Information Security
CS3002
(Sections BDS-7A/B)
Lecture 23

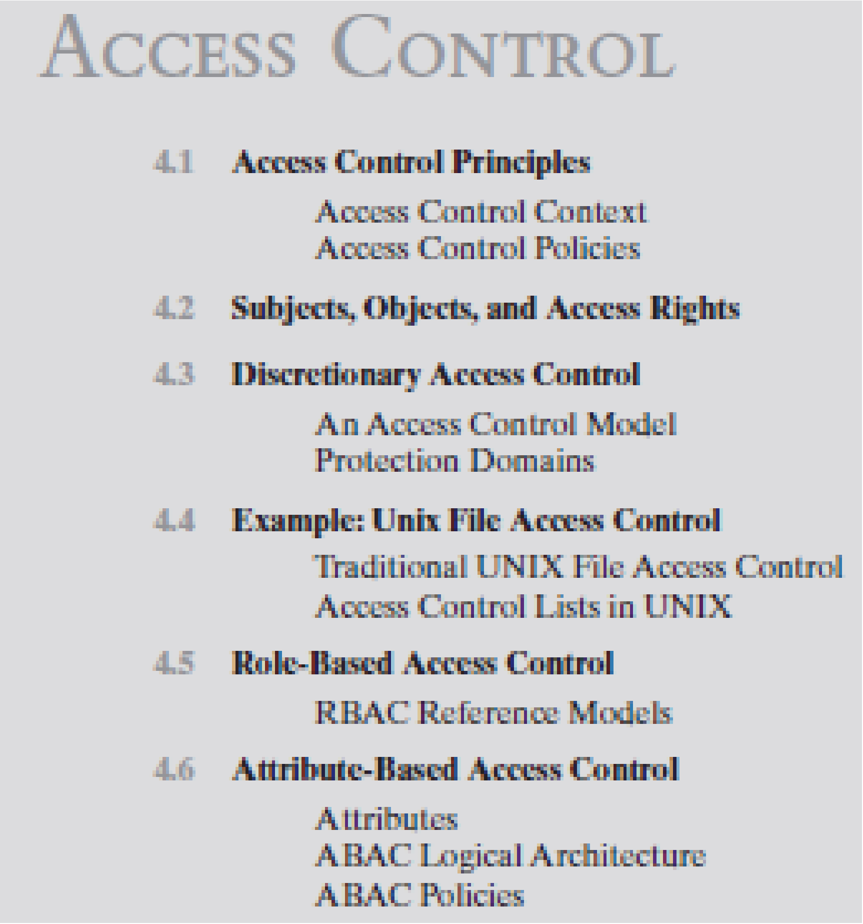
Instructor: Dr. Syed Mohammad Irteza

Assistant Professor, Department of Computer Science

11 November, 2024

Previous Lecture

- Access Control
  - Maps to some parts of Chapter 4 in Computer Security: Principles and Practices (William Stallings)



ACCESS CONTROL

4.1 **Access Control Principles**
Access Control Context
Access Control Policies

4.2 **Subjects, Objects, and Access Rights**

4.3 **Discretionary Access Control**
An Access Control Model
Protection Domains

4.4 **Example: Unix File Access Control**
Traditional UNIX File Access Control
Access Control Lists in UNIX

4.5 **Role-Based Access Control**
RBAC Reference Models

4.6 **Attribute-Based Access Control**
Attributes
ABAC Logical Architecture
ABAC Policies

Second Lecture After Mid-02 Exam

Remaining Lectures (Content)

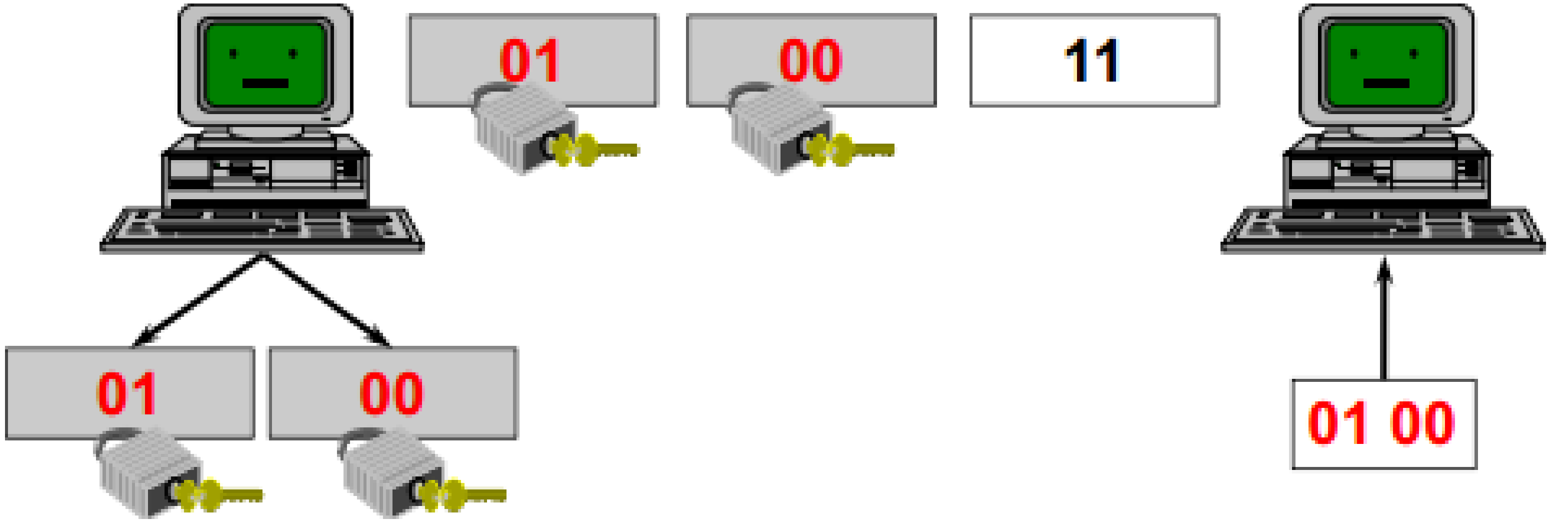
- Network Security (4 lectures)
- Theoretical Models of Access Control (1 lecture)
- Cybercrime Laws and Ethics (1 lecture)
- Project Presentations (2 lectures at least)

