MONASH University
Information Technology

FIT2093 INTRODUCTION TO CYBER SECURITY

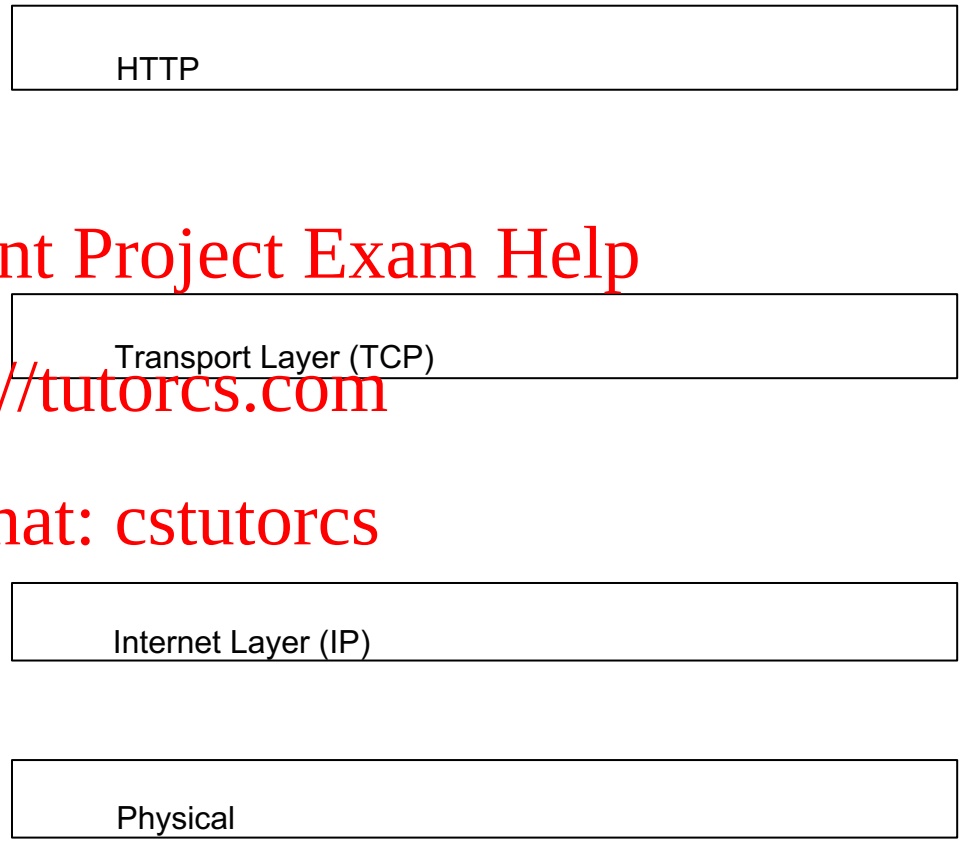
Week 6 Lecture

Security Protocols


Outline

Security Protocols

- SSL/TLS
  - Handshake
  - Record
- IPsec
  - AP headers
  - ESP headers
- Bluetooth
  - Secure Connections



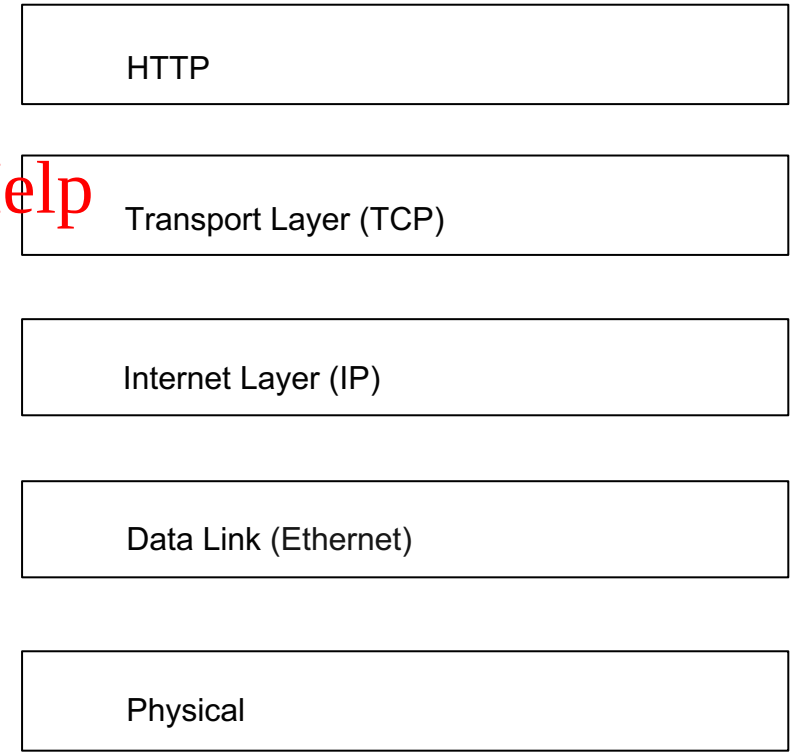
MONASH University

SSL/TLS: Securing the Transport Layer

Security for Communications

Security Protocols

- recall:
  - 7 **layers** of communications
  - e.g. OSI model
- can do security at **any** layer



