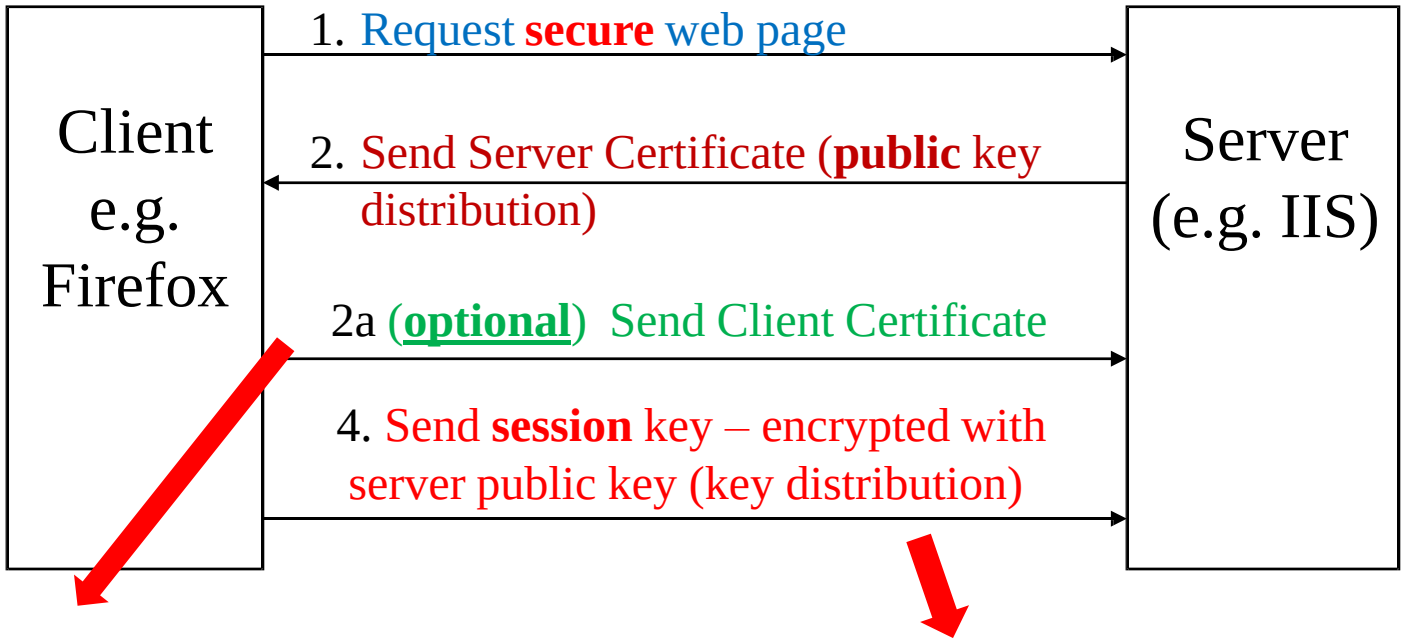
Browser asks server for certificate containing server's public key.

Browser sends its symmetric key to server encrypted using the server's public key.

Server reads browser's symmetric key so they both now have the same secret key which both use from now on.

https://www.hostinger.com/tutorials/what-is-ssl

TLS – How it works – (a business conceptual view)



3. Validate server's certificate: signature via root certificate, URL matches name on certificate, validity from-to current. If not valid, user is warned. *Generate symmetric session key*

5. Server uses its private key to decrypt session key. Now both sides use this 128 bit symmetric session key to exchange secured data. Server's public key typically 1024 bits long.