Network Security – I

- SSL – Introduction
- SSL certificate
- SSL architecture
- SSL handshake

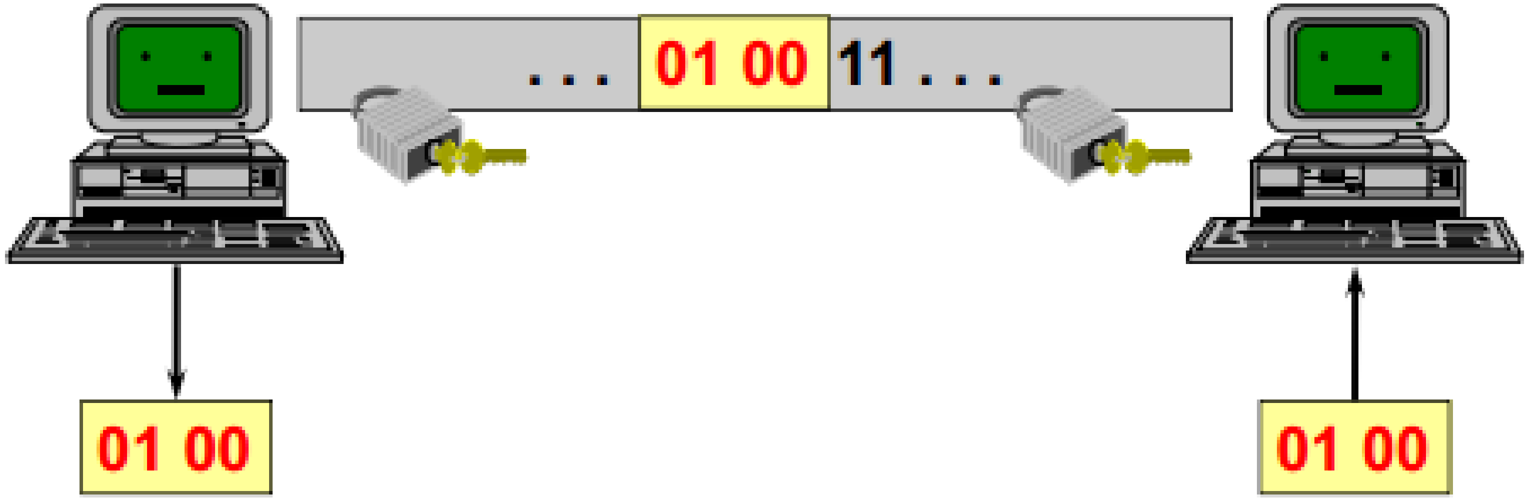
Message/Data Security

- Authentication (single), integrity and privacy self contained in the message
- Possibility of non repudiation
- Requires modification of applications



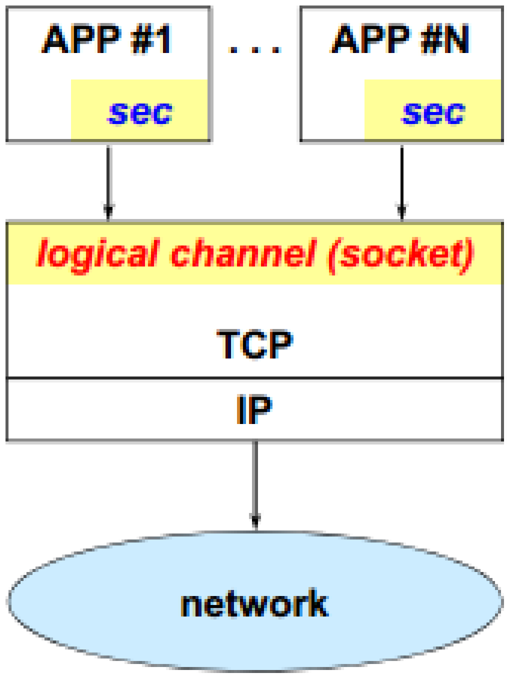
Channel Security

- Authentication (single or mutual), integrity and privacy only during the transit inside the communication channel
- No possibility of non repudiation
- Requires no (or small) modification of applications