Securing the Comms Channel: Gist

Security Protocols

- Gist
  - ultimately:
    - want to have (virtual) **secure channel** between two points
    - **secure** in terms of
      - CONFidentiality
      - INTegrity
      - AUTHentication

Securing the Comms Channel: Gist

Security Protocols

- how to establish secure channel? need shared key K
- *Q: why need shared key K?*

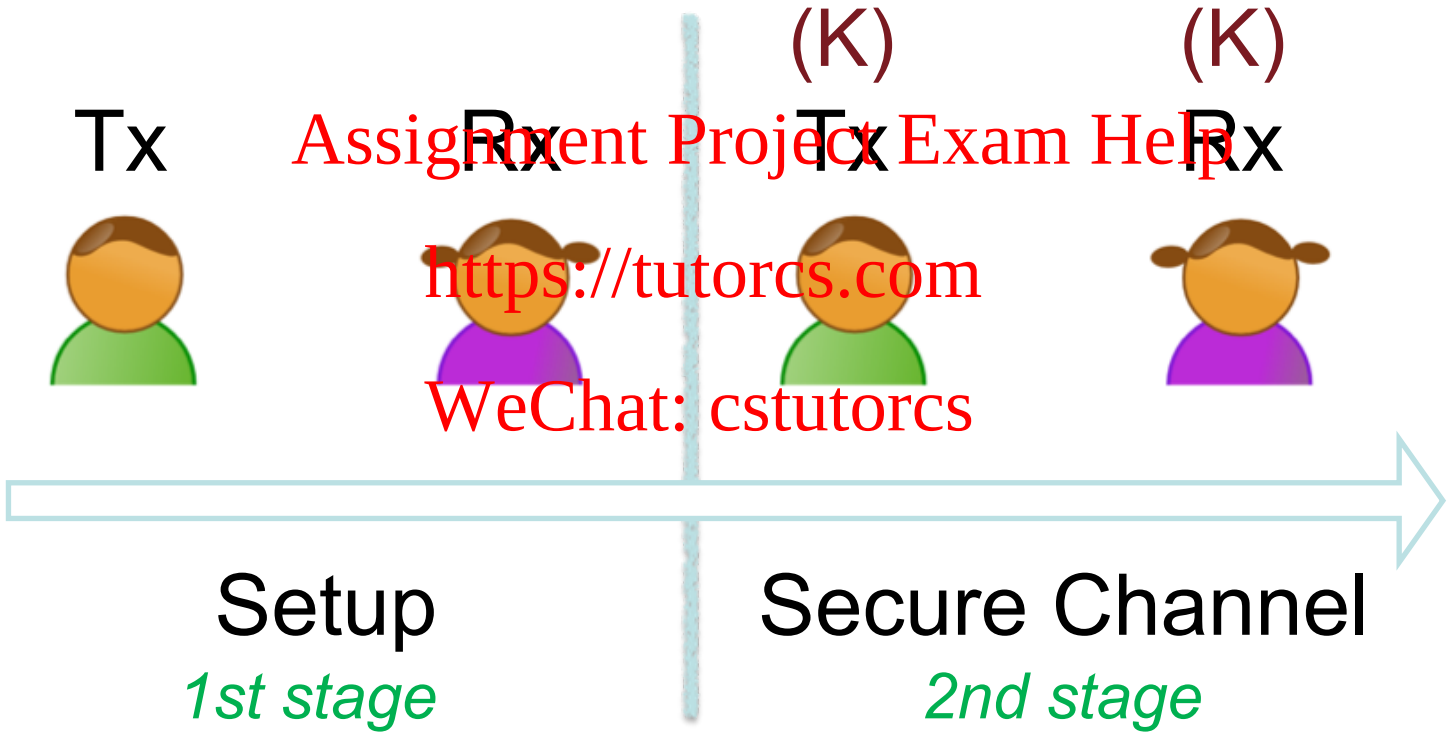
- **hybrid**:
  - symmetric key crypto (SKC):
    - fast but key distr problem
  - public-key crypto (PKC):
    - no key distr prob but slow