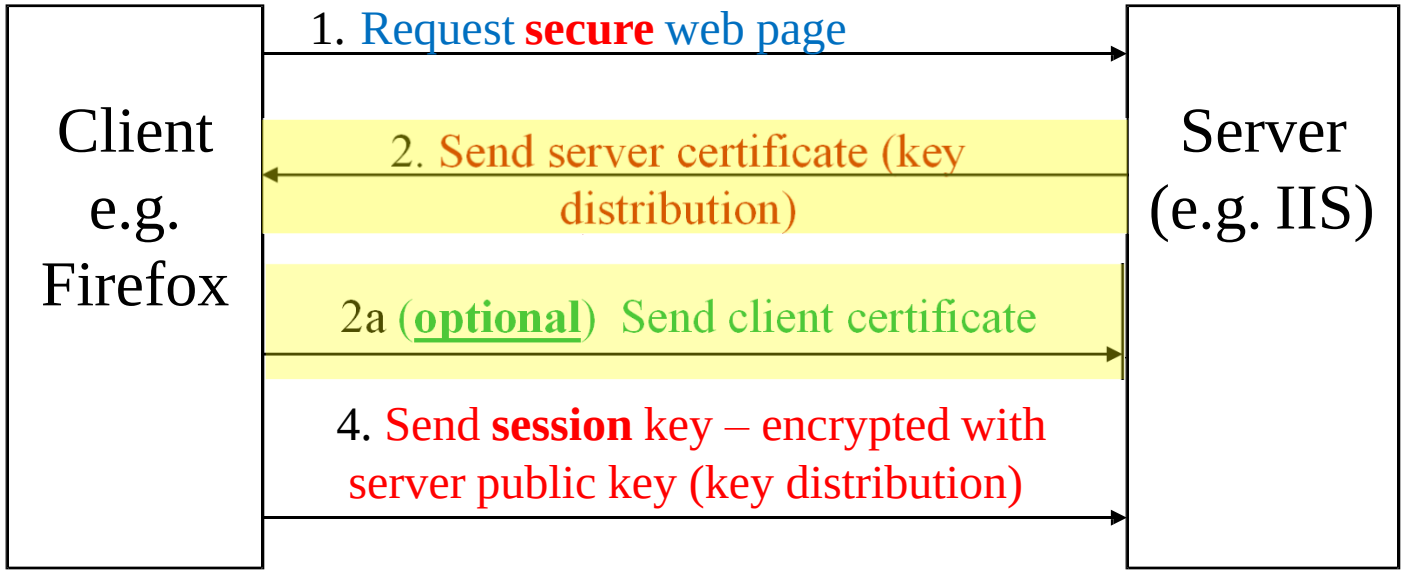
Does TLS do everything? NO!

- TLS **transmits data securely** – does not check transaction integrity/correctness!
  - Does not check card number
  - Does not check customer status
  - Does not complete the transaction
  - Does not separate information on a 'need to know' basis
- Frequently leads to design solutions where credit card numbers are stored on the merchant's server
- Does not separate our merchant and bank data

Now let's look at:

- Levels of authentication (partial or full – implications)
- Paradigms of security (it is hybrid, symmetric key and public key)
- Key distribution problem
- Certificate usage – private key usage to confirm ownership of certificate

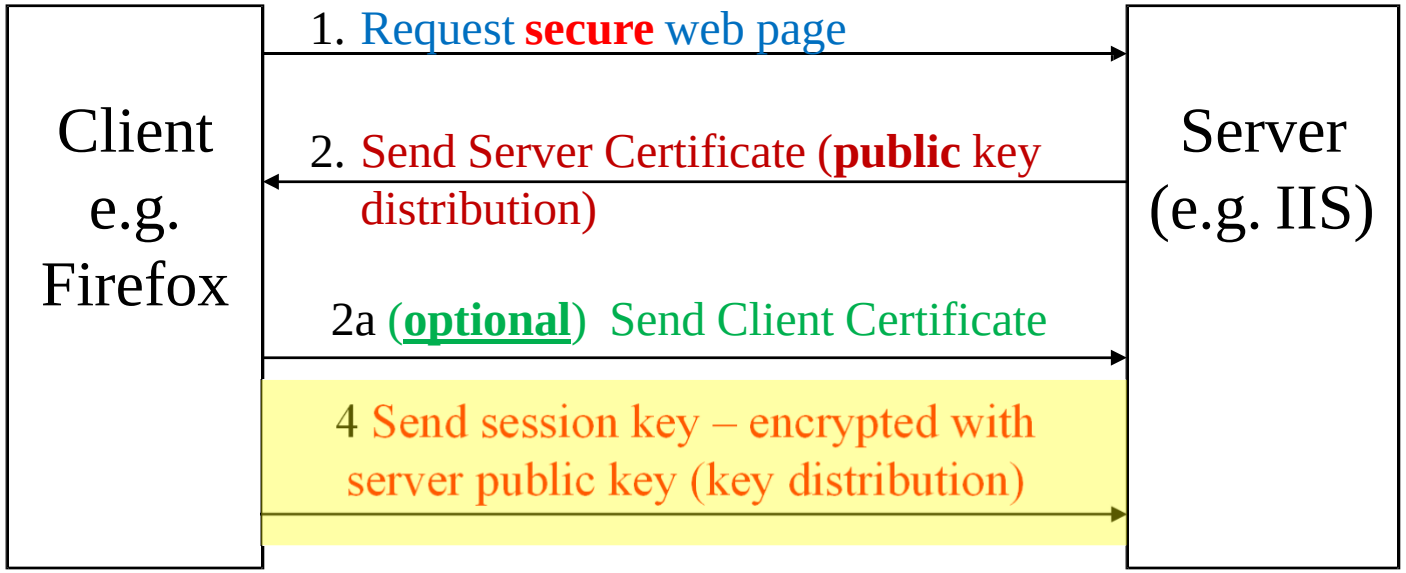
TLS: Levels of authentication – server (always) – client (rarely)