Security internal to applications

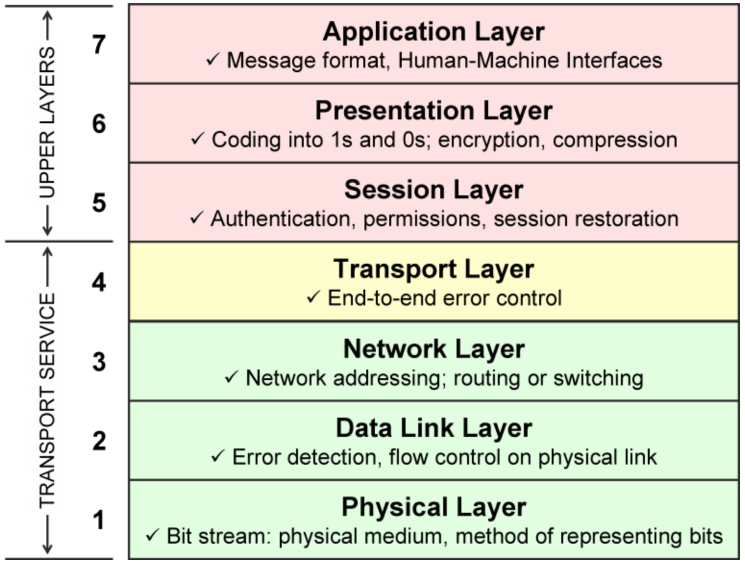
- Each application implements security *internally*
- The common part is limited to the communication channels (*socket*)
- Possible implementation errors (inventing security protocols is *not simple*!)
- Does not guarantee *interoperability*



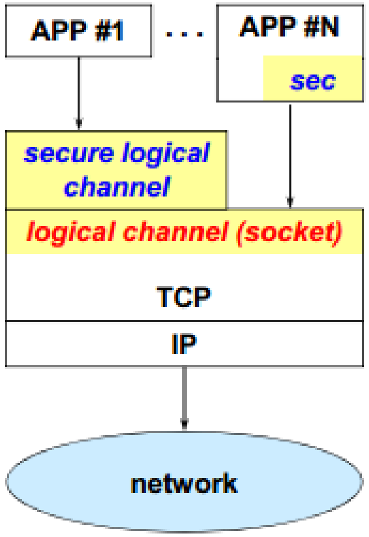
Security external to applications

- The *session* level would be the ideal one to be used to implement many security functions
- … but it does not exist in *TCP/IP*!
- a “secure session” level was proposed:
  - it simplifies the work of application developers
  - it avoids implementation errors
  - it is up to the application to select it (or not)

OSI Model (not the TCP/IP Model)



UPPER LAYERS	7	**Application Layer** ✓ Message format, Human-Machine Interfaces
	6	**Presentation Layer** ✓ Coding into 1s and 0s; encryption, compression
	5	**Session Layer** ✓ Authentication, permissions, session restoration
TRANSPORT SERVICE	4	**Transport Layer** ✓ End-to-end error control
	3	**Network Layer** ✓ Network addressing; routing or switching
	2	**Data Link Layer** ✓ Error detection, flow control on physical link
	1	**Physical Layer** ✓ Bit stream: physical medium, method of representing bits

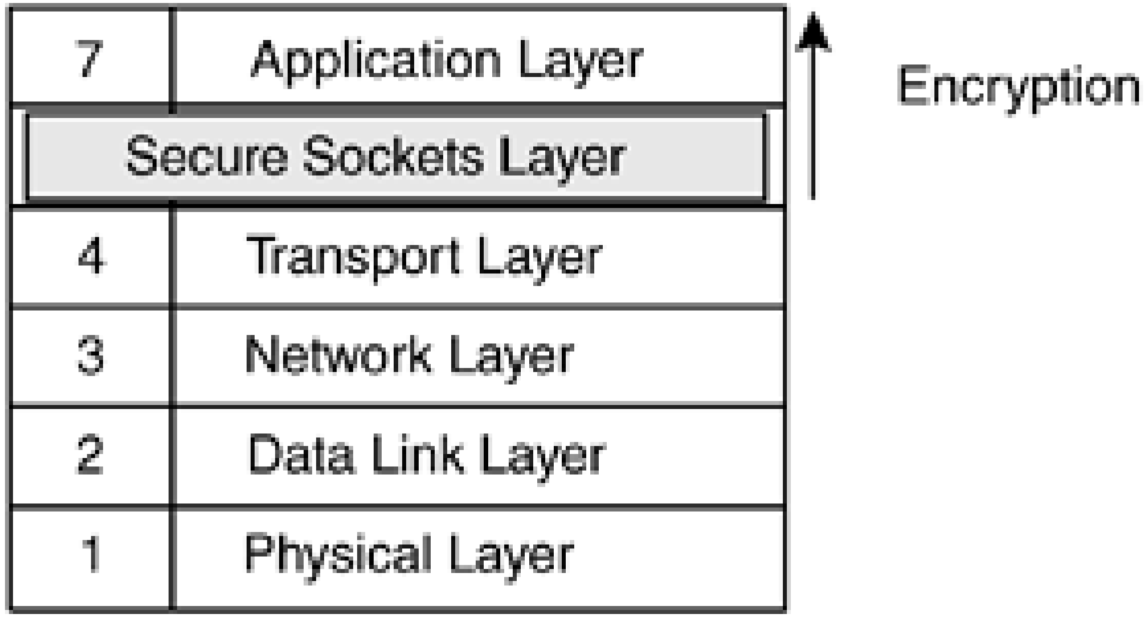


SSL: What is it?

