Security

Web Applications and Services

Spring Term

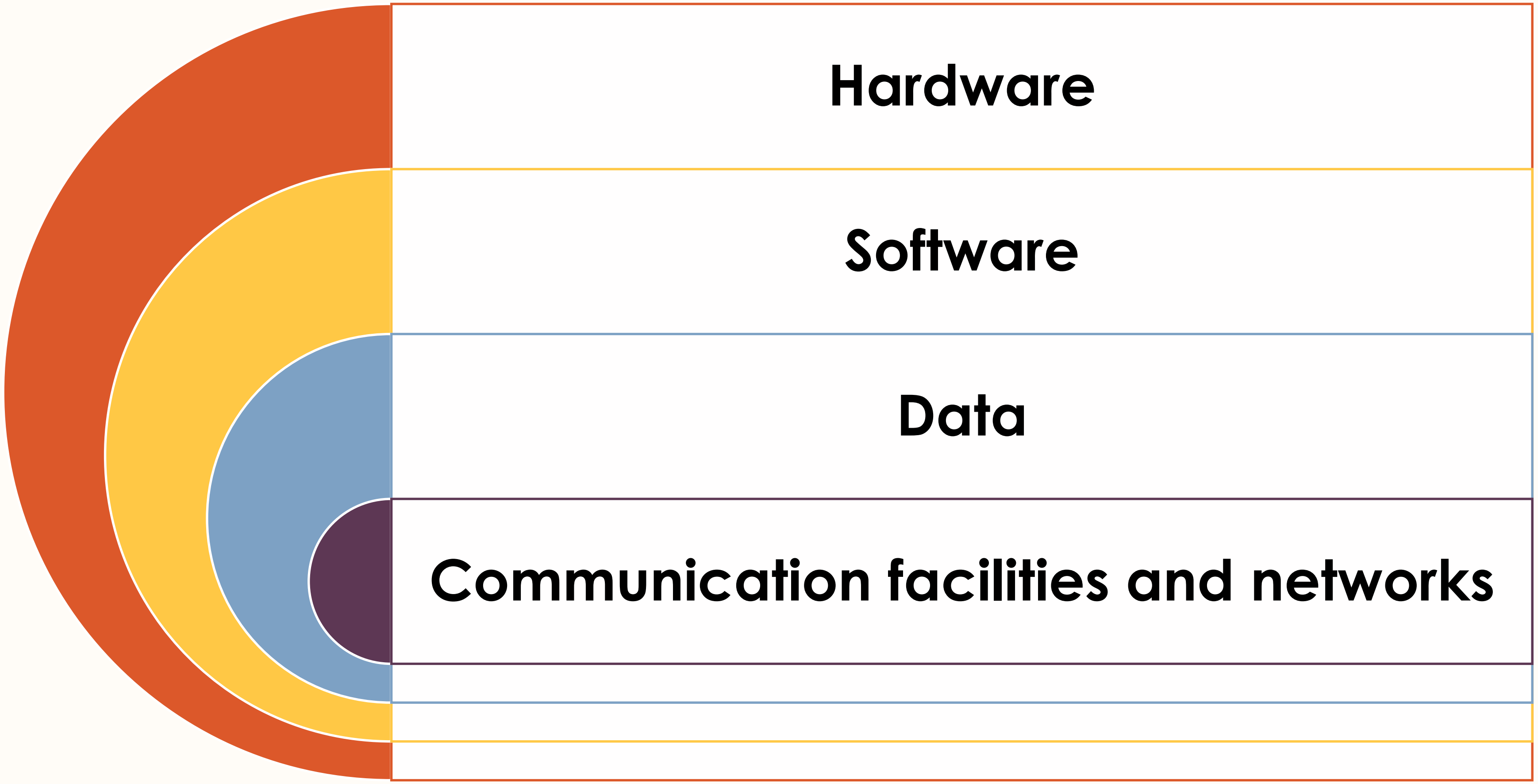
Naercio Magaia

UNIVERSITY
OF SUSSEX

Contents

- Introduction
- Symmetric Encryption
- Cryptographic Hash Function
- Public-Key Encryption
- Digital Signatures
- Role-Based Access Control
- Transport-Level Security
- Security in Django

Assets of a Computer System



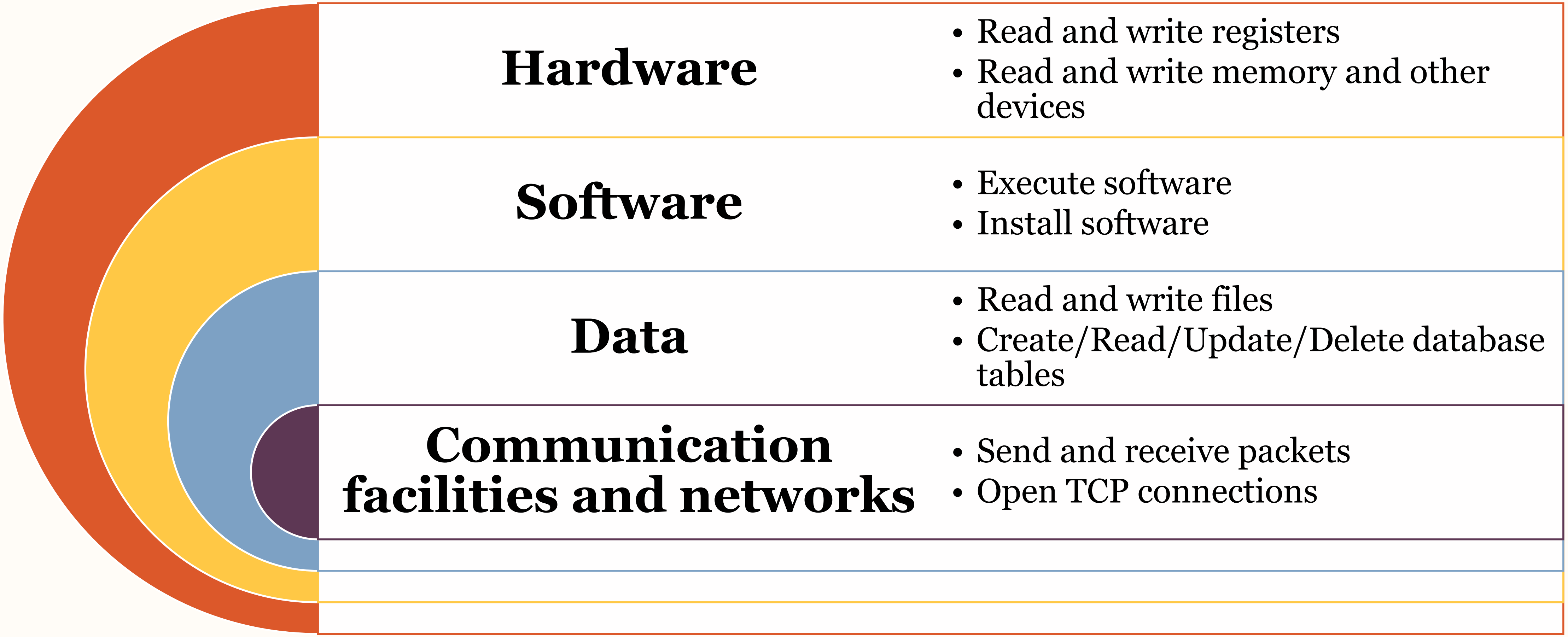
Hardware

Software

Data

Communication facilities and networks

What can be done with them?



Hardware

- Read and write registers
- Read and write memory and other devices

Software

- Execute software
- Install software

Data

- Read and write files
- Create/Read/Update/Delete database tables

Communication facilities and networks

- Send and receive packets
- Open TCP connections

Key Security Concepts

Confidentiality

- Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information

Integrity

- Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity

Availability

- Ensuring timely and reliable access to and use of information

UNIVERSITY OF SUSSEX

Key Security Concepts

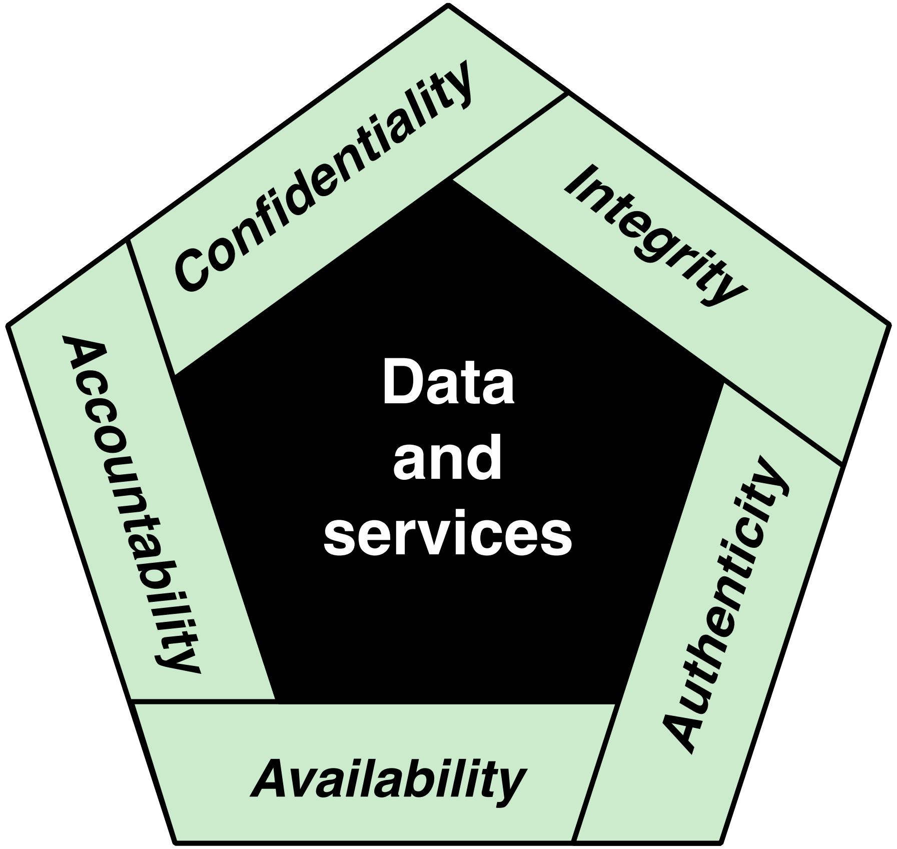
Authenticity

- The property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, message or message originator. Requires verifying users checking the origin of each input

Accountability

- Providing the capability of actions being traced to their originator. Supports nonrepudiation, deterrence, fault isolation, intrusion detection and prevention. Records are kept to provide post-attack analysis and meet legal requirements

Essential Security Requirements



UNIVERSITY OF SUSSEX

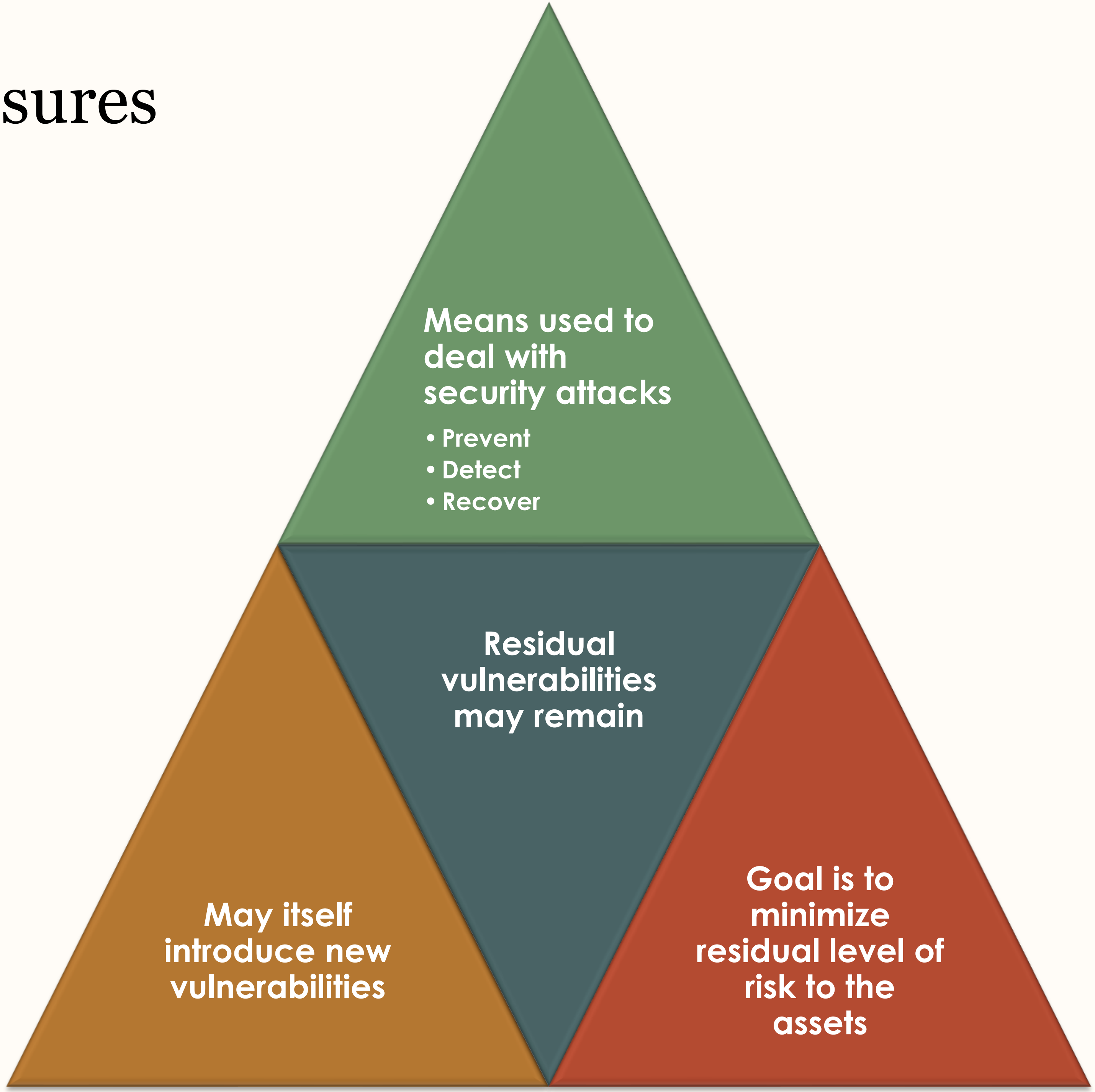
Vulnerabilities, Threats, and Attacks

- Categories of vulnerabilities
  - Corrupted (loss of integrity)
  - Leaky (loss of confidentiality)
  - Unavailable or very slow (loss of availability)
- Threats
  - Capable of exploiting vulnerabilities
  - Represent potential security harm to an asset

Vulnerabilities, Threats, and Attacks

- Attacks (threats carried out)
  - Passive – attempt to learn or make use of information from the system that does not affect system resources
  - Active – attempt to alter system resources or affect their operation
  - Insider – initiated by an entity inside the security parameter
  - Outsider – initiated from outside the perimeter