3. Validate server's certificate: signature via root certificate, URL matches name on certificate, validity from-to current. If not valid, user is warned. *Generate symmetric session key*

5. Server uses its private key to decrypt session key. Now both sides use this session key to exchange secured data. Session key typically 128 bits long. Server's public key typically 1024 bits long.

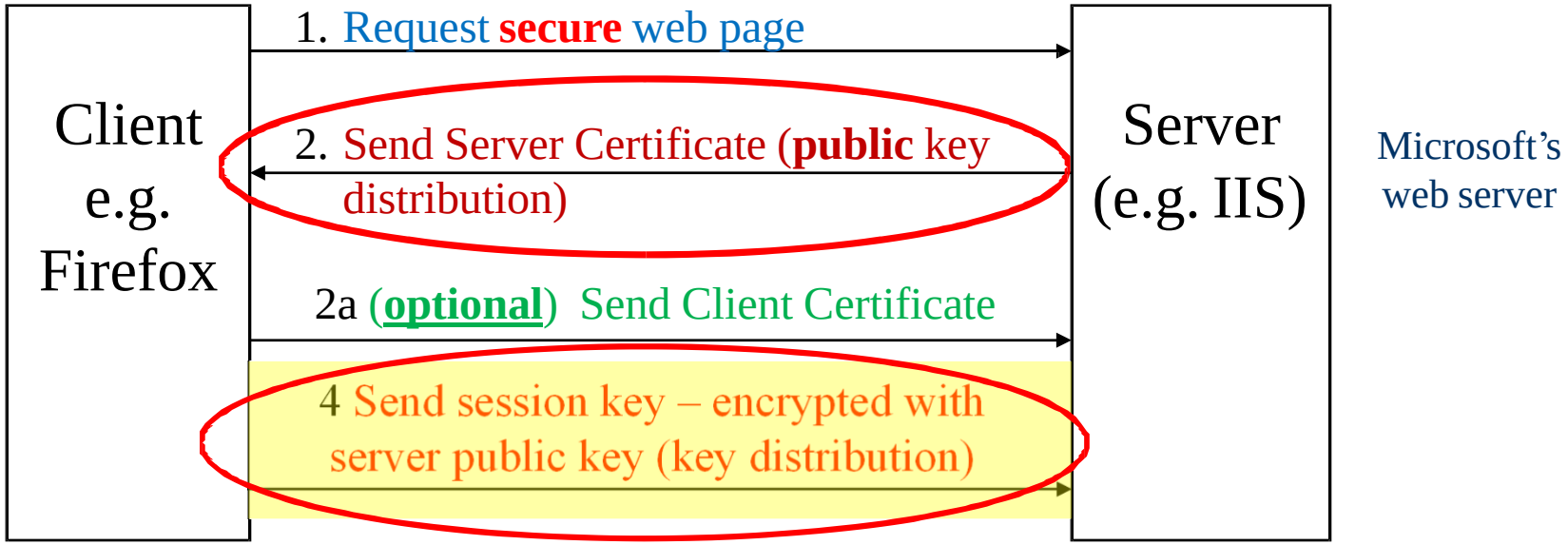
TLS: Paradigms of cryptography – asymmetric & symmetric (hybrid)



3. Validate server's certificate: signature via root certificate, URL matches name on certificate, validity from-to current. If not valid, user is warned. *Generate symmetric session key*

5. Server uses its private key to decrypt session key. Now both sides use this session key to exchange secured data. Session key typically 128 bits long. Server's public key typically 1024 bits long.