- Security at layer 4 (transport layer)
- **S**ecure **S**ockets **L**ayer (SSL)
- Secure transport channel (session level):
  - Peer authentication (server, server + client)
  - Message confidentiality
  - Message authentication and integrity
  - Protection against replay attacks
- Easily applicable to all protocols based on TCP:
  - HTTP, SMTP, FTP, TELNET, ...
  - e.g. the famous secure HTTP (https://....) = 443/TCP

SSL/TLS

- Philosophy of SSL: Easier to deploy something if no changes in OS required
- Application’s API (Socket) is interface to SSL: Hence secure socket layer
- API to SSL is the superset of API to TCP
- SSL/TLS operate above TCP. OS doesn’t change, applications do!

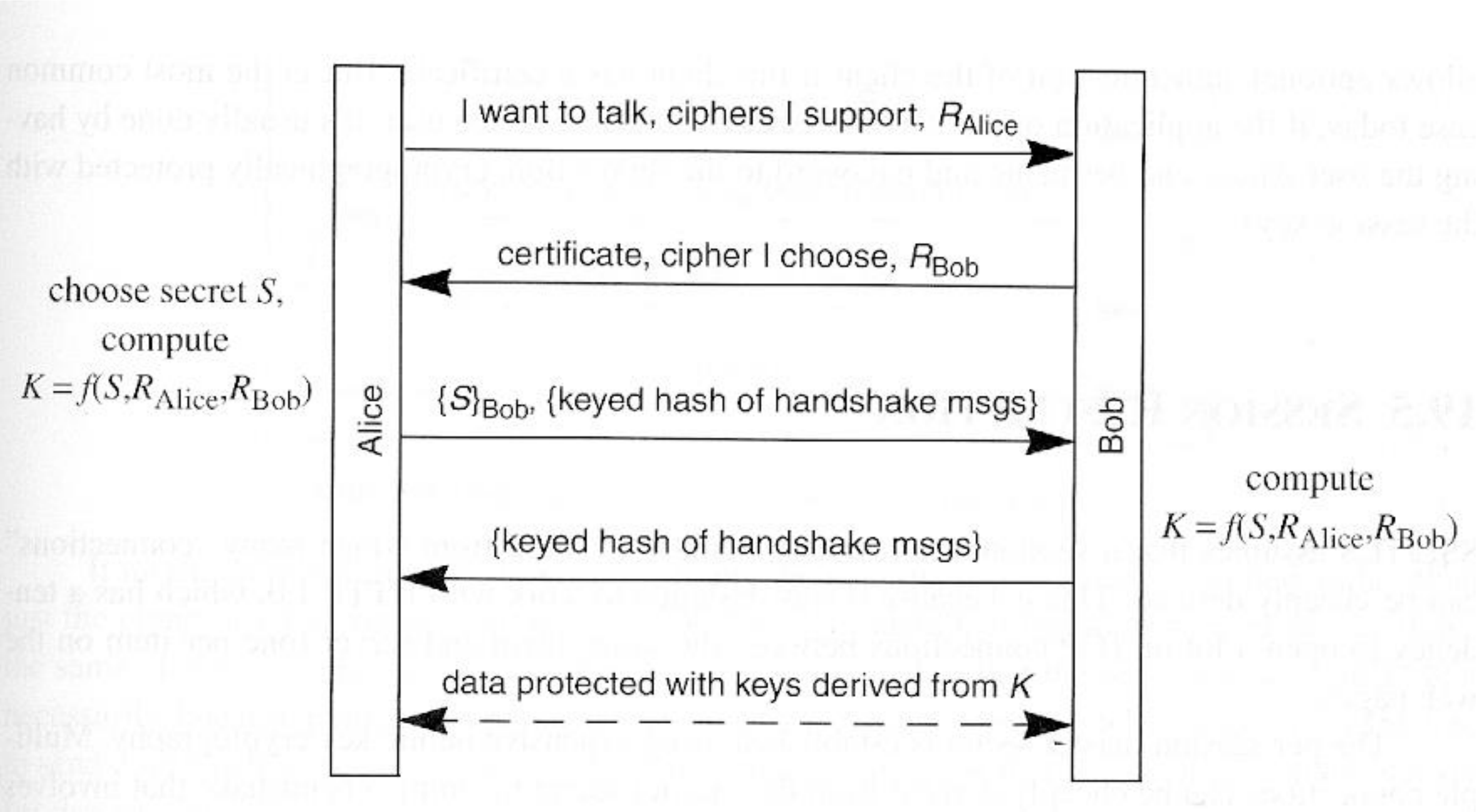


7	Application Layer
Secure Sockets Layer	
4	Transport Layer
3	Network Layer
2	Data Link Layer
1	Physical Layer

SSL Handshake

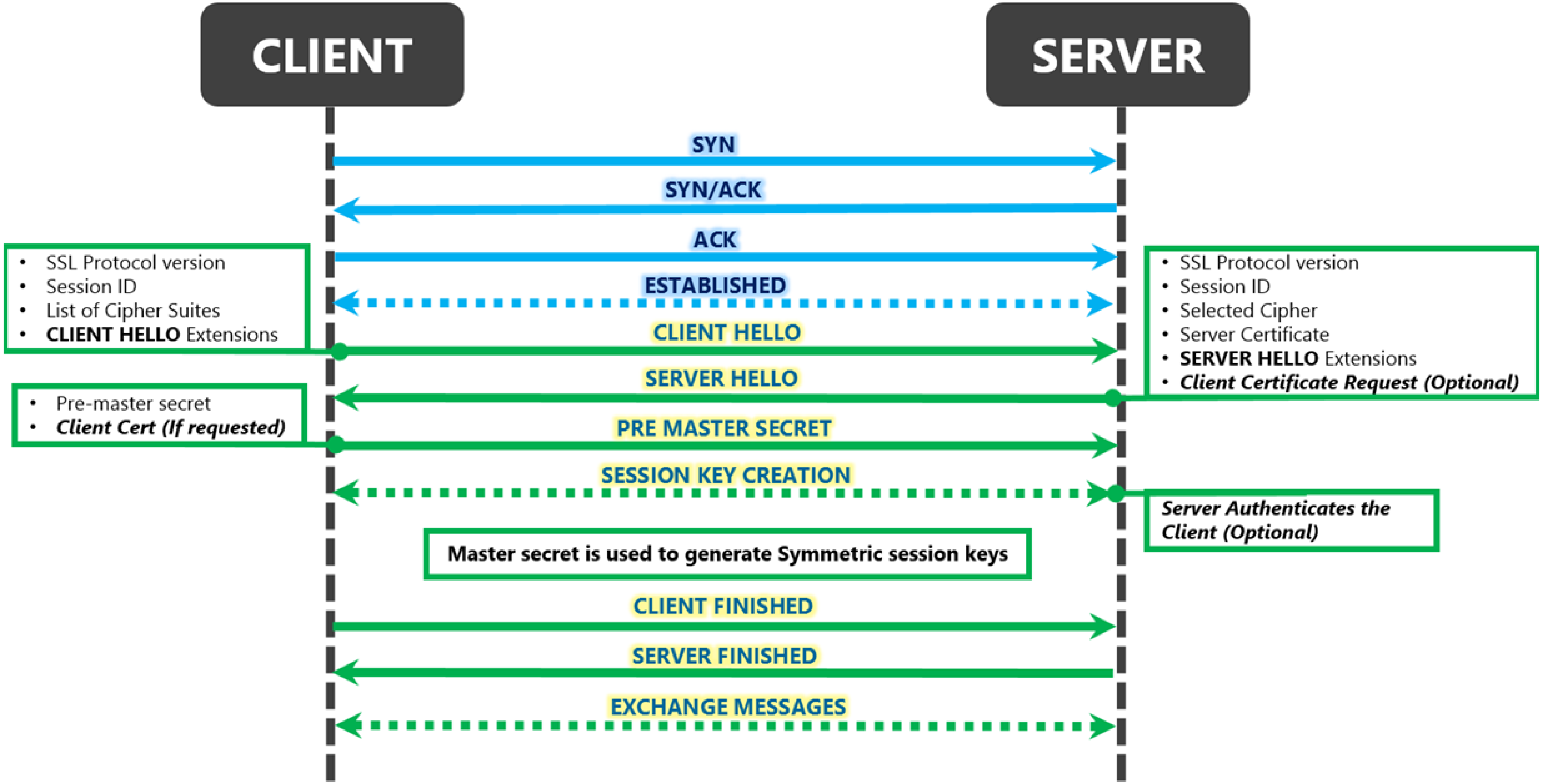
- Agree on a set of algorithms for confidentiality, integrity and authentication
- Exchange random numbers between the client and the server to be used for the subsequent generation of the keys
- Establish a symmetric key by means of public key operations, e.g. RSA
- Negotiate the session-ID
- Exchange the necessary certificates

SSL Handshake: Simplified


I want to talk, ciphers I support, R_{Alice}
certificate, cipher I choose, R_{Bob}
choose secret S, compute $K = f(S, R_{Alice}, R_{Bob})$
Alice
$\{S\}_{Bob}$, {keyed hash of handshake msgs}
Bob
compute $K = f(S, R_{Alice}, R_{Bob})$
{keyed hash of handshake msgs}
data protected with keys derived from K

- Secrets are:
  - Pre-master key S
  - Master Key K
- Server authentication
- Client authentication by password (optional)

SSL Handshake: In Detail



- SSL Protocol version
- Session ID
- List of Cipher Suites
- **CLIENT HELLO** Extensions

- SSL Protocol version
- Session ID
- Selected Cipher
- Server Certificate
- **SERVER HELLO** Extensions
- ***Client Certificate Request (Optional)***

- Pre-master secret
- ***Client Cert (If requested)***

Server Authenticates the Client (Optional)

Master secret is used to generate Symmetric session keys

SSL Handshake: Figure of CS: P&P (William Stallings)