Countermeasures



Means used to deal with security attacks

- Prevent
- Detect
- Recover

Residual vulnerabilities may remain

May itself introduce new vulnerabilities

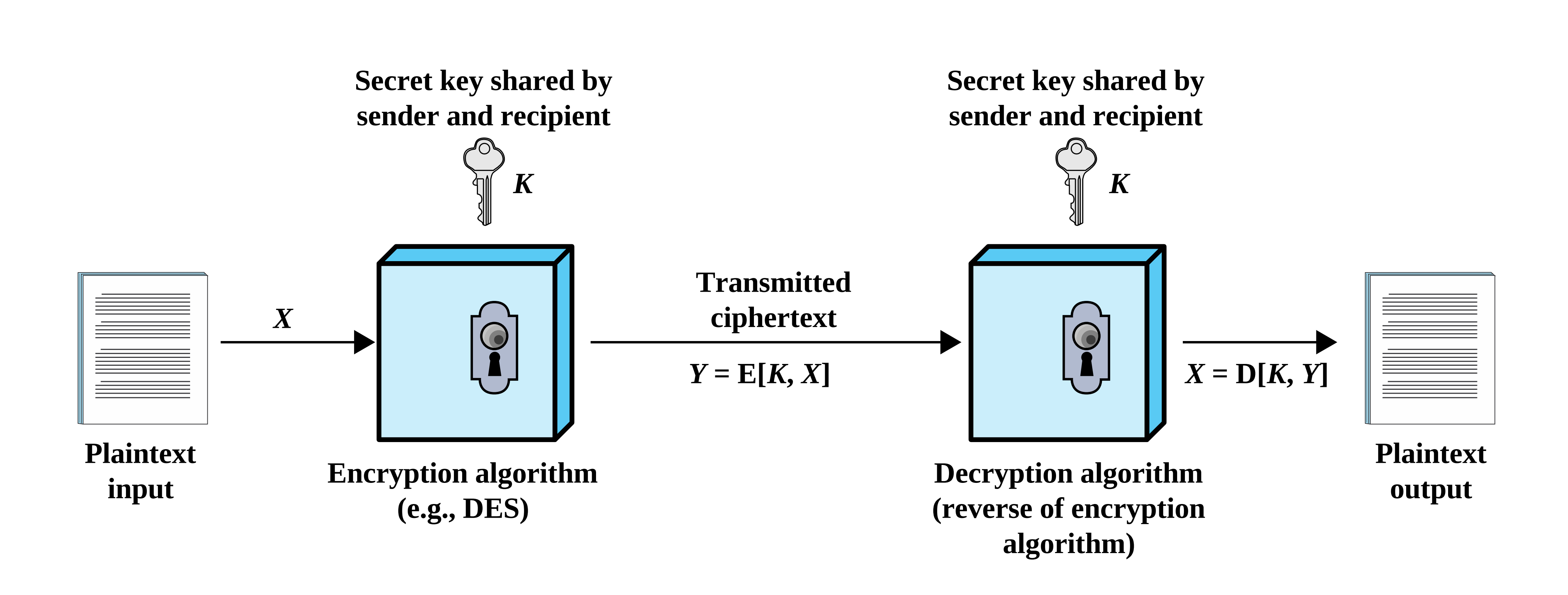
Goal is to minimize residual level of risk to the assets

Symmetric Encryption

- The universal technique for providing confidentiality for transmitted or stored data
- Also referred to as **conventional** or **single-key** encryption
- Two requirements for its secure use:
  - Need a strong encryption algorithm
  - Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure

- **Plaintext**: The original message fed into the algorithm
- **Encryption algorithm**: The rules for substituting and transforming the plaintext
- **Secret Key**: The sequence of bytes that guides the encryption algorithm
- **Ciphertext**: The scrambled message out of the algorithm
- **Decryption algorithm**: The rules for substituting and transforming the ciphertext to return the plaintext

Information Flow


Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E[*K*, *X*]
X = D[*K*, *Y*]
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

UNIVERSITY OF SUSSEX

Data Encryption Standard (DES)

- The first widely used encryption scheme
  - FIPS PUB 46 (January 1977)
  - Referred to as the Data Encryption Algorithm (DEA)
  - Uses 64 bits plaintext block and 56 bit key to produce a 64 bits ciphertext block
- Strength concerns about
  - the algorithm itself
    - DES is the **most studied** encryption algorithm in existence
  - the use of a 56-bit key
    - The speed of commercial off-the-shelf processors makes this key length woefully inadequate

Triple DES (3DES)

- Repeats basic DES algorithm three times using either two or three unique keys
- First standardized for use in financial applications in ANSI standard X9.17 in 1985
- Attractions:
  - 168-bit key length overcomes the vulnerability to brute-force attack of DES
  - Underlying encryption algorithm is the same as in DES
- Drawbacks:
  - Algorithm is sluggish in software
  - Uses a 64-bit block size

Advanced Encryption Standard (AES)

Needed a replacement for 3DES

3DES was not reasonable for long term use

NIST called for proposals for a new AES in 1997

- Should have a security strength equal to or better than 3DES
- Significantly improved efficiency
- Symmetric block cipher
- 128 bits data and 128/192/256 bits keys

Selected Rijndael in November 2001

Published as FIPS 197

Practical Security Issues

- Typically, symmetric encryption is applied to a unit of data larger than a single 64-bit or 128-bit block
- Electronic codebook (ECB) mode is the simplest approach to multiple-block encryption
  - Each block of plaintext is encrypted using the same key
  - Cryptanalysts may be able to exploit regularities in the plaintext
  - Can be massively parallelised
- Modes of operation
  - Alternative techniques (i.e., Chaining and Streaming) developed to increase the security of symmetric block encryption for large sequences
  - Overcomes the weaknesses of ECB

Block & Stream Ciphers

Block Cipher

- Processes the input one block of elements at a time
- Produces an output block for each input block
- Identical blocks generate identical output
- More common

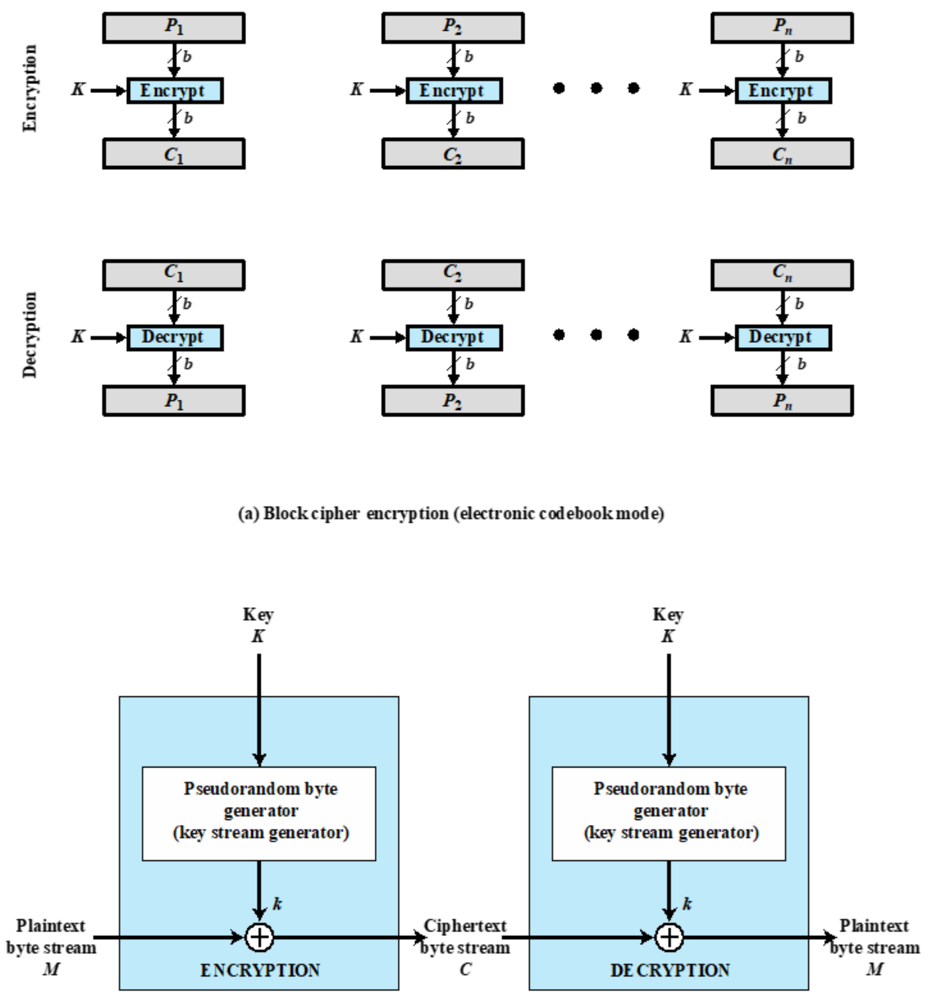
Cipher Block Chaining

- Process one block at a time
- Use the output of the current block XORed with the input of the next block
- Identical blocks generate different output

Stream Cipher

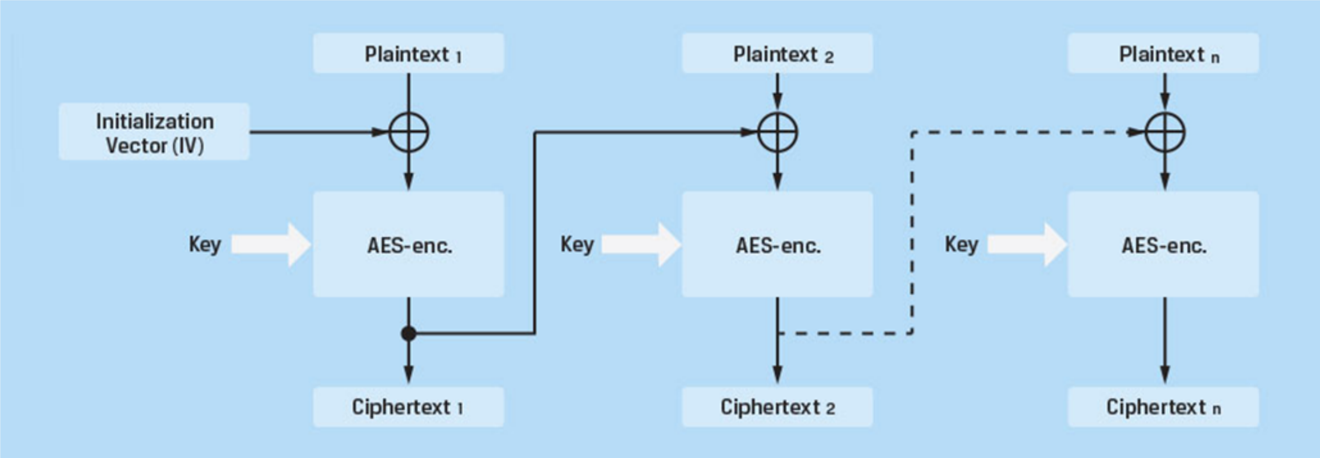
- Processes the input elements continuously
- Produces output one element at a time
- Primary advantage is that they are almost always faster and **use far less code**
- Encrypts plaintext **one byte at a time**
- **Pseudorandom stream** is one that is unpredictable without knowledge of the input key

Types of Symmetric Encryption



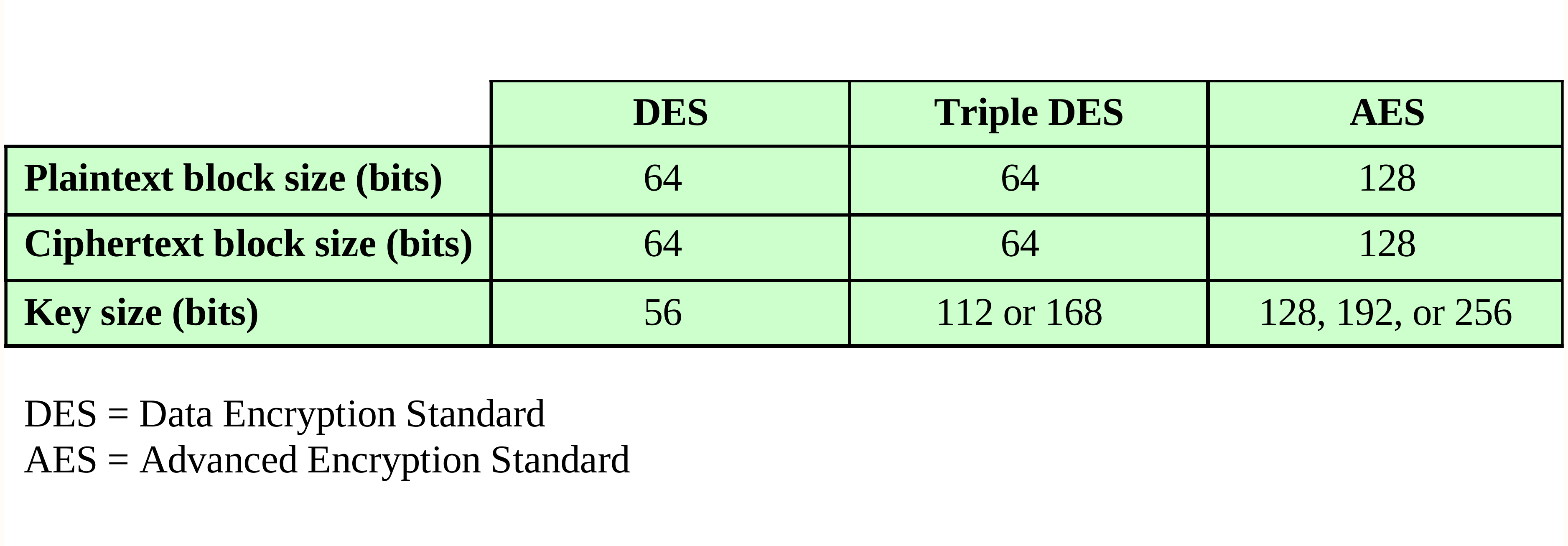
(a) Block cipher encryption (electronic codebook mode)

(b) Stream encryption



UNIVERSITY OF SUSSEX

Comparison of SE Algorithms



	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

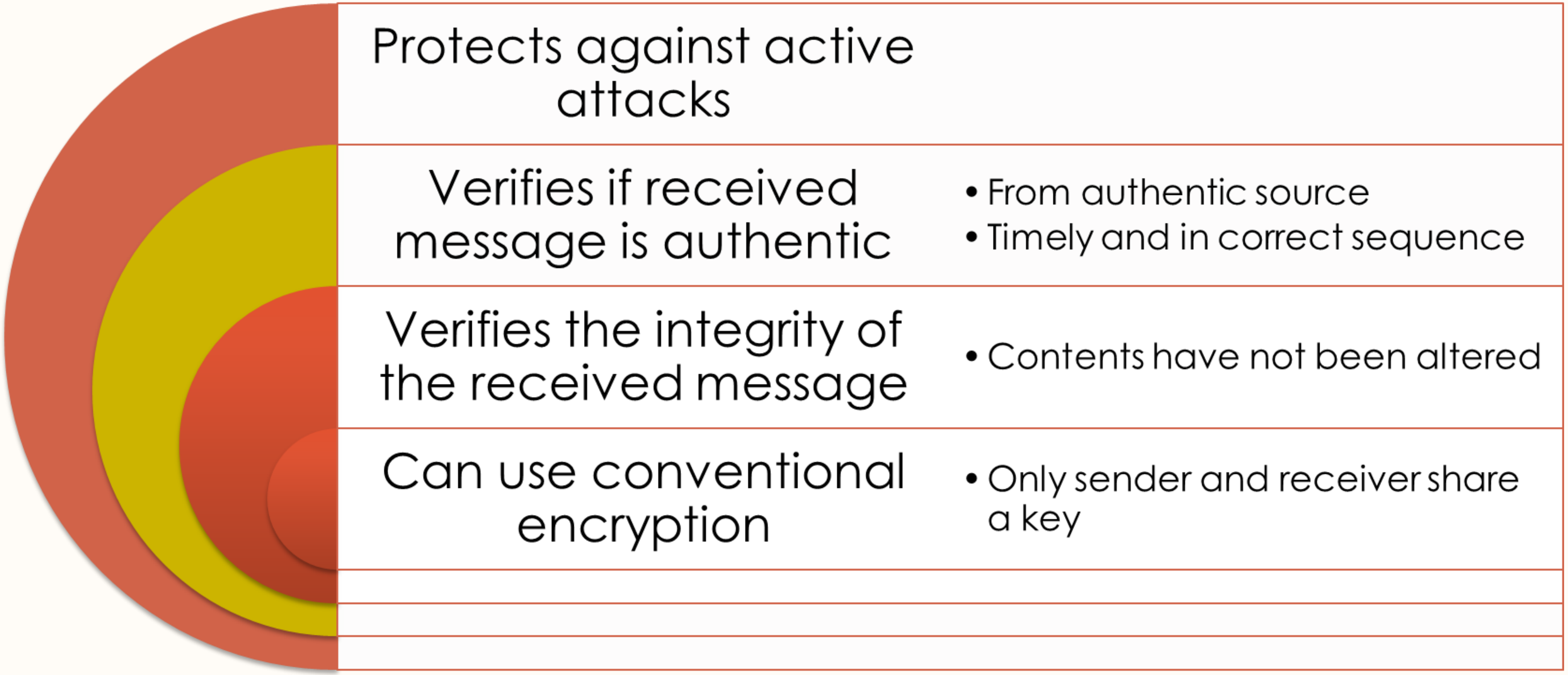
DES = Data Encryption Standard
AES = Advanced Encryption Standard