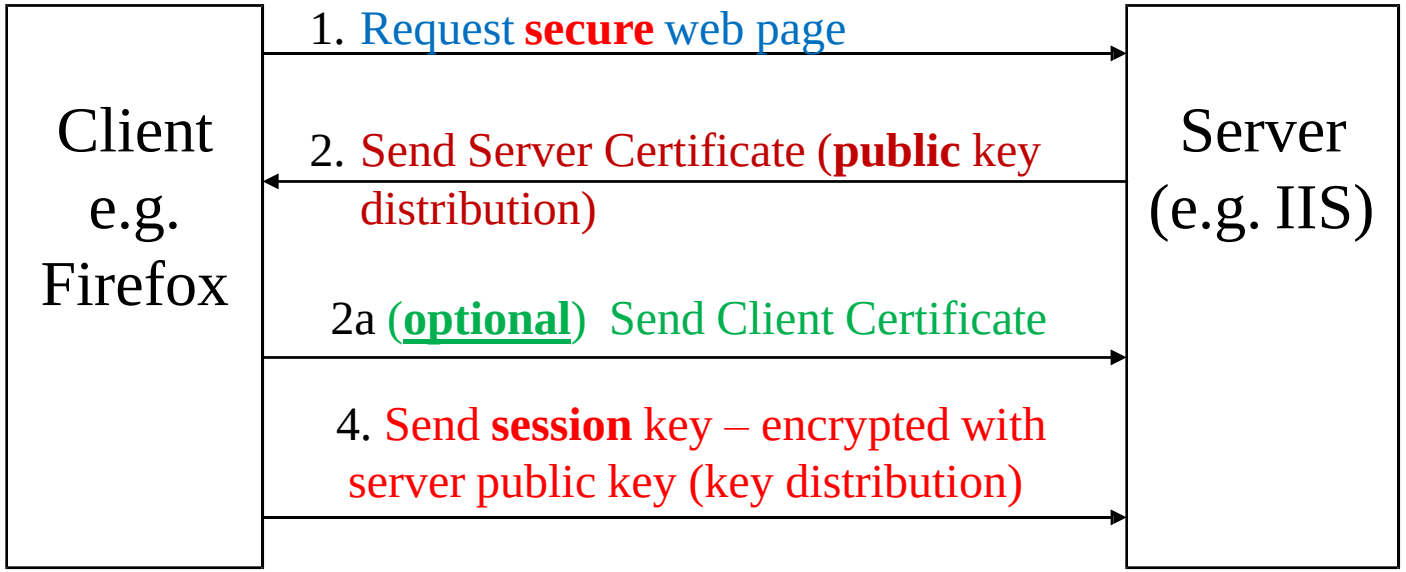
TLS: Key distribution problem



3. Validate server's certificate: signature via root certificate, URL matches name on certificate, validity from-to current. If not valid, user is warned. *Generate symmetric session key*

5. Server uses its private key to decrypt session key. Now both sides use this session key to exchange secured data. Session key typically 128 bits long. Server's public key typically 1024 bits long.

TLS: Certificate usage – server must use its private key



3. Validate server's certificate: signature via root certificate, URL matches name on certificate, validity from-to current. If not valid, user is warned. *Generate symmetric session key*