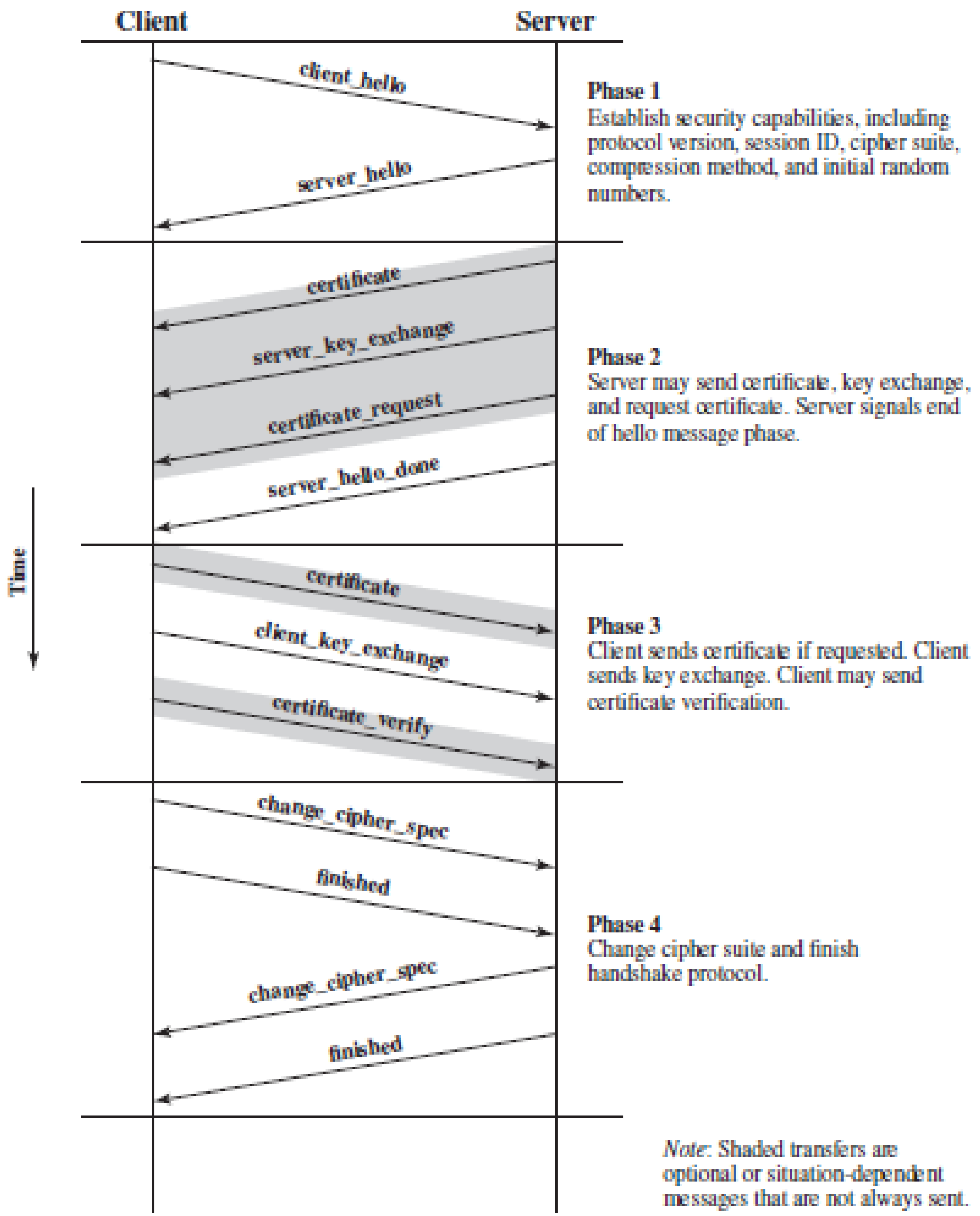
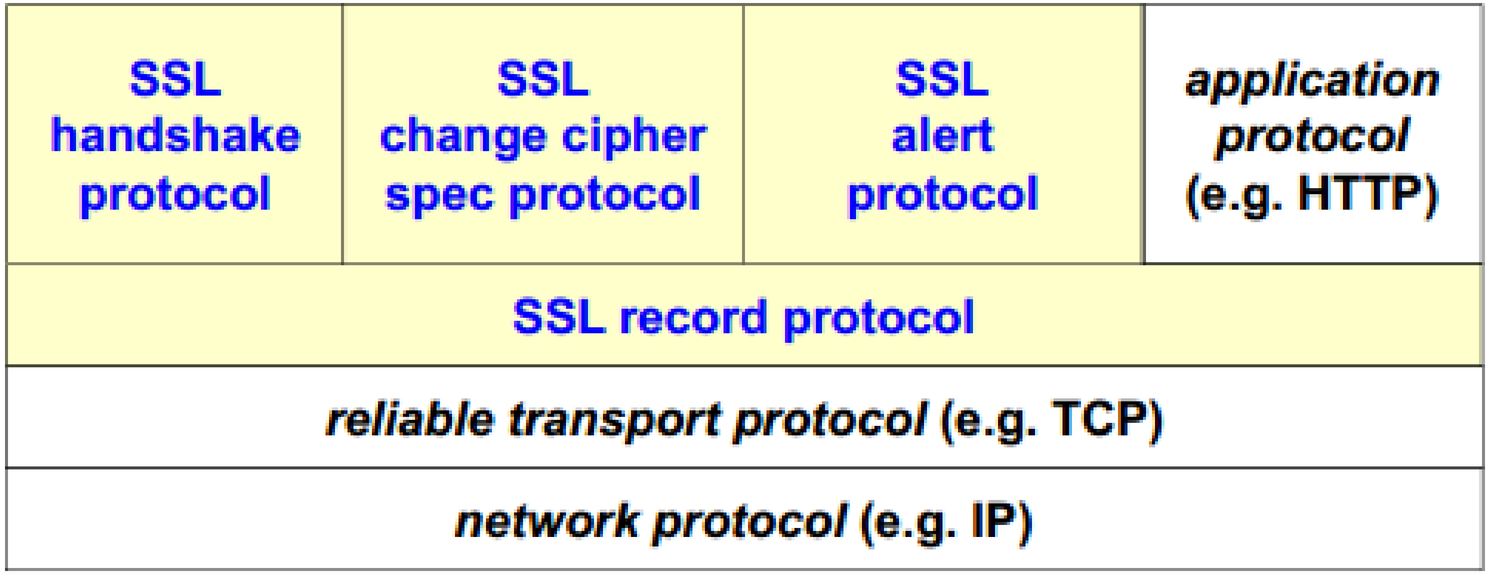