Average Time Required for Exhaustive Key Search

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/s	Time Required at 10^{13} decryptions/s
56	DES	2^{56} ≈ 7.2 ´ 10^{16}	2^{55} ns = 1.125 years	1 hour
128	AES	2^{128} ≈ 3.4 ´ 10^{38}	2^{127} ns = 5.3 ´ 10^{21} years	5.3 ´ 10^{17} years
168	Triple DES	2^{168} ≈ 3.7 ´ 10^{50}	2^{167} ns = 5.8 ´ 10^{33} years	5.8 ´ 10^{29} years
192	AES	2^{192} ≈ 6.3 ´ 10^{57}	2^{191} ns = 9.8 ´ 10^{40} years	9.8 ´ 10^{36} years
256	AES	2^{256} ≈ 1.2 ´ 10^{77}	2^{255} ns = 1.8 ´ 10^{60} years	1.8 ´ 10^{56} years

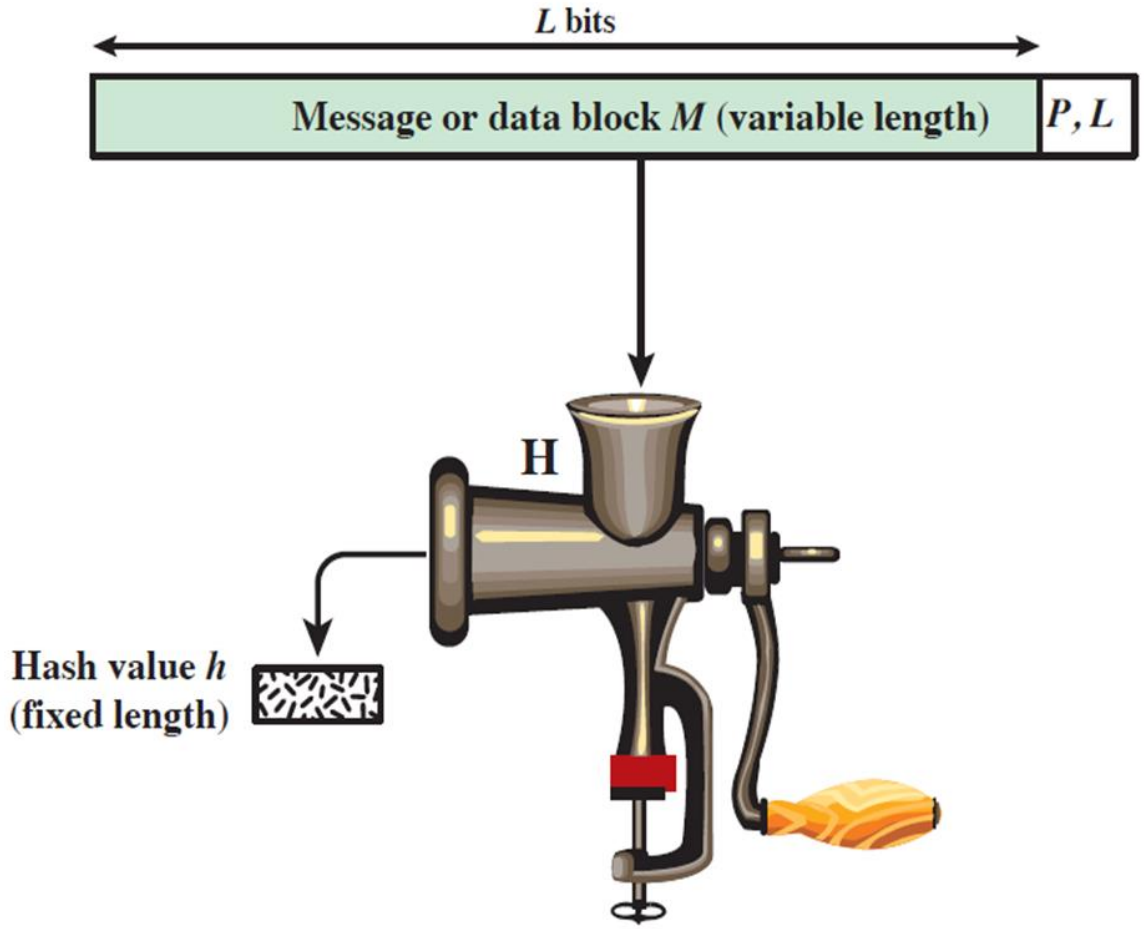
UNIVERSITY
OF SUSSEX

Message Authentication



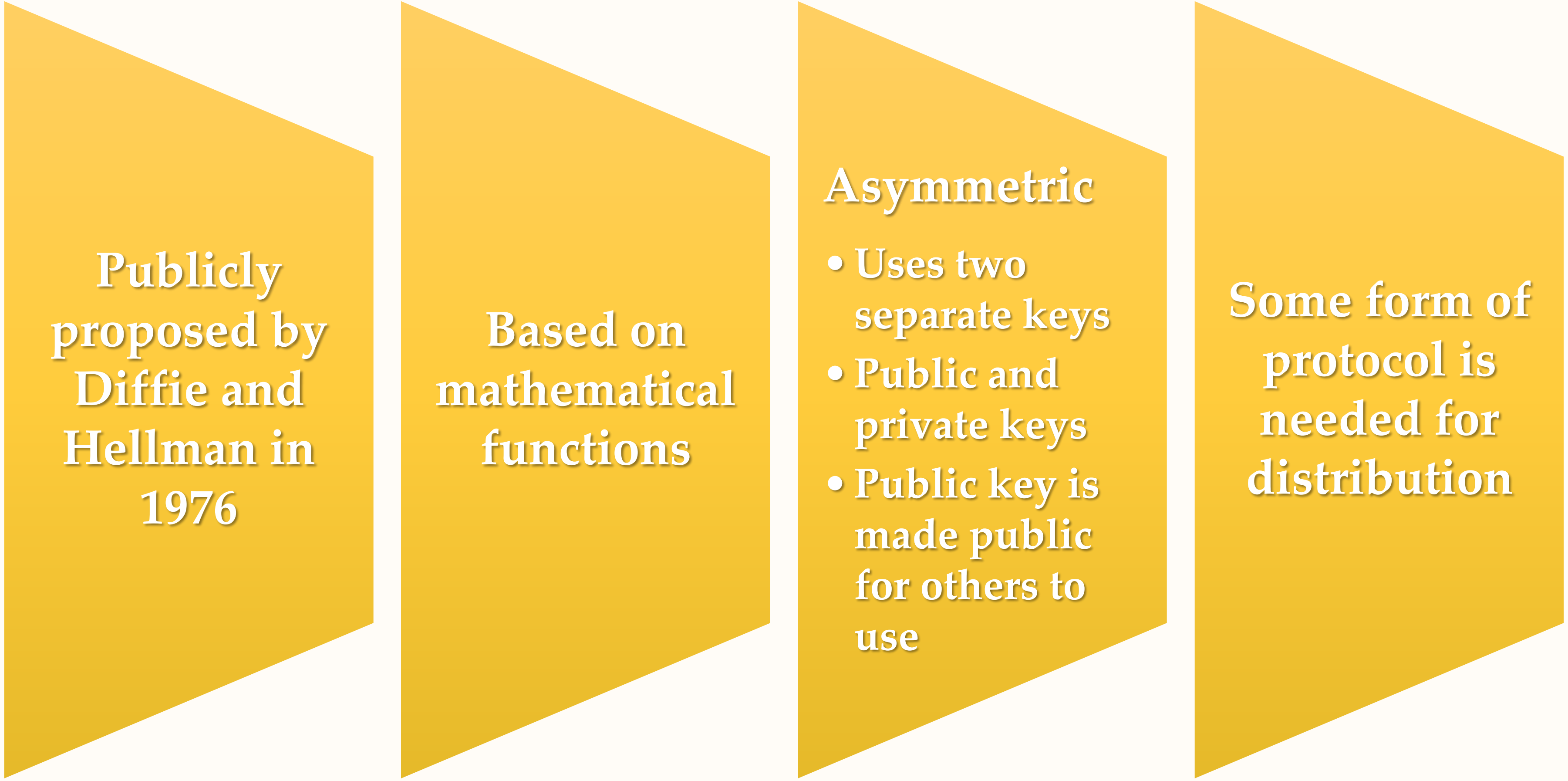
Protects against active attacks	
Verifies if received message is authentic	- From authentic source - Timely and in correct sequence
Verifies the integrity of the received message	- Contents have not been altered
Can use conventional encryption	- Only sender and receiver share a key

Cryptographic Hash Function



P, *L* = padding plus length field

Public-Key Encryption Structure



Publicly proposed by Diffie and Hellman in 1976

Based on mathematical functions

Asymmetric

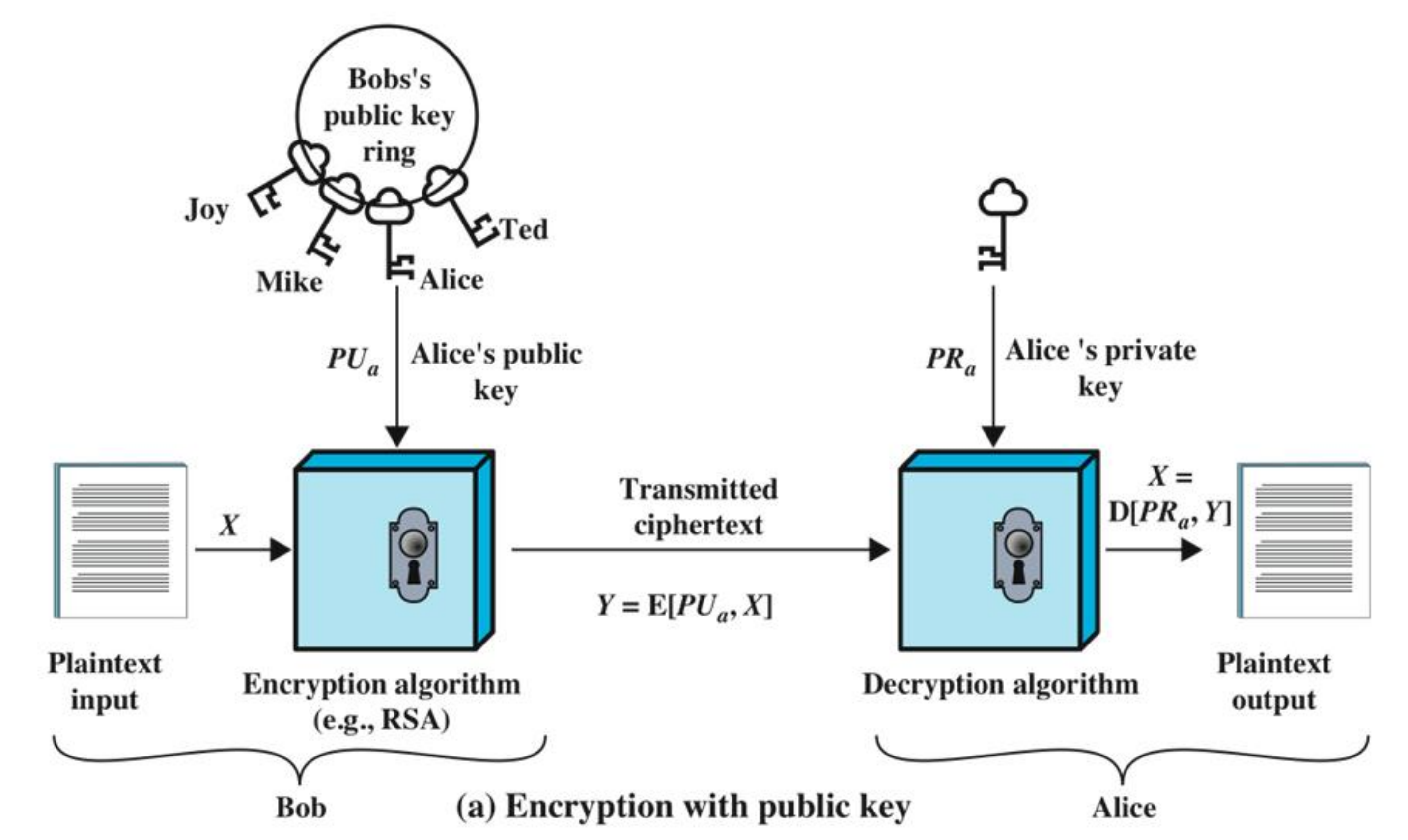
- Uses two separate keys
- Public and private keys
- Public key is made public for others to use

Some form of protocol is needed for distribution

Public-Key Encryption

- Plaintext
  - Readable message or data that is fed into the algorithm as input
- Encryption algorithm
  - Performs transformations on the plaintext
- Public and private key
  - Pair of keys, one for encryption, one for decryption
- Ciphertext
  - Scrambled message produced as output
- Decryption algorithm
  - Produces the original plaintext

Information Flow

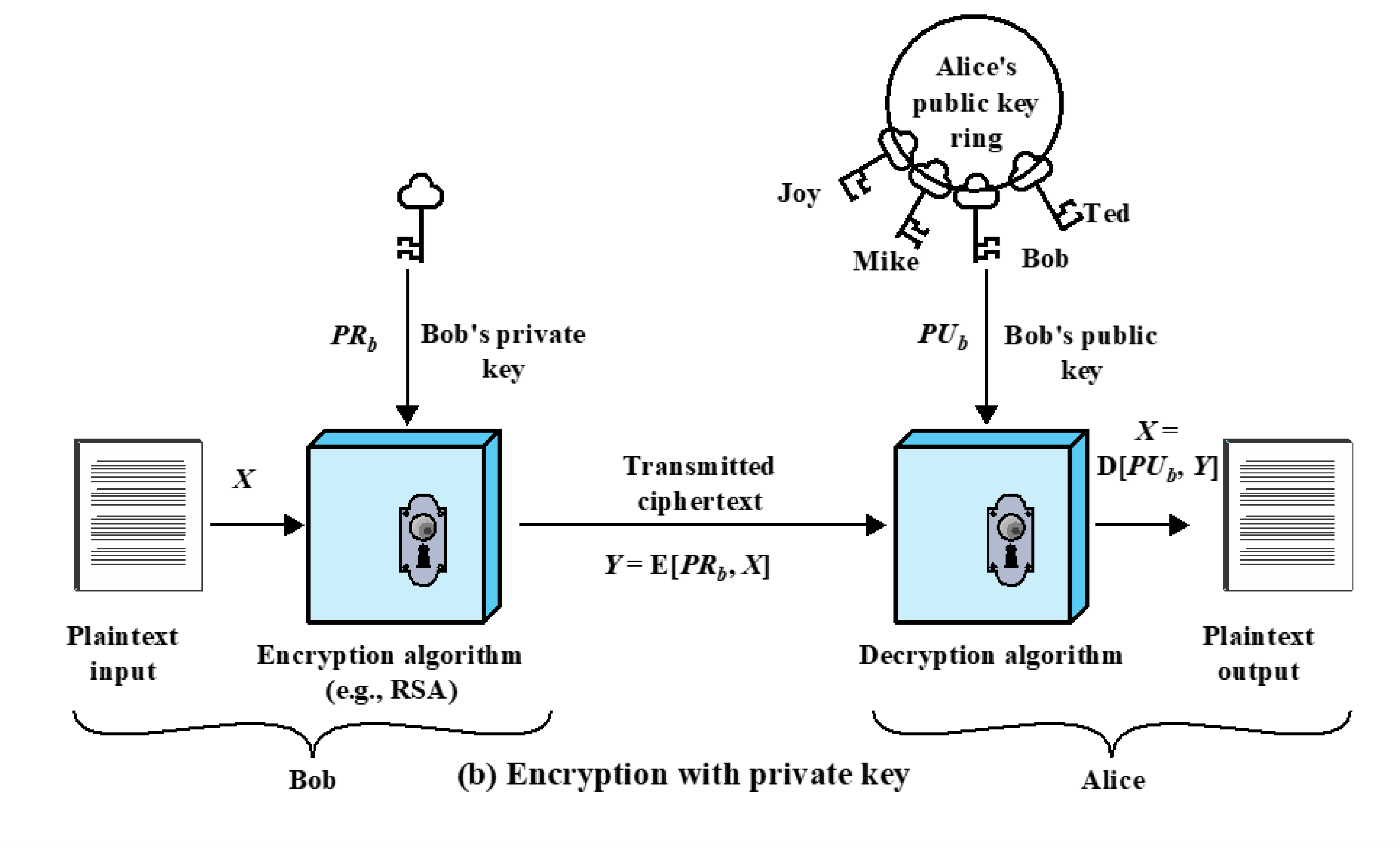


(a) Encryption with public key

Provides Confidentiality

Why?