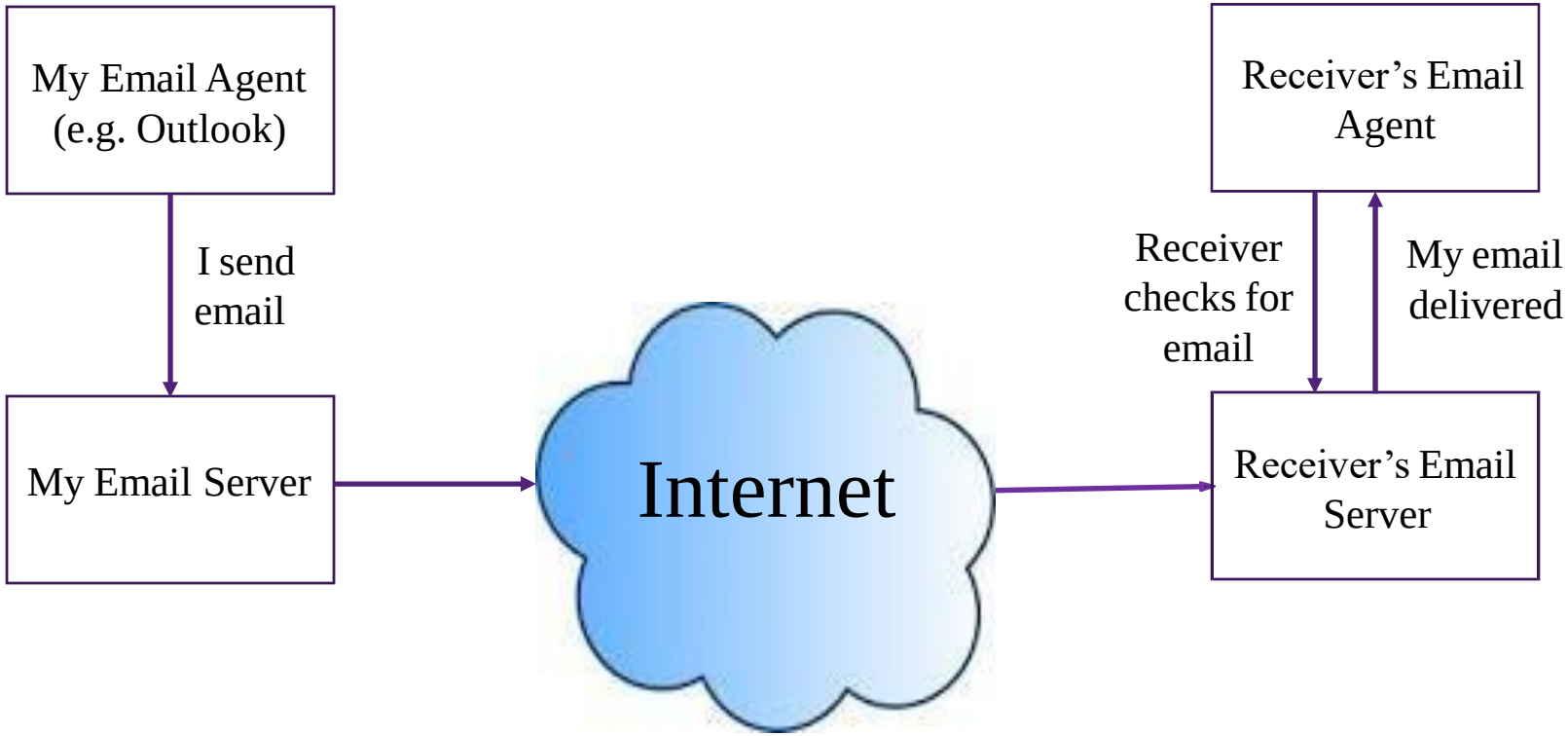
5. Server uses its private key to decrypt session key. Now both sides use this session key to exchange secured data. Session key typically 128 bits long. Server's public key typically 1024 bits long.

Web security provides to the user authentication (*on the server-side*), integrity (*step 2*) confidentiality (*step 4).* You always get those three. You can't use TLS and only want confidentiality and not care about authentication. You always get all three.

Email secure communication

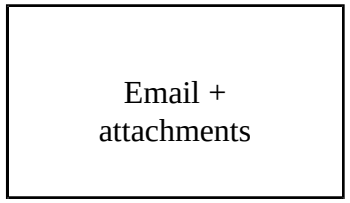
- S/MIME (Secure/Multipurpose Internet Mail Extensions) – Enables **end to end encryption** (***confidentiality***) and also authentication + integrity (user to user)
- PGP (Pretty Good Privacy) – Also enables end-to-end.
- Unlike TLS, we can have either confidentiality OR authentication/integrity OR all three together
- Both protocols (S/MIME PGP) work in a very similar fashion.
- Both protocols are offline.
- Our focus is on S/MIME
  - this protocol is attracting most corporate support (adopted by Microsoft)

Email is offline – therefor S/MIME is offline
(not communicating in 'real-time')

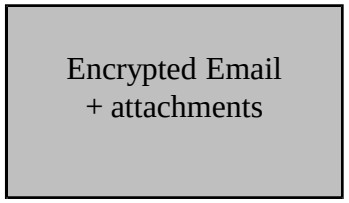


S/MIME cannot be designed the same way as TLS – S/MIME is offline (not real time) whilst TLS is online (real time)

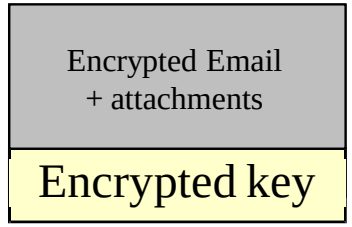
S/MIME – How it (logically) works



1. Compose email text and attachments.

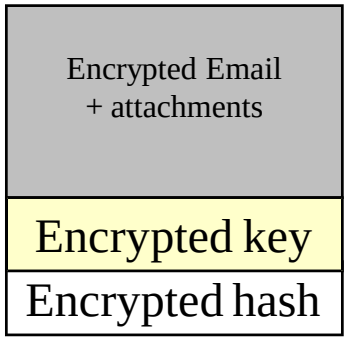


2. Create a symmetric session key – encrypt the email text and attachments with this key.



3. Encrypt the session key with the recipient's public key – and add this to the email.

My encrypted (using receiver's public key) symmetric session key I used to encrypt my message with.