Client
Server
client_hello
server_hello
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Phase 4
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

Figure 22.6 **Handshake Protocol Action**

Key Terms

- HELLO Extensions: request extended functionality by sending data in the extensions field.
  - For example: `max_fragment_length`, `status` request
  - The server may not oblige
  - Client may abort the handshake
- Pre-shared Secret (key): generated by client OR directly obtained from the key exchange. E.g: (DH: g^{ab} `mod p`)
- Master keys: generated from the `pre-shared secret + random.client + random.server` by applying a PRF (*pseudo random function*)
- Master key = PRF (`pre-shared secret, random.client, random.server`)

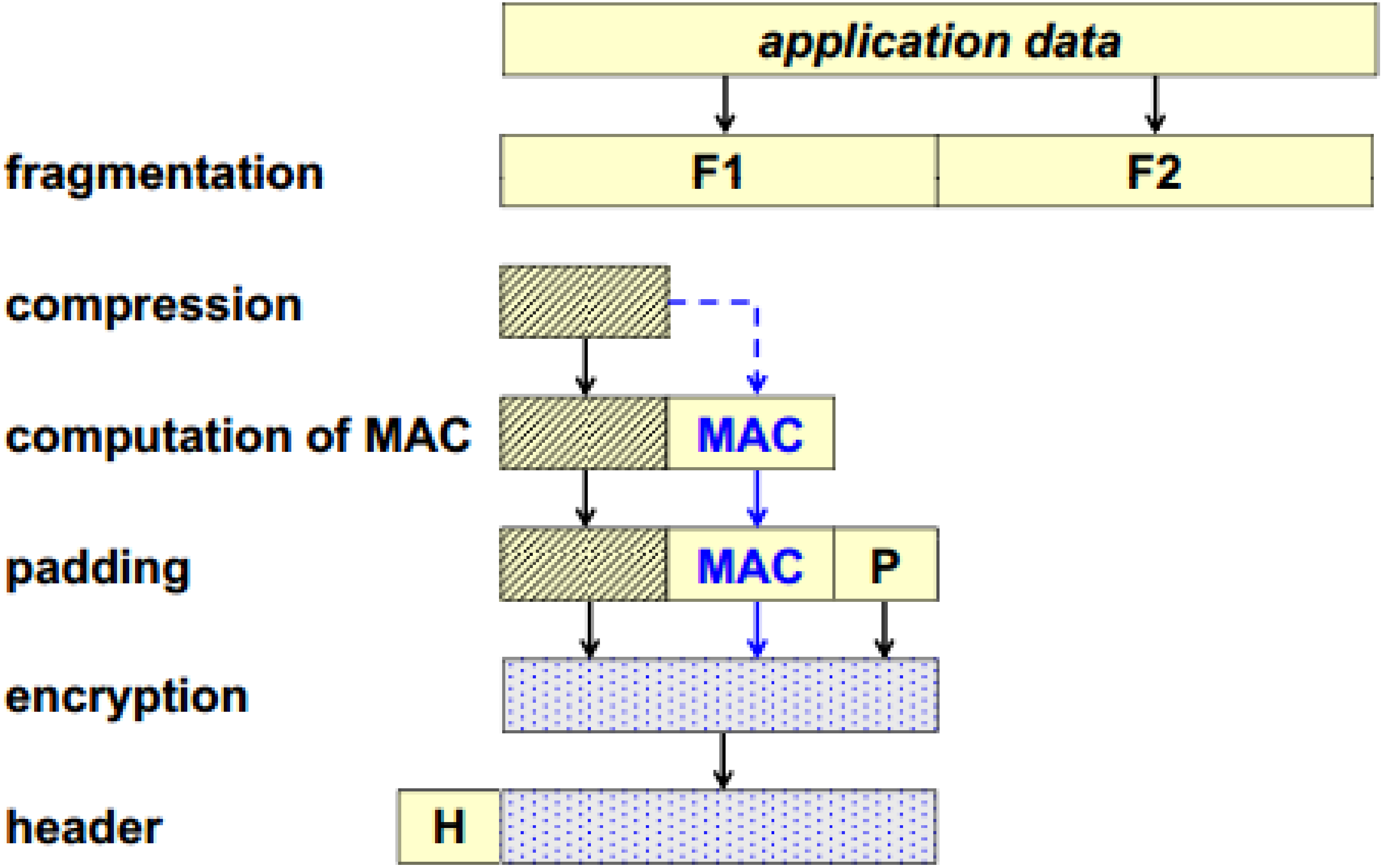
SSL: V3 Architecture



SSL handshake protocol	SSL change cipher spec protocol	SSL alert protocol	*application protocol* (e.g. HTTP)
SSL record protocol			
reliable transport protocol (e.g. TCP)			
network protocol (e.g. IP)			