Information Flow

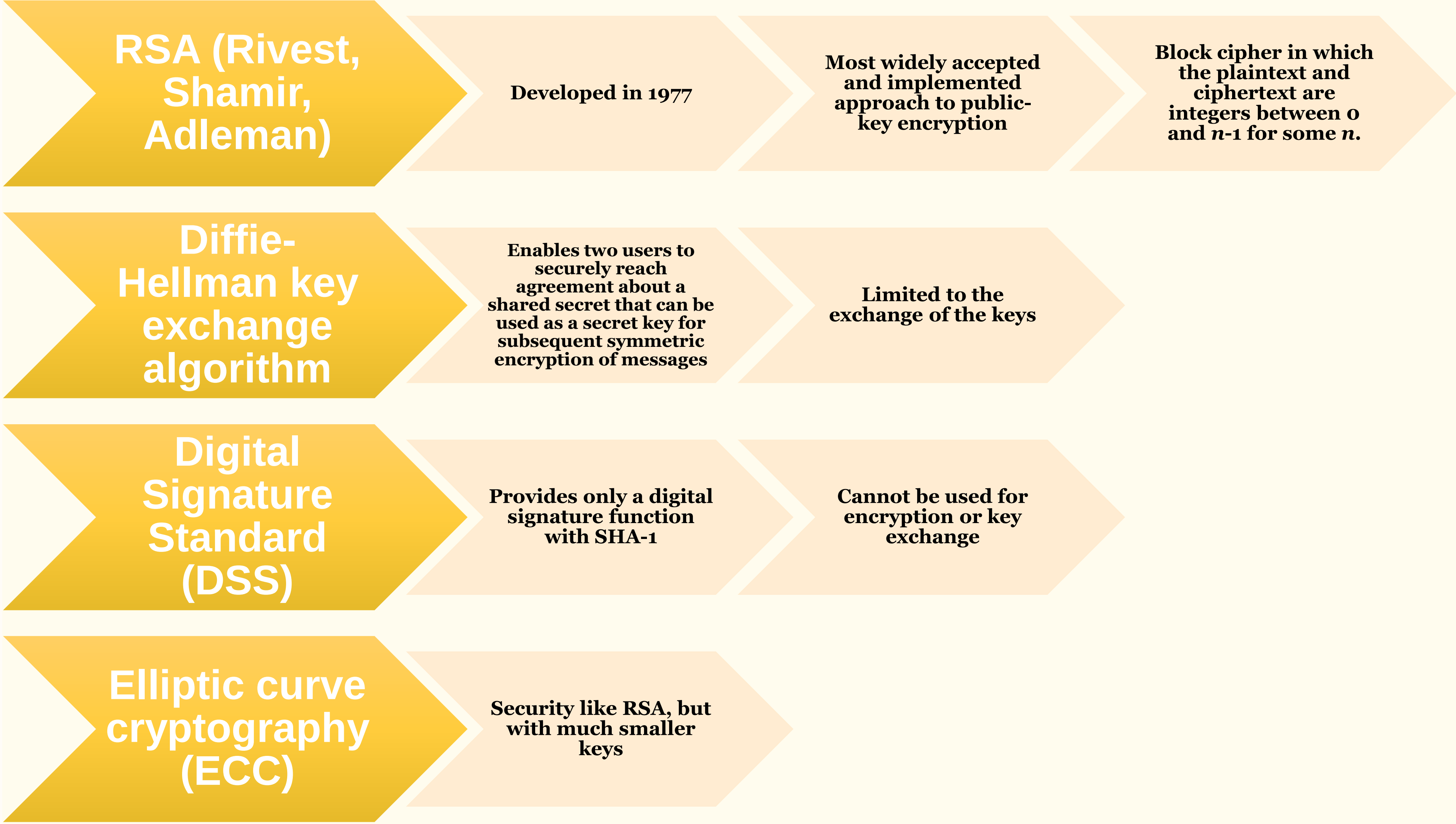


(b) Encryption with private key

Provides Authentication & Data Integrity

Why?

Asymmetric Encryption Algorithms



RSA (Rivest, Shamir, Adleman)

- Developed in 1977
- Most widely accepted and implemented approach to public-key encryption
- Block cipher in which the plaintext and ciphertext are integers between 0 and *n*-1 for some *n*.

Diffie-Hellman key exchange algorithm

- Enables two users to securely reach agreement about a shared secret that can be used as a secret key for subsequent symmetric encryption of messages
- Limited to the exchange of the keys

Digital Signature Standard (DSS)

- Provides only a digital signature function with SHA-1
- Cannot be used for encryption or key exchange

Elliptic curve cryptography (ECC)

- Security like RSA, but with much smaller keys

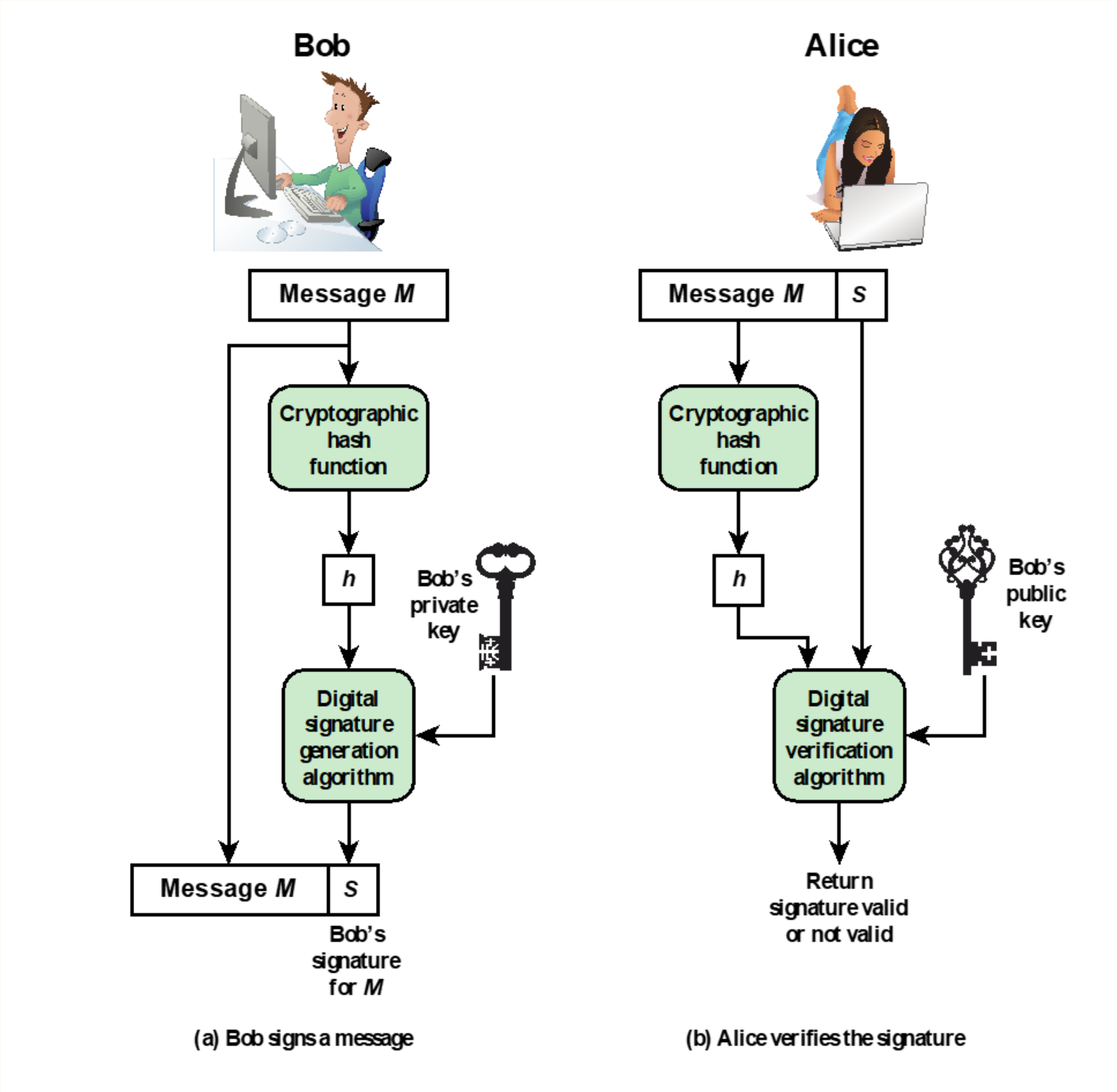
Digital Signatures

- NIST FIPS PUB 186-4 defines a digital signature as:

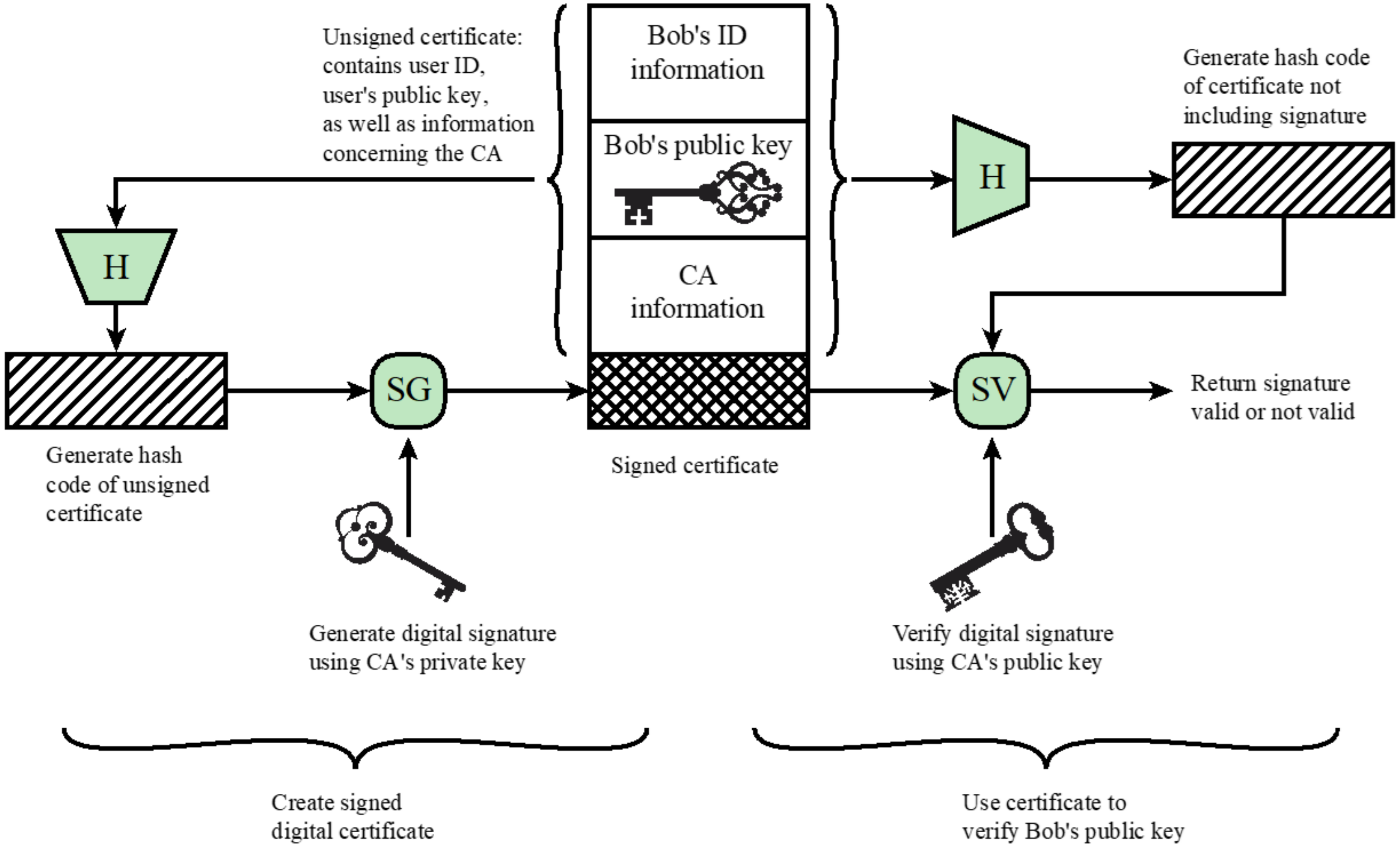
  > “The result of a cryptographic transformation of data that, when properly implemented, provides a mechanism for verifying origin authentication, data integrity and signatory non-repudiation.”