4. Digital signature if authentication is required. Hash the total email (text, attachments, and encrypted key). Encrypt the resulting digest with the sender's private key. Add this encrypted hash to the email package. Send the lot to the recipient.

So if we want a digitally signed email, an authenticated and integrity-controlled email, all we need is our own certificate. If we want to send an email controlled for confidentiality, we need the receiver's certificate.

Questions: What keys/certificates are needed at the sender's end, and at the receiver's end? How is the key distribution problem solved? Is this a hybrid solution? Is this online or offline? How is the package processed at the receiver's end?

Secure Email options for our use (1)

Standard Gmail uses TLS for securing email. TLS is an open security protocol (very good) and it is the successor to SSL. But TLS only secures data between the sending client (usually a browser) and the receiving server (we stressed this in our course – TLS is very secure it is totally a 'transmission' security protocol). The data (in our case, an email) is not stored securely on the sending client – and it is not secured whilst on the receiving server. You could call this 'point to point'. It is certainly not 'end to end'. It does not, for example, guarantee that the message will remain private or available only to the intended recipient once it reaches the destination mail server. Google itself can see messages associated with your account, which is what allows it to scan your email for potential spam and phishing attacks (and show you related ads).

Beyond that basic form of encryption, Gmail supports an enhanced standard known as S/MIME — or Secure/Multipurpose Internet Mail Extensions (explained next paragraph). You have to supply a certificate that may cost money. S/MIME allows emails to be encrypted with user-specific keys so that they remain protected during delivery and can be decrypted only by the intended recipient. This is the meaning of 'end to end' – the sender (e.g. you/me) is one end. The receiver (i.e. the person – not the email server) is the other end.

Secure Email options for our use (2)

S/MIME is very secure – it actually 'triple wraps' email data. When we use S/MIME we have the option to secure our email for integrity/authentication (digital signing) or for confidentiality (via encryption) OR BOTH. If we want to secure our email for confidentiality, we do the following: (1) compose the email; (2) our email agent (e.g. Outlook) then generates a symmetric session (one time) key and uses this key to encrypt our email; (3) our email agent then uses the public key of the intended recipient to encrypt the session key – and this encrypted result is added to the email. This is then sent to the recipient who will then use her/his private key to decrypt the session key – and then use the session key to decrypt the actual email.

This 'end to end' encryption ensures nobody (in transit) can view the actual email content. (Actually the 'triple wrap' does a little more – but that is not important here). S/MIME is now very popular within corporates and government departments. However S/MIME does present 'key management' challenges (because nobody – other that sender/receiver) can access the emails – and because the sender needs the public key (via the digital certificate) of the intended receiver.

Secure Email options for our use (3)

PGP has been around since the early 1990s and it has always been highly respected. At a high level of analysis, it is similar to S/MIME (or probably better to say S/MIME is similar to PGP). The message is encrypted via a symmetric (session) key generated by the sender. The message and its session key are sent to the receiver. The session key is protected during transmission because it is encrypted with the receiver's public key. PGP is an excellent email security system – however it seems to have 'lost out' to S/MIME (especially when Microsoft adopted S/MIME for its email system Outlook).

ProtonMail is a relative newcomer. It is very well designed at many levels to achieve high security. It also uses 'end to end' security for its email content – achieved via a combination of public and symmetric key crypto (like S/MIME and PGP). It also offers high level of security for user authentication (i.e. logging onto email accounts). Possibly the only area of criticism of ProtonMail is that it has (in the past years) been subject to concentrated distributed denial of service attacks that have been designed to 'complicate' user access to email and accounts.


```
        end;

    write(Endtext);
end.
CREATE TABLE product(
    class MultinomialNB(object):
    def __init__(self):
2})
        self.X = None
        self.y = None
    def __loading(self):
        self.list_labels = cl.Counter(
int acc(std::function<int(int, int)>
    auto it = operands.begin();
    int result = func(*it, *(++it));
    if (operands.size() > 2) {
        for (++it; it!=operands.end();
            result = func(result, *it);
        }
    }
    return result;
```

Thank you