- *Handshake*: enables the SSL or TLS client and server to establish the secret keys with which they communicate
- *Change cipher spec*: indicates the usage of secret key for data communication
- *Alert*: signal problems with SSL connection, give current status
- *Record Protocol*: permits the encapsulation of higher level protocols

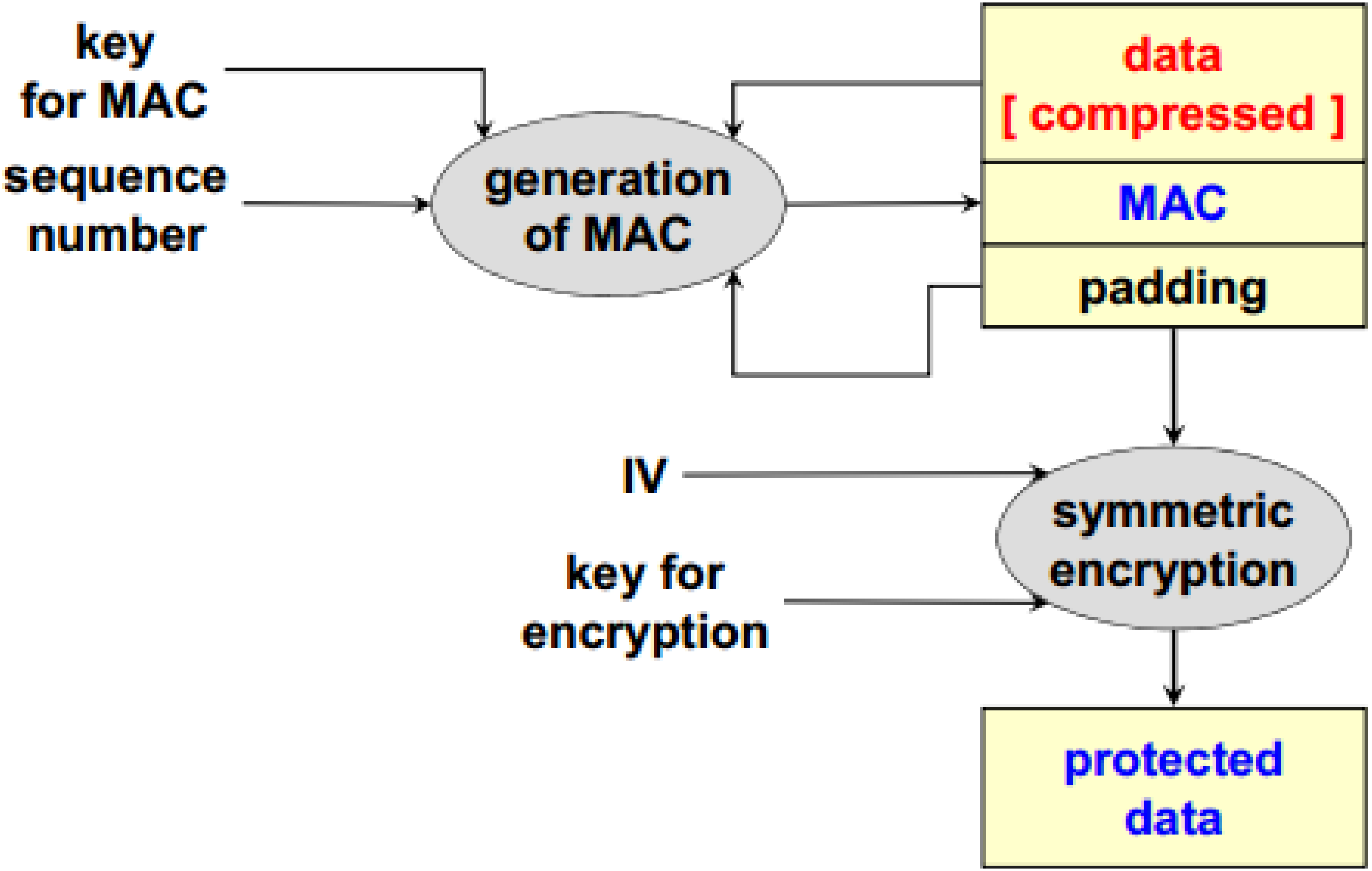
SSL3/TLS Record Protocol



SSL MAC Computation

- MAC = `message_digest(key, seq_number | type | version | length | fragment)`
- `message_digest`
  - depends on the chosen algorithm
- `key`
  - sender-write-key or receiver-read-key
- `seq_number`
  - 32-bit integer
- `type`
  - Type of record
    - change cipher spec (20)
    - alert (21)
    - Handshake (22)
    - Application data (23)
- `length`
  - length of the fragment/plaintext

Data Protection in SSL



SSL-3: new features with respect to SSL-2

- Data compression:
  - optional
  - Done before encryption

- Data encryption is optional: in order to have only authentication and integrity

- Possibility to re-negotiate the SSL connection:
  - periodical change of keys
  - change of the algorithms

Acknowledgments

- Dr Haroon Mahmood (FAST-NU)

Appendix

- SSL, TLS, HTTPS Explained (ByteByteGo, Youtube)
- SSL-TLS (University of Auckland, NZ)