- Thus, a digital signature is a data-dependent bit pattern, generated by an agent as a function of a file, message, or other form of data block
- FIPS 186-4 specifies the use of one of three digital signature algorithms:
  - Digital Signature Algorithm (DSA)
  - RSA Digital Signature Algorithm
  - Elliptic Curve Digital Signature Algorithm (ECDSA)

Digital Signatures Process


Bob
Alice
Message M
Cryptographic hash function
h
Bob's private key
Digital signature generation algorithm
Message M
S
Bob's signature for M
(a) Bob signs a message
Message M
S
Cryptographic hash function
h
Bob's public key
Digital signature verification algorithm
Return signature valid or not valid
(b) Alice verifies the signature

Public-Key Certificate


Unsigned certificate:
contains user ID,
user's public key,
as well as information
concerning the CA
Bob's ID information
Bob's public key
CA information
H
Generate hash code of certificate not including signature
H
Generate hash code of unsigned certificate
SG
Signed certificate
SV
Return signature valid or not valid
Generate digital signature using CA's private key
Verify digital signature using CA's public key
Create signed digital certificate
Use certificate to verify Bob's public key

X.509

- Specified in RFC 5280
- The most widely accepted format for public-key certificates
- Certificates are used in most network security applications, including:
  - IP security (IPSEC)
  - Secure sockets layer (SSL)
  - Secure electronic transactions (SET)
  - S/MIME
  - eBusiness applications

X.509

- A number of specialized variants also exist, distinguished by particular element values or the presence of certain extensions
  - Conventional (long-lived) certificates
    - CA and “end user” certificates
    - Typically issued for validity periods of months to years
  - Short-lived certificates
    - Used to provide authentication for applications such as grid computing, while avoiding some of the overheads and limitations of conventional certificates
    - They have validity periods of hours to days, which limits the period of misuse if compromised
    - Because they are usually not issued by recognized CAs there are issues with verifying them outside their issuing organization

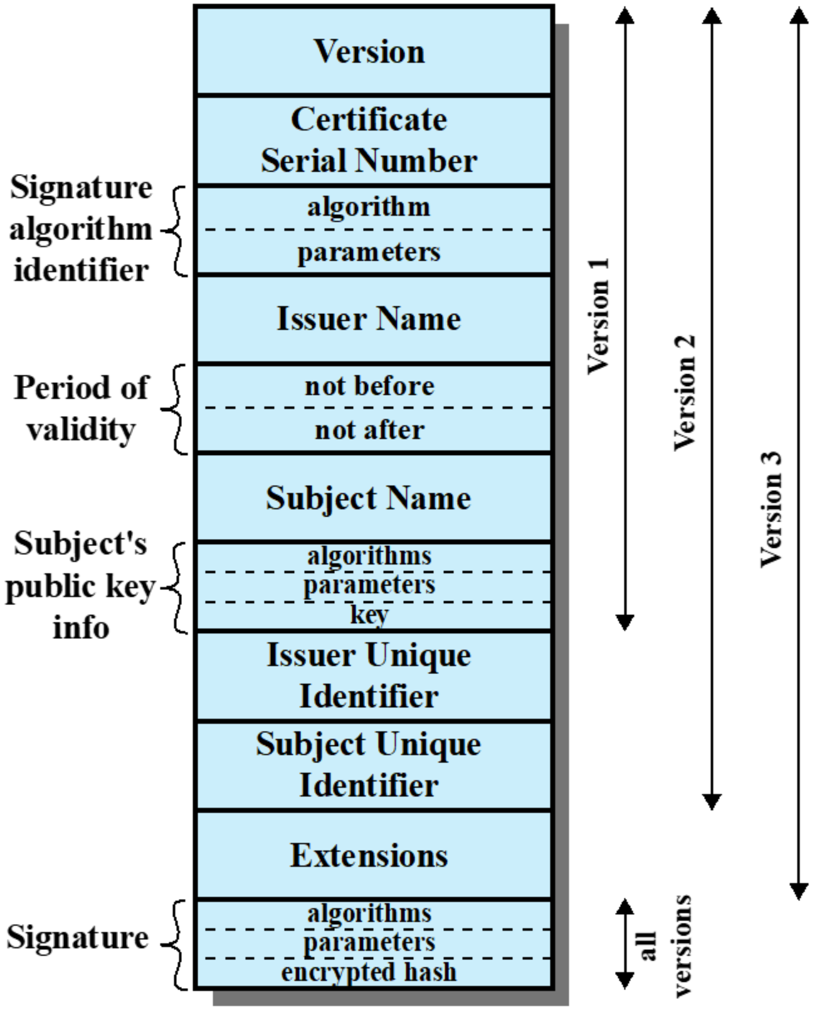
X.509

- A number of specialized variants also exist, distinguished by particular element values or the presence of certain extensions
  - Proxy certificates
    - Widely used to provide authentication for applications such as grid computing, while addressing some of the limitations of short-lived certificates
    - Defined in RFC 3820
    - Identified by the presence of the “proxy certificate” extension
    - They allow an “end user” certificate to sign another certificate
    - Allow a user to easily create a credential to access resources in some environment, without needing to provide their full certificate and right

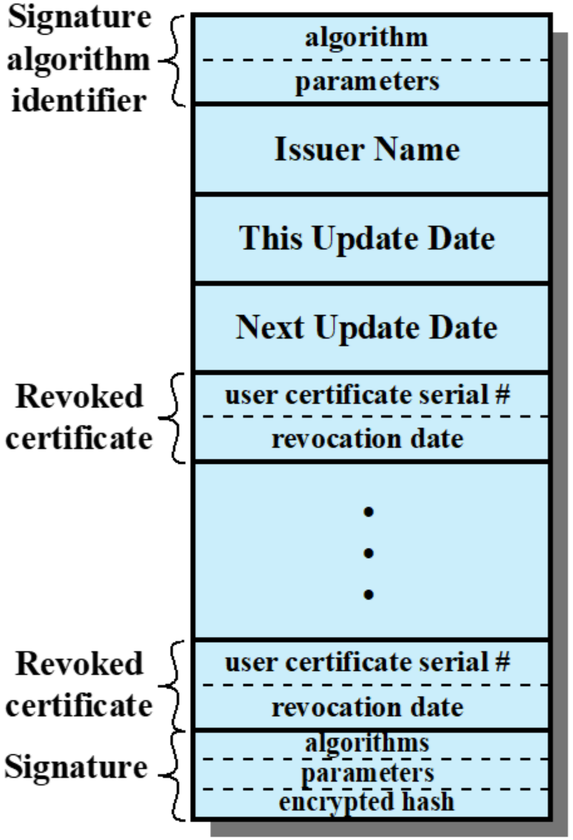
X.509

- A number of specialized variants also exist, distinguished by particular element values or the presence of certain extensions
  - Attribute certificates
    - Defined in RFC 5755
    - Use a different certificate format to link a user’s identity to a set of attributes that are typically used for authorization and access control
    - A user may have a number of different attribute certificates, with different set of attributes for different purposes
    - Defined in an “Attributes” extension

X.509 Formats



(a) X.509 Certificate



(b) Certificate Revocation List

Nowadays, most browsers reject certificates using SHA-1 or MD5 due to collisions

Applications for Public-Key Cryptosystem

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Access Control

- The Internet Security Glossary RFC 4949 defines access control as:

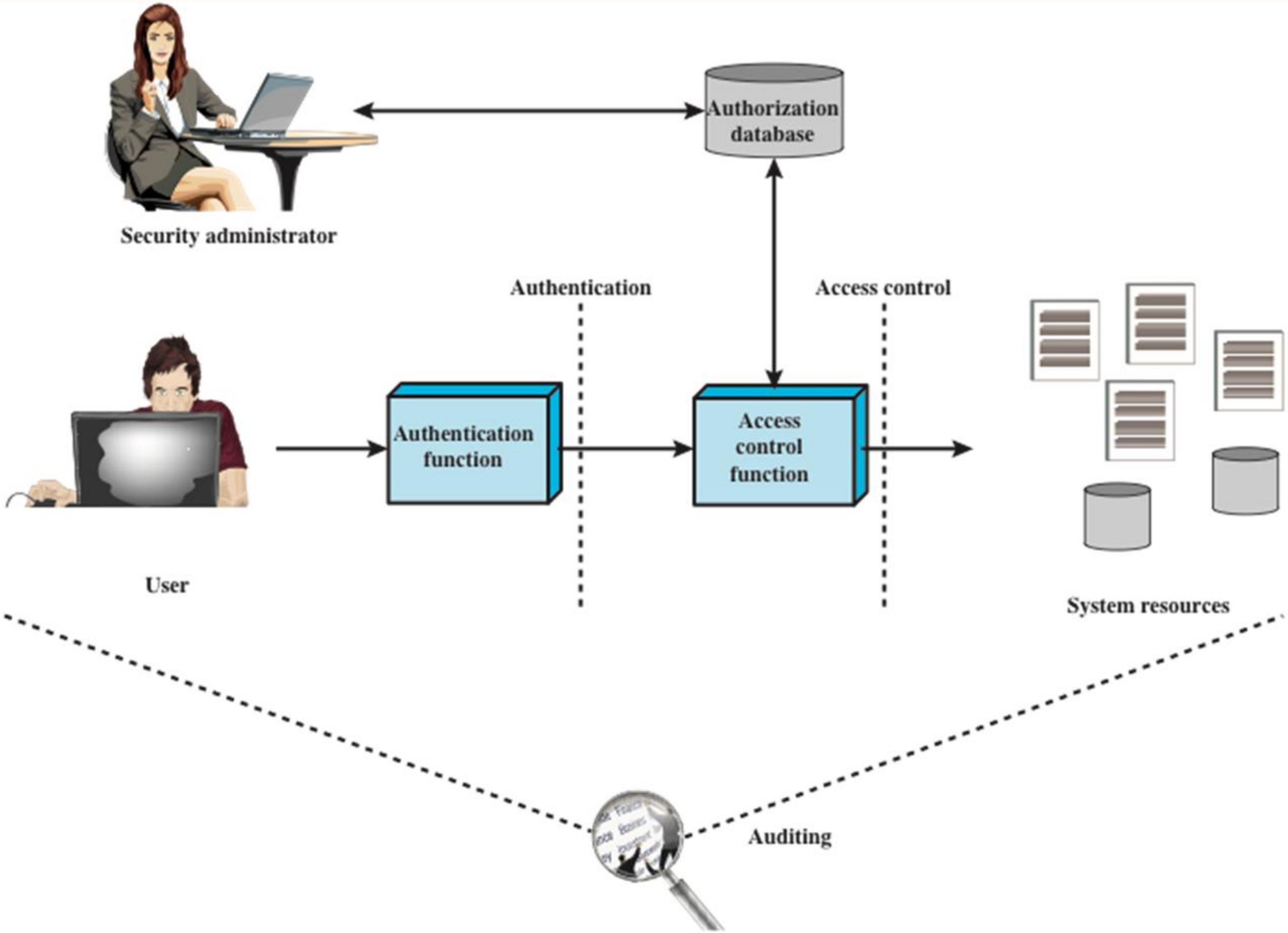
> “a process by which use of computer assets and system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy”

Access Control Principles

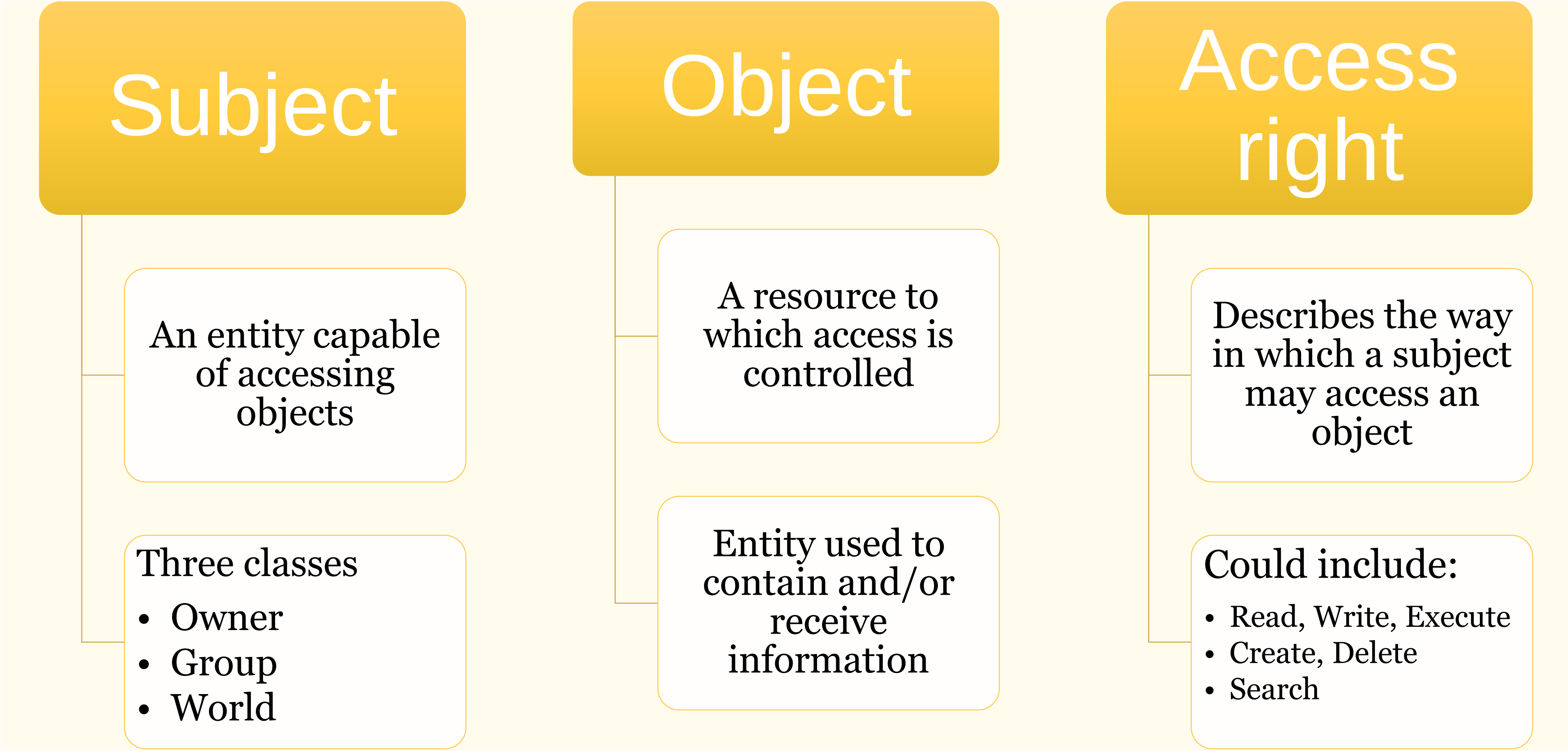
- In a broad sense, all of computer security is concerned with access control
- RFC 4949 defines computer security as:

“measures that implement and assure security services in a computer system, particularly those that assure access control service”

Broader context of access control



Basic Elements of Access Control



Subject

An entity capable of accessing objects

Three classes

- Owner
- Group
- World

Object

A resource to which access is controlled

Entity used to contain and/or receive information

Access right

Describes the way in which a subject may access an object

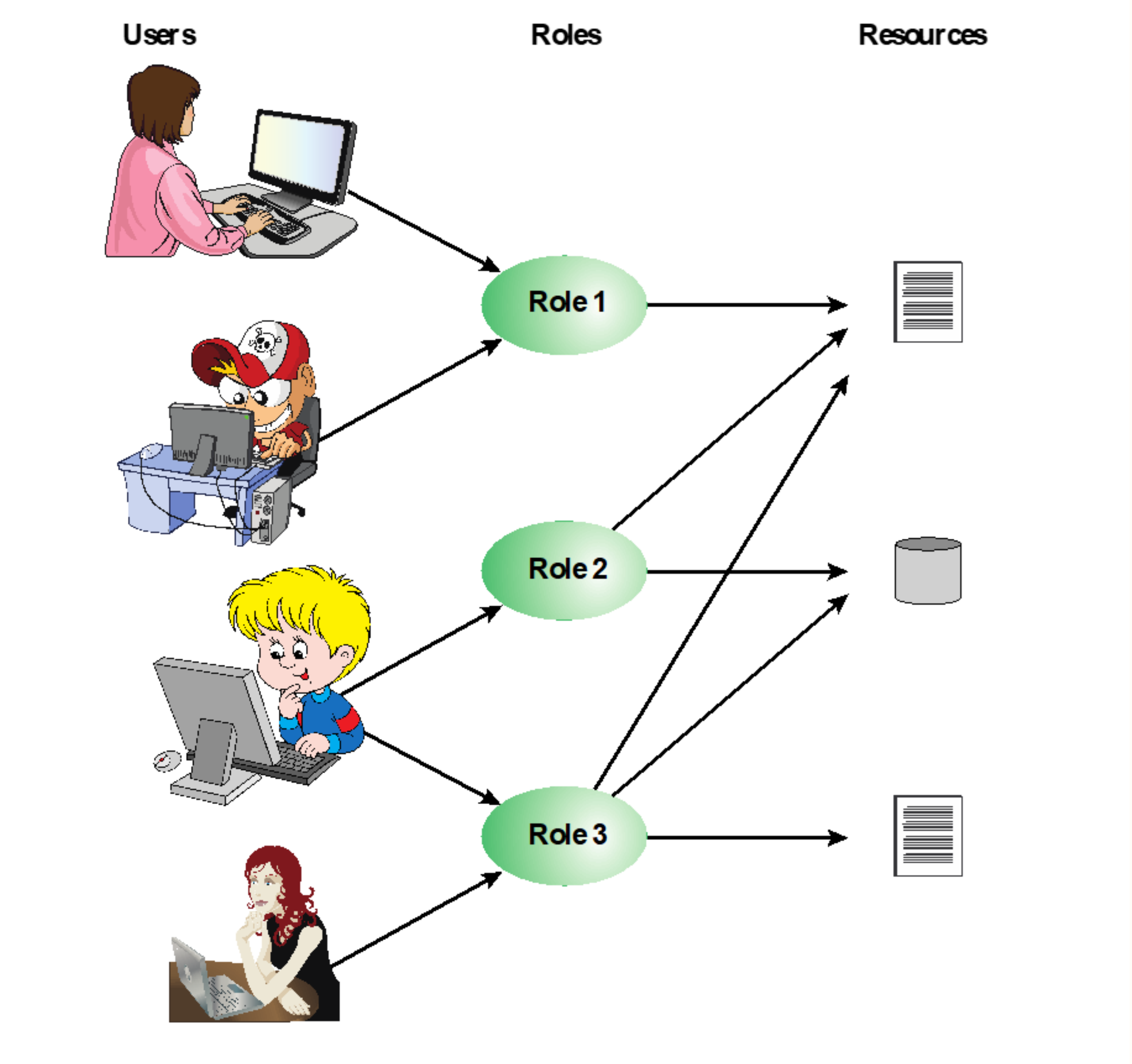
Could include:

- Read, Write, Execute
- Create, Delete
- Search

Role Based Access Control

- Real computer systems are used by organizations with defined job descriptions
- We can use the job descriptions to define roles within the computer system
- We can then define the security policy around what a role needs to do to deliver on the job

Users, Roles, and Resources



Web Security Considerations

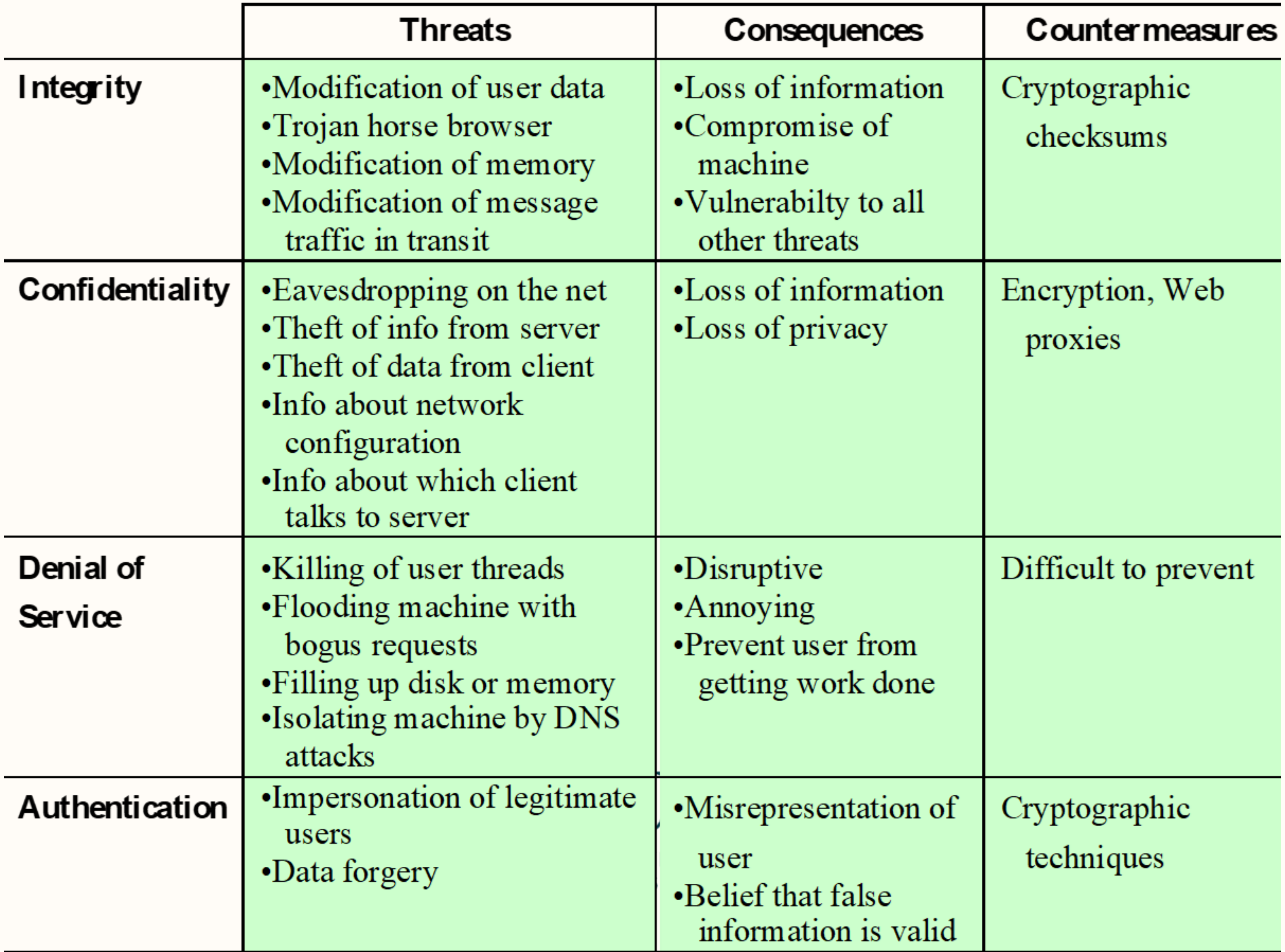
- The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets
- The following characteristics of Web usage suggest the need for tailored security tools:
  - Web servers are relatively easy to configure and manage
  - Web content is increasingly easy to develop
  - The underlying software is extraordinarily complex
    - May hide many potential security flaws

UNIVERSITY OF SUSSEX

Web Security Considerations

- A Web server can be exploited as a launching pad into the corporation’s or agency’s entire computer complex
- Casual and untrained (in security matters) users are common clients for Web-based services
  - Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures

A comparison of Threats on the Web



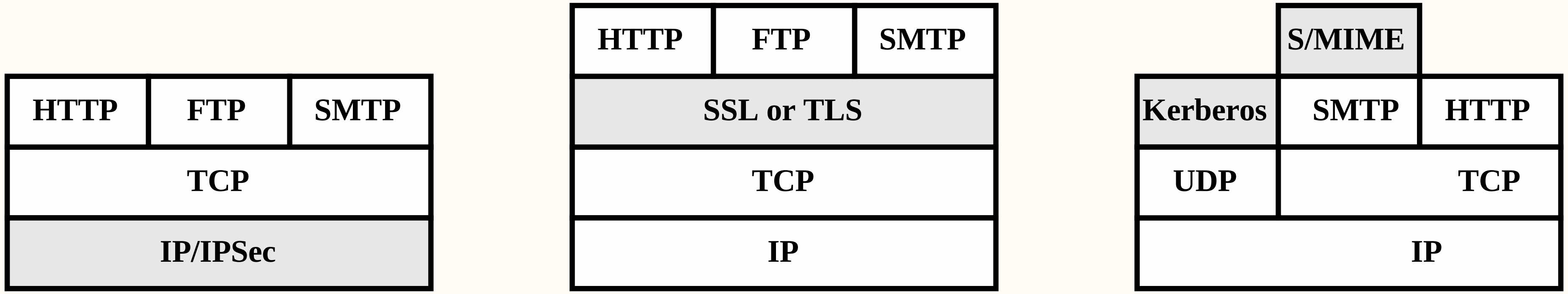
	Threats	Consequences	Countermeasures
Integrity	•Modification of user data •Trojan horse browser •Modification of memory •Modification of message traffic in transit	•Loss of information •Compromise of machine •Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	•Eavesdropping on the net •Theft of info from server •Theft of data from client •Info about network configuration •Info about which client talks to server	•Loss of information •Loss of privacy	Encryption, Web proxies
Denial of Service	•Killing of user threads •Flooding machine with bogus requests •Filling up disk or memory •Isolating machine by DNS attacks	•Disruptive •Annoying •Prevent user from getting work done	Difficult to prevent
Authentication	•Impersonation of legitimate users •Data forgery	•Misrepresentation of user •Belief that false information is valid	Cryptographic techniques

Relative Location of Security Facilities in the TCP/IP Protocol Stack

IPsec is transparent to end users and applications

virtually all browsers come equipped with TLS

security services are embedded within the particular application

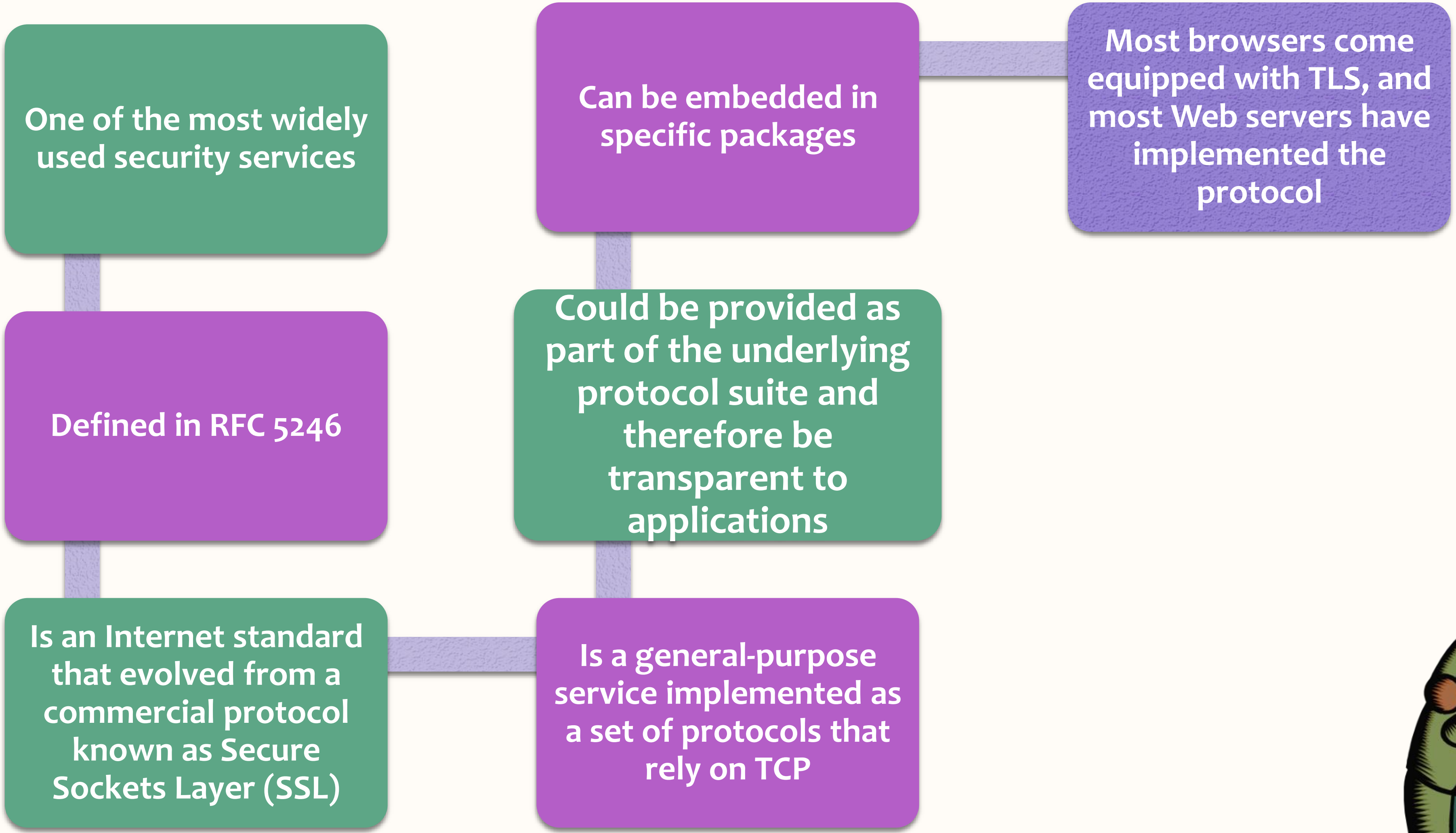


(a) Network Level

(b) Transport Level

(c) Application Level

Transport Layer Security (TLS)



One of the most widely used security services

Defined in RFC 5246

Is an Internet standard that evolved from a commercial protocol known as Secure Sockets Layer (SSL)

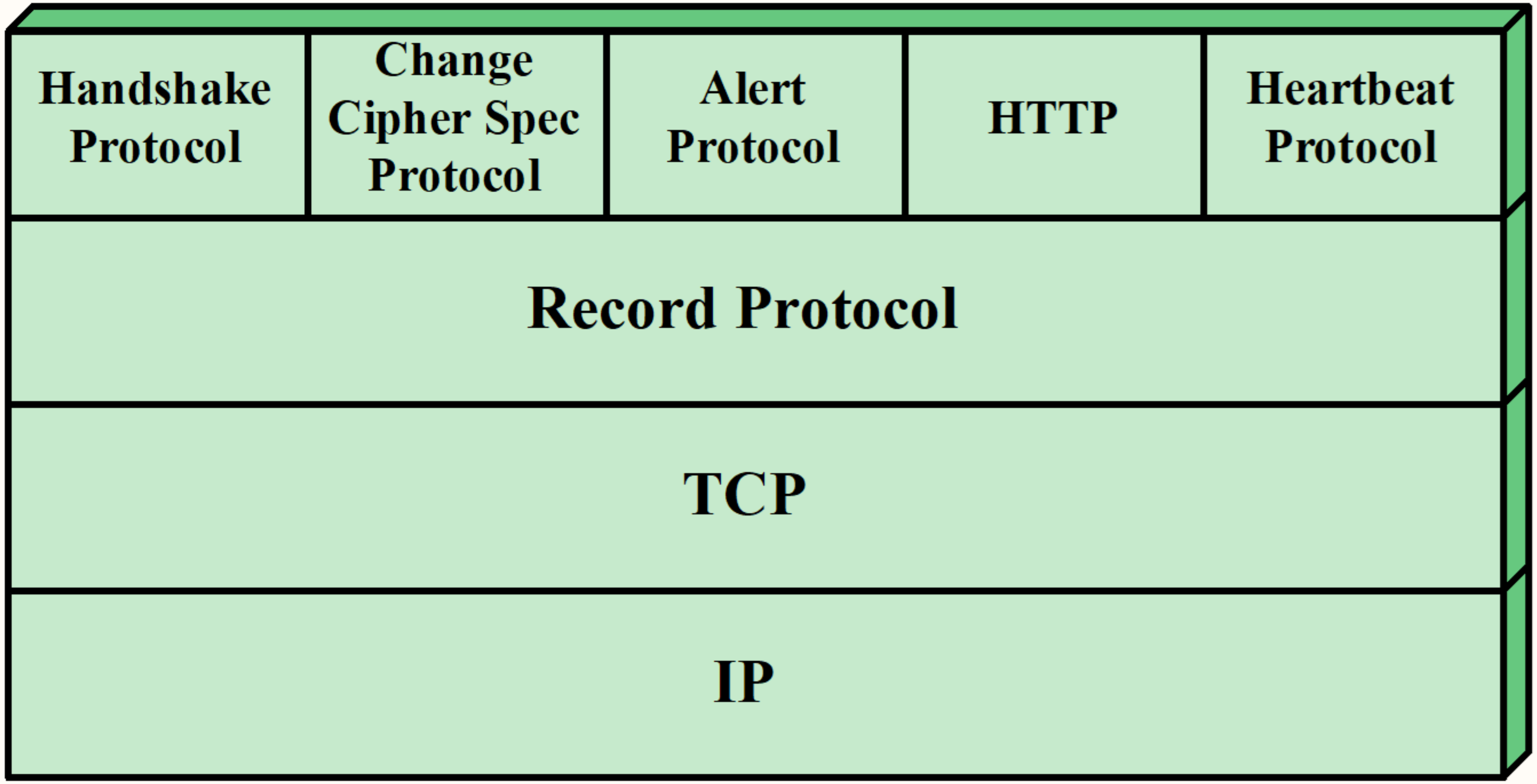
Is a general-purpose service implemented as a set of protocols that rely on TCP

Could be provided as part of the underlying protocol suite and therefore be transparent to applications

Can be embedded in specific packages

Most browsers come equipped with TLS, and most Web servers have implemented the protocol

SSL/TLS Protocol Stack



Handshake Protocol

Change Cipher Spec Protocol

Alert Protocol

HTTP

Heartbeat Protocol

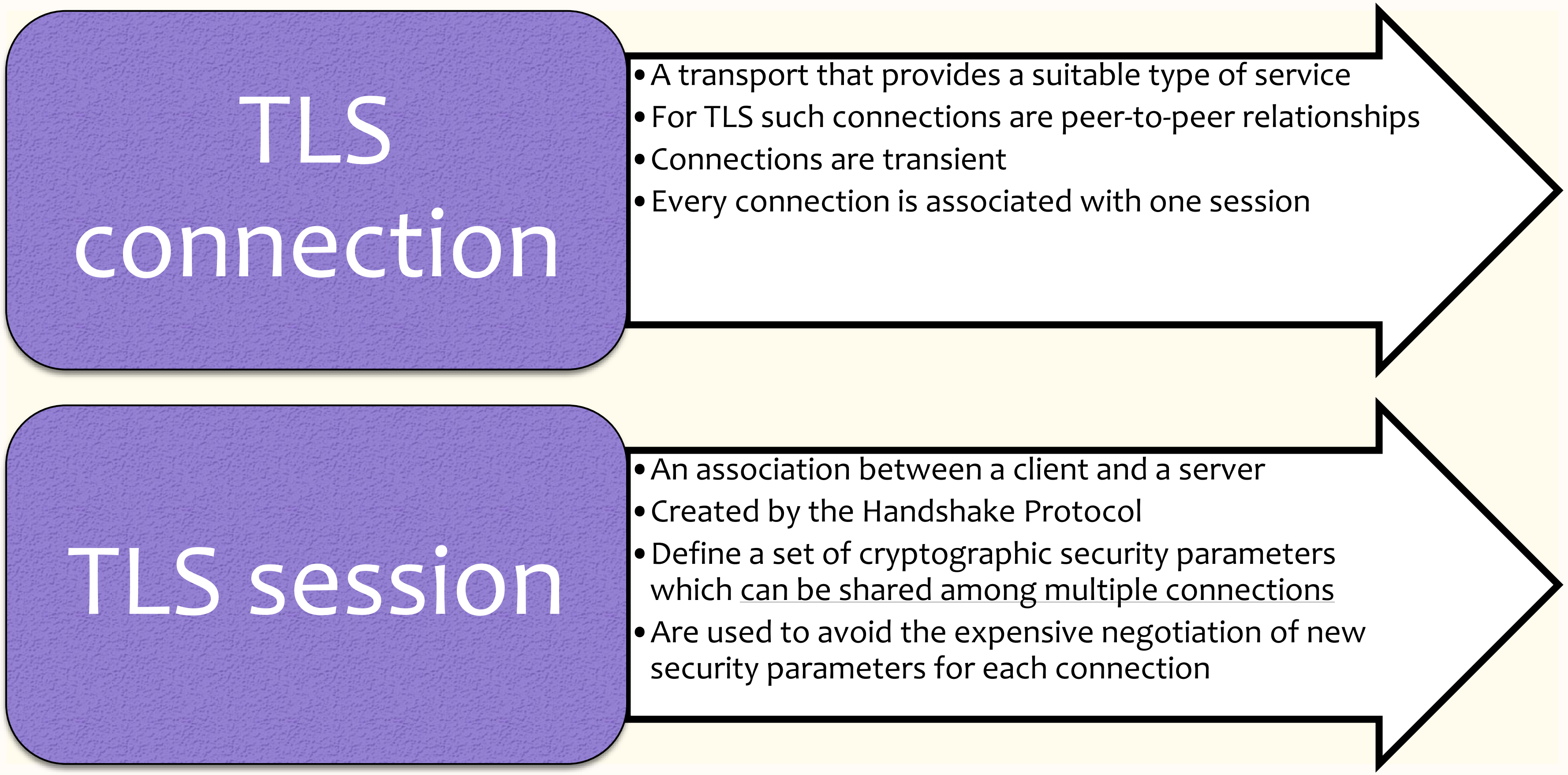
Record Protocol

TCP

IP

TLS Architecture

- Two important TLS concepts are:



TLS connection

- A transport that provides a suitable type of service
- For TLS such connections are peer-to-peer relationships
- Connections are transient
- Every connection is associated with one session

TLS session

- An association between a client and a server
- Created by the Handshake Protocol
- Define a set of cryptographic security parameters which can be shared among multiple connections
- Are used to avoid the expensive negotiation of new security parameters for each connection

Session State Parameters

Session identifier

An arbitrary byte sequence chosen by the server to identify an active or resumable session state

Peer certificate

An X509.v3 certificate of the peer; this element of the state may be null

Compression method

The algorithm used to compress data prior to encryption

Cipher spec

Specifies the bulk data encryption algorithm and a hash algorithm used for MAC calculation; also defines cryptographic attributes such as the hash_size

Master secret

48-byte secret shared between the client and the server

Is resumable

A flag indicating whether the session can be used to initiate new connections

Connection State Parameters

Server and client random

- Byte sequences that are chosen by the server and client for each connection

Server write MAC secret

- The secret key used in MAC operations on data sent by the server

Client write MAC secret

- The secret key used in MAC operations on data sent by the client

Server write key

- The secret encryption key for data encrypted by the server and decrypted by the client

Client write key

- The symmetric encryption key for data encrypted by the client and decrypted by the server

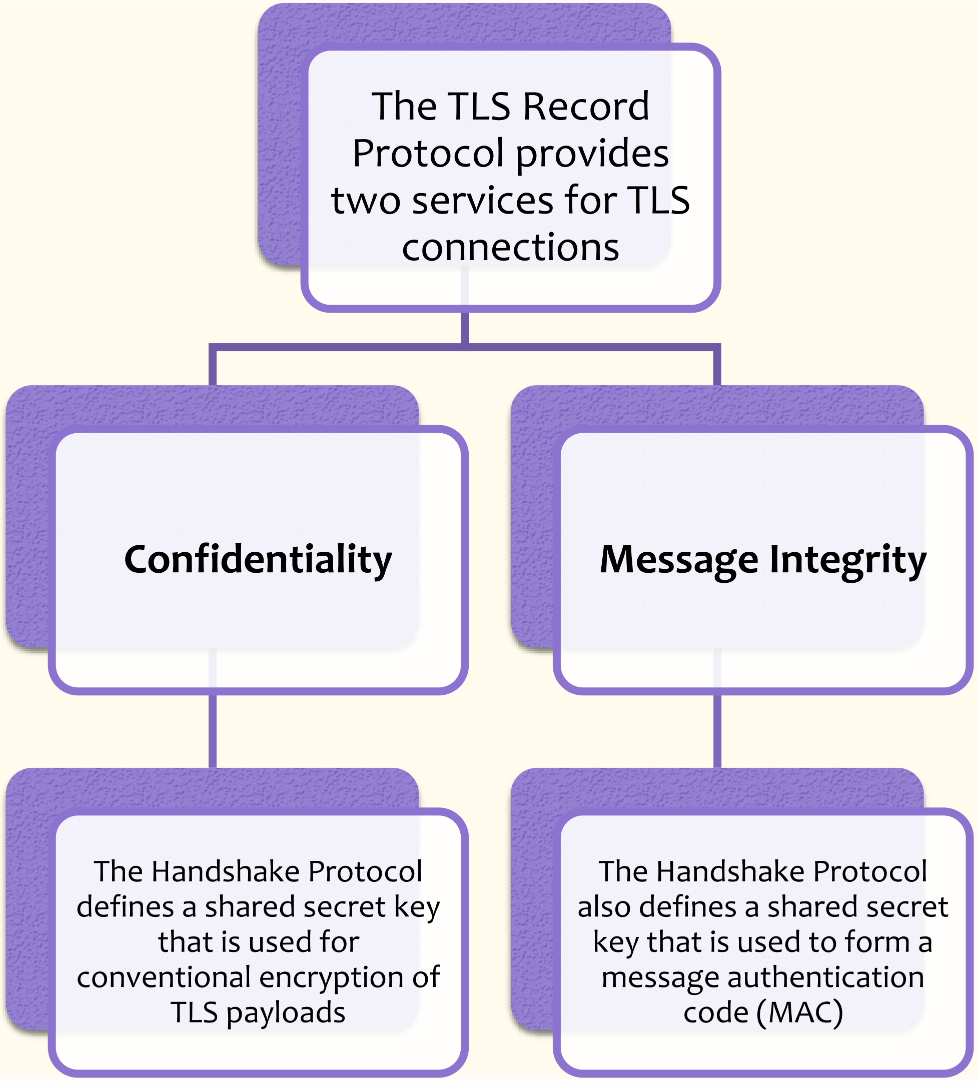
Initialization vectors

- When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key
- This field is first initialized by the TLS Handshake Protocol
- The final ciphertext block from each record is preserved for use as the IV with the following record

Sequence numbers

- Each party maintains separate sequence numbers for transmitted and received messages for each connection
- When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero
- Sequence numbers may not exceed 2^{64} - 1

TLS Record Protocol



The TLS Record Protocol provides two services for TLS connections

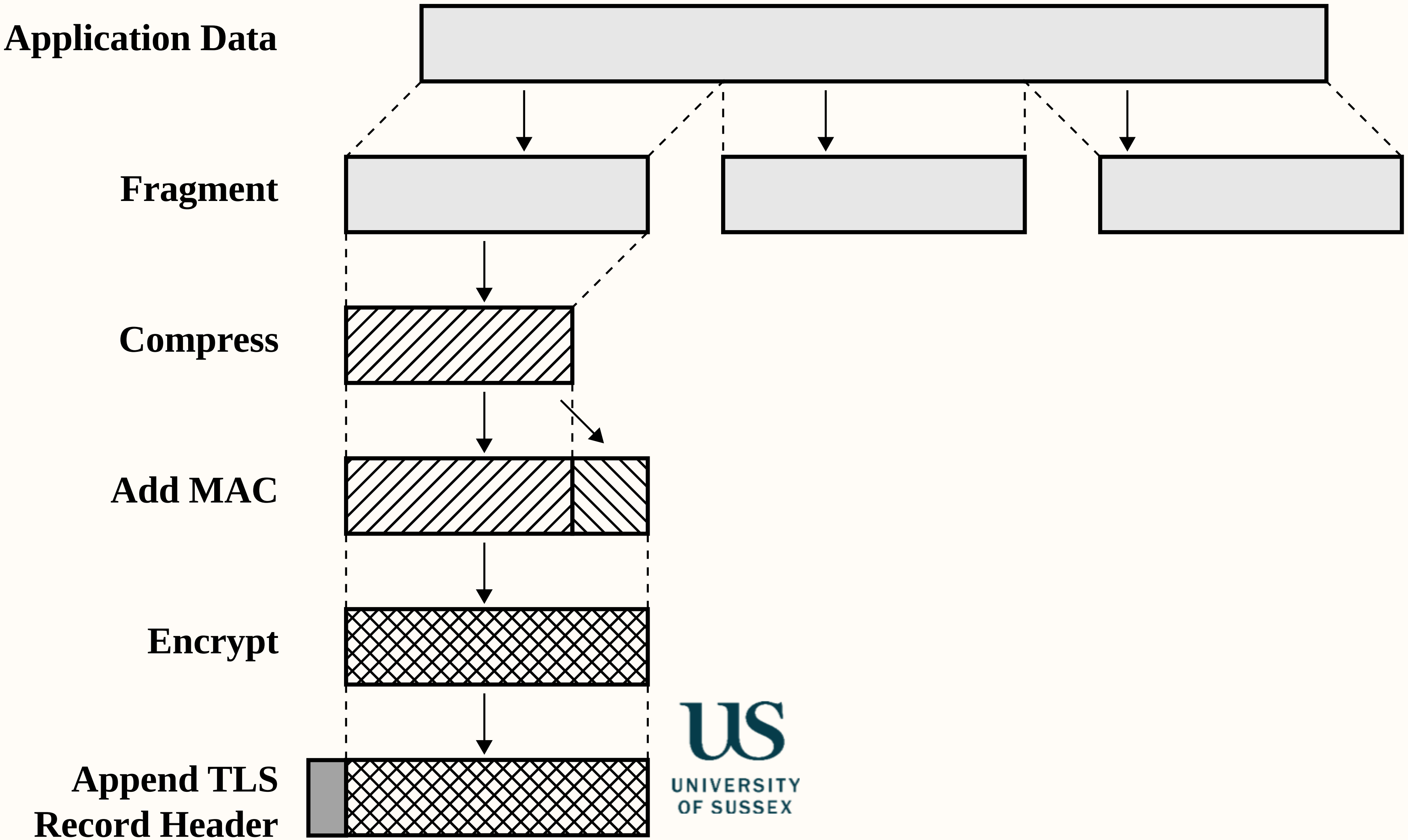
Confidentiality

The Handshake Protocol defines a shared secret key that is used for conventional encryption of TLS payloads

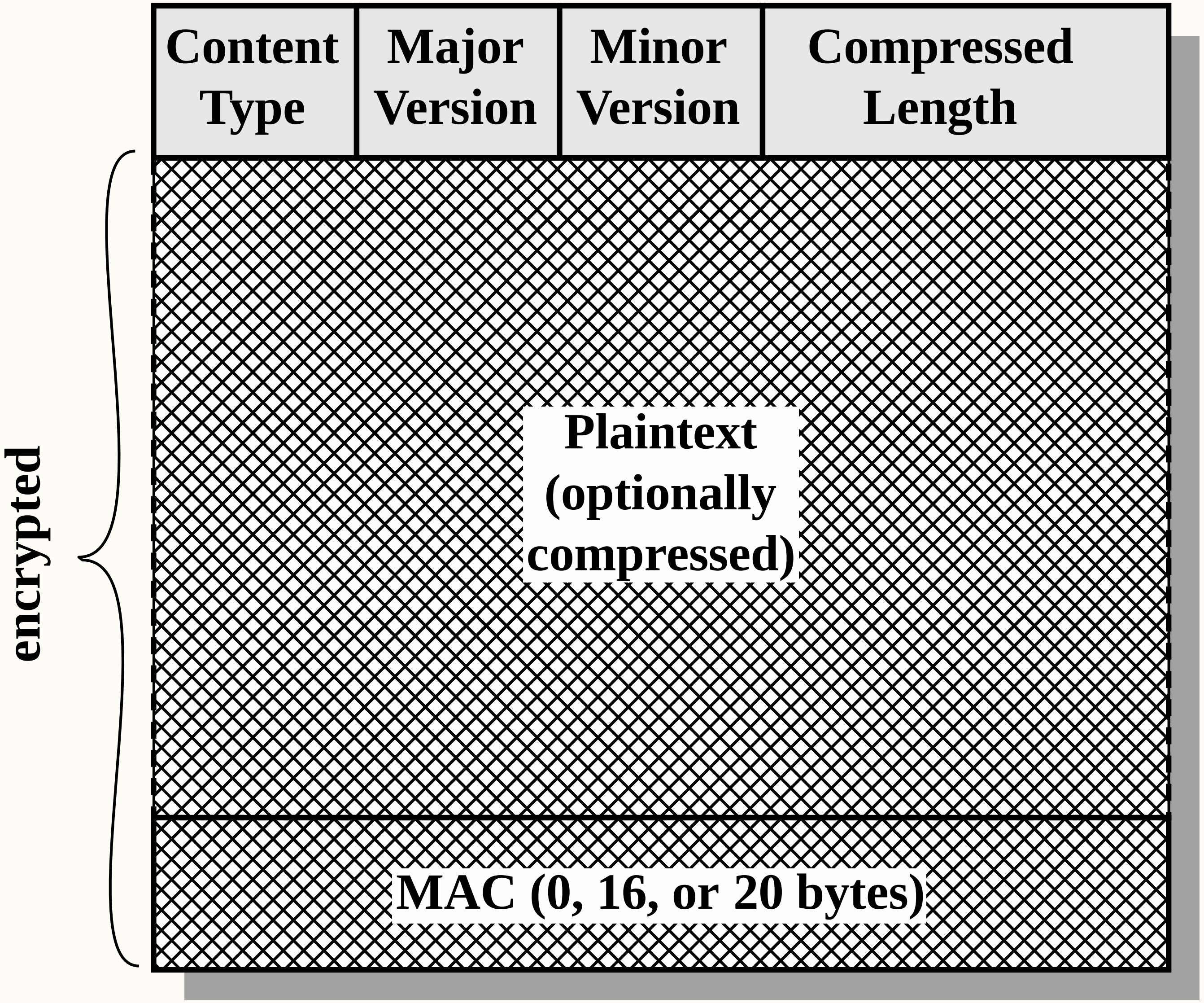
Message Integrity

The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC)

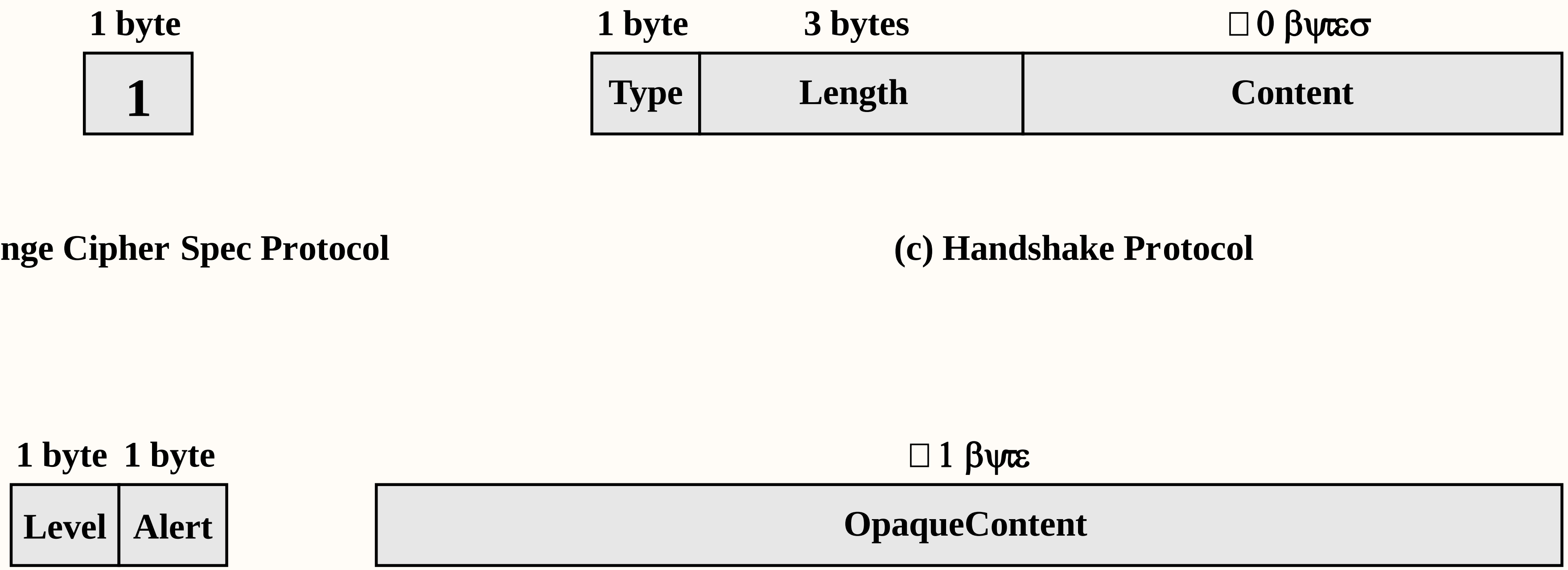
TLS Record Protocol Operation



TLS Record Format



TLS Record Protocol Payload



(a) Change Cipher Spec Protocol

(c) Handshake Protocol

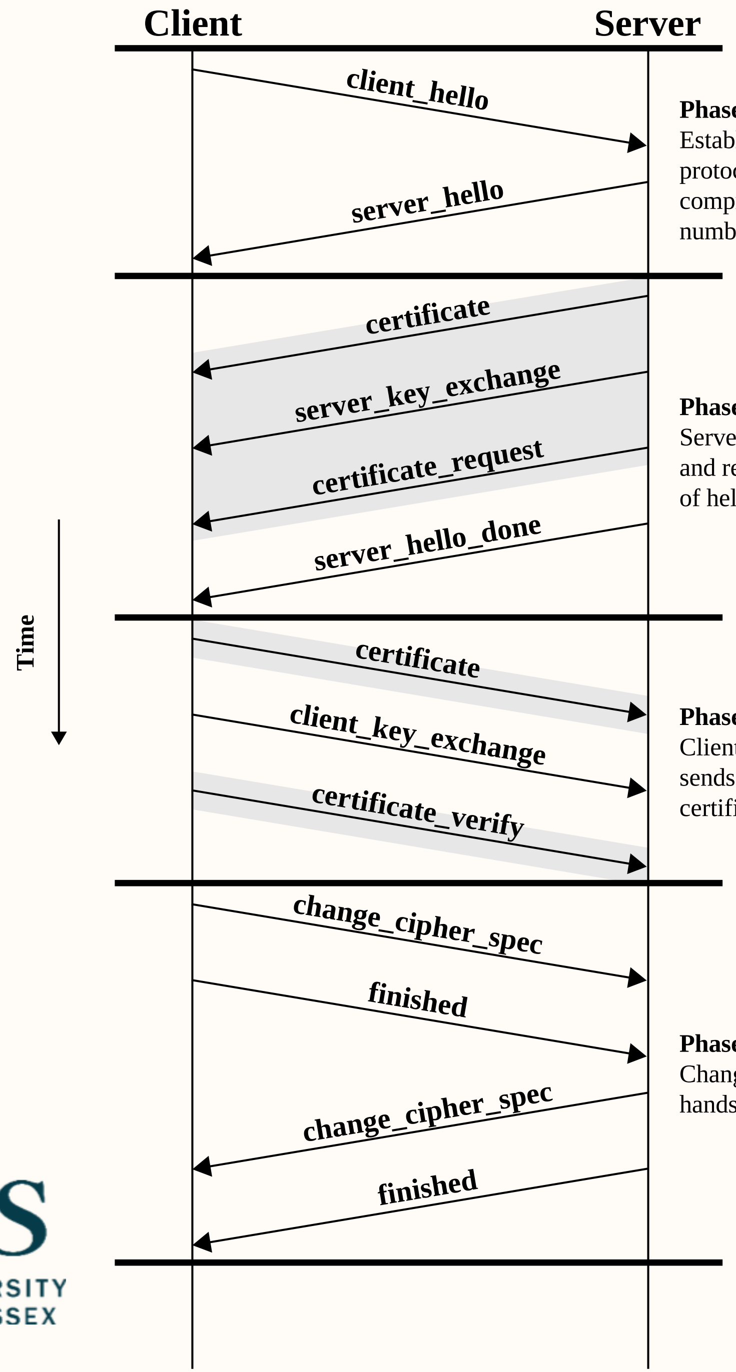
(b) Alert Protocol

(d) Other Upper-Layer Protocol (e.g., HTTP)

TLS Handshake Protocol Message Types

Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

Handshake Protocol Action



Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.

Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.

Phase 4
Change cipher suite and finish handshake protocol.

Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

UNIVERSITY OF SUSSEX

SSL/TLS Attacks

- The attacks can be grouped into four general categories:
  - Attacks on the handshake protocol
  - Attacks on the record and application data protocols
  - Attacks on the PKI
  - Other attacks
- The constant back-and-forth between threats and countermeasures determines the evolution of Internet-based protocols

TLSv1.3

- Primary aim is to improve the security of TLS
- Significant changes from version 1.2 are:
  - TLSv1.3 removes support for a number of options and functions. Deleted items include:
    - Compression
    - Ciphers that do not offer authenticated encryption
    - Static RSA and DH key exchange
    - 32-bit timestamp as part of the Random parameter in the client_hello message
    - Renegotiation
    - Change Cipher Spec Protocol
    - RC4
    - Use of MD5 and SHA-224 hashes with signatures

TLSv1.3

- TLSv1.3 uses Diffie-Hellman or Elliptic Curve Diffie-Hellman for key exchange and does not permit RSA
- TLSv1.3 allows for a “1 round trip time” handshake by changing the order of message sent with establishing a secure connection

UNIVERSITY
OF SUSSEX

Hyper Text Transfer Protocol Secure (HTTPS)

- The secure version of HTTP
- HTTPS encrypts all communications between the browser and the website
- Data sent using HTTPS provides three important areas of protection:
  - Encryption
  - Data integrity
  - Authentication

Connection Initiation

For HTTPS, the agent acting as the HTTP client also acts as the TLS client

The client initiates a connection to the server on the appropriate port and then sends the TLS ClientHello to begin the TLS handshake

When the TLS handshake has finished, the client may then initiate the first HTTP request

All HTTP data is to be sent as TLS application data

There are three levels of awareness of a connection in HTTPS:

At the HTTP level, an HTTP client requests a connection to an HTTP server by sending a connection request to the next lowest layer

- Typically, the next lowest layer is TCP, but is may also be TLS/SSL

At the level of TLS, a session is established between a TLS client and a TLS server

- This session can support one or more connections at any time

A TLS request to establish a connection begins with the establishment of a TCP connection between the TCP entity on the client side and the TCP entity on the server side

Connection Closure

- An HTTP client or server can indicate the closing of a connection by including the line `Connection: close` in an HTTP record
- The closure of an HTTPS connection requires that TLS close the connection with the peer TLS entity on the remote side, which will involve closing the underlying TCP connection

Connection Closure

- TLS implementations must initiate an exchange of closure alerts before closing a connection
  - A TLS implementation may, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an “incomplete close”
- An unannounced TCP closure could be evidence of some sort of attack so the HTTPS client should issue some sort of security warning when this occurs

Security in Django

- Django security features cover protection against well-known attacks such as
  - Cross site scripting (XSS)
  - Cross site request forgery (CSRF)
  - SQL injection
  - Clickjacking
- besides allowing
  - host header validation
  - Referrer policy
  - Cross-origin opener policy
  - Session security
  - Etc.

Cross site scripting (XSS) protection

- XSS attacks allow a user to inject client-side scripts into the browsers of other users, being achieved by
  - storing the malicious scripts in the database where it will be retrieved and displayed to other users
  - getting users to click a link which will cause the attacker’s JavaScript to be executed by the user’s browser
- XSS attacks can originate from any untrusted source of data, such as cookies or web services
  - whenever the data is not sufficiently sanitized before including in a page.

Cross site scripting (XSS) protection

- Django templates protects against the majority of XSS attacks
  - by escaping specific characters which are particularly dangerous to HTML
  - but it is not entirely foolproof
- For example, `<style class={{ var }}>...</style>`
  - If `var` is set to `'class1 onmouseover=javascript:func()'`
  - this can result in unauthorized JavaScript execution, depending on how the browser renders imperfect HTML
- Recommendations
  - be particularly careful when using `is_safe` with custom template tags
  - be very careful when storing HTML in the database

Cross site request forgery (CSRF) protection

- CSRF attacks allow a malicious user to execute actions using the credentials of another user without that user's knowledge or consent
- Django has built-in CSRF protection, if it has been enabled and used where appropriate
- CSRF protection works by checking for a secret in each POST request
  - This ensures that a malicious user cannot "replay" a form POST to the website and have another logged in user unwittingly submit that form
  - The malicious user would have to know the secret, which is user specific (using a cookie)

UNIVERSITY
OF SUSSEX

Cross site request forgery (CSRF) protection

- If combined with HTTPS, it checks that the HTTP referer header is set to a URL on the same origin (including subdomain and port).
- However, there are limitations
  - it is possible to disable the CSRF module globally or for particular views
  - if the site has subdomains that are outside of the developer's control
- Recommendation
  - Be very careful with marking views with the `csrf_exempt` decorator unless it is absolutely necessary

UNIVERSITY OF SUSSEX

SQL injection protection

- SQL injection is a type of attack where a malicious user can execute arbitrary SQL code on a database
  - This can result in records being deleted or data leakage
- Django’s queries are constructed using query parameterization, hence protecting against such attack
  - A query’s SQL code is defined separately from the query’s parameters
  - parameters are escaped by the underlying database driver
- Recommentation
  - exercise caution when using `extra()` and `RawSQL`, used express a complex WHERE clause

UNIVERSITY OF SUSSEX

Clickjacking protection

- Clickjacking is a type of attack where a malicious site wraps another site in a frame
  - This attack can result in an unsuspecting user being tricked into performing unintended actions on the target site
- Django’s clickjacking protection is in the form of the `X-Frame-Options middleware`
  - It can prevent a site from being rendered inside a frame, in a supported browser
  - It is possible to disable the protection on a per view basis or to configure the exact header value sent

HTTPS

- HTTPS deployment is always desirable for security reasons
  - to avoid possible for malicious network users to sniff authentication credentials or any other information transferred between client and server, and
  - in the case of active network attackers, to alter data that is sent in either direction
- Besides enabling HTTPS on the server, it may be necessary to
  - set `SECURE_PROXY_SSL_HEADER` and validate any warnings thoroughly.
    - Failure to do this can result in CSRF vulnerabilities, and failure to do it correctly can also be dangerous!
  - Set `SECURE_SSL_REDIRECT` to `True`, so that requests over HTTP are redirected to HTTPS.

HTTPS

- Besides enabling HTTPS on the server, it may be necessary to
  - Use ‘secure’ cookies.
    - If a browser connects initially via HTTP, which is the default for most browsers, it is possible for existing cookies to be leaked
    - Hence, `SESSION_COOKIE_SECURE` and `CSRF_COOKIE_SECURE` settings should be set to `True`
  - Use HTTP Strict Transport Security (HSTS)
    - It is an HTTP header that informs a browser that all future connections to a particular site should always use HTTPS
    - It may either be configured with `SECURE_HSTS_SECONDS`, `SECURE_HSTS_INCLUDE_SUBDOMAINS`, and `SECURE_HSTS_PRELOAD`, or on the web server.

Additional security topics

- Nonetheless Django security features
  - it is importante to properly deploy the web application and take advantage of the security protection of the web server, operating system and other components
- Specifically
  - Make sure that the Python code is outside of the web server’s root.
    - This will ensure that the Python code is not accidentally served as plain text (or accidentally executed).
  - Take care with any user uploaded files
  - Django does not throttle requests to authenticate users
    - To protect against brute-force attacks against the authentication system, one may consider deploying a Django plugin or web server module to throttle these requests
  - Keep your `SECRET_KEY`, and `SECRET_KEY_FALLBACKS` if in use, secret.

Additional security topics

- Specifically
  - It is a good idea to limit the accessibility of the caching system and database using a firewall
  - Take a look at the Open Web Application Security Project (OWASP) Top 10 list which identifies some common vulnerabilities in web applications
    - While Django has tools to address some of the issues, other issues must be accounted for in the design of the project
  - Mozilla discusses various topics regarding web security
    - Their pages also include security principles that apply to any system.

Next Lecture ...

- ✓ Introduction
- ✓ HTTP, Caching, and CDNs
- ✓ Views
- ✓ Templates
- ✓ Forms
- ✓ Models
- ✓ Security

- ➢ **Transactions**
- RPC/RMI/Thrift
- Web Services
- RESTful Services
- Time
- Elections/Group Communication
- Zookeeper

UNIVERSITY
OF SUSSEX