  - use PKC first during setup, then do SKC for bulk data

1st stage *2nd stage*

Securing the Comms Channel: Stages

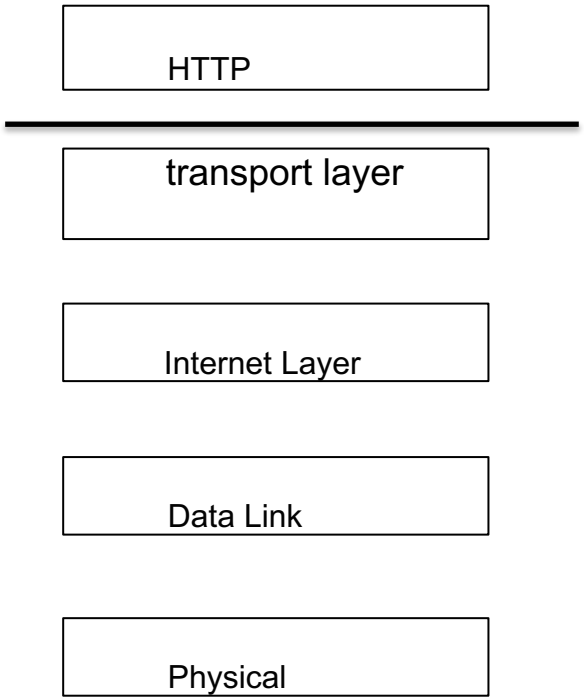
Security Protocols



Securing the Transport Layer

Security Protocols

- use SSL/TLS
- **between** Application layer & Transport layer
  - Applications need to interface to TLS
- SSL (Secure Sockets Layer):
  - by Netscape (de facto browser then)
- **TLS** (Transport Layer Security)
  - by Internet Engineering Taskforce
  - latest version: 1.3 (IETF RFC 8446, 2018)



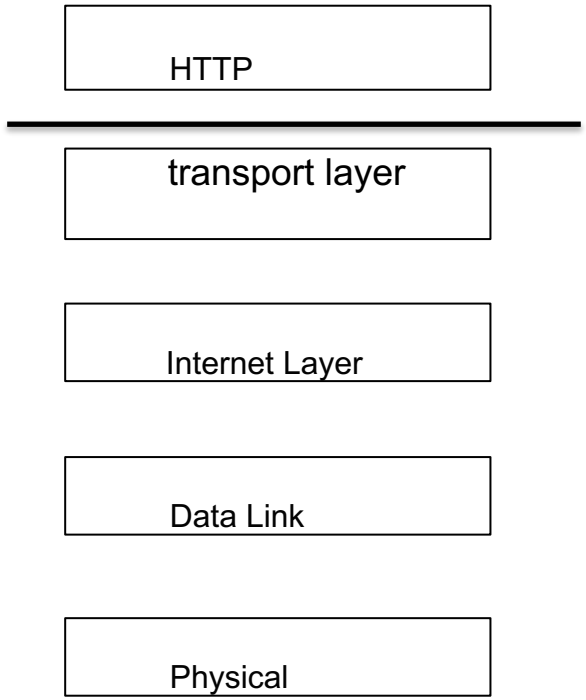
TLS v1.3

Security Protocols

- perfect forward secrecy (PFS) mandatory
- part of openSSL since TLS v1.3 (2018)
- MD5 (broken hash function) removed
- SHA-224 (member of SHA-2 family) removed

vs TLS v1.2 [2008]

- support for Authenticated Encryption AES modes
  - Galois Counter Mode (GCM)
  - Counter with CBC-MAC (CCM)



Perfect Forward Secrecy

Security Protocols

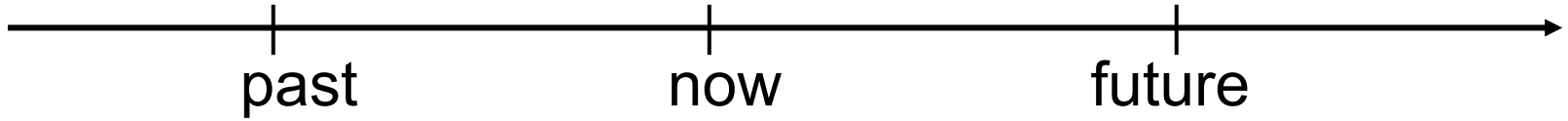
- Günther [Eurorypt 1989]
- Diffie et al. [1992]

- The right to **retain security**: "my security now, should remain secure in future, even if secrets compromised in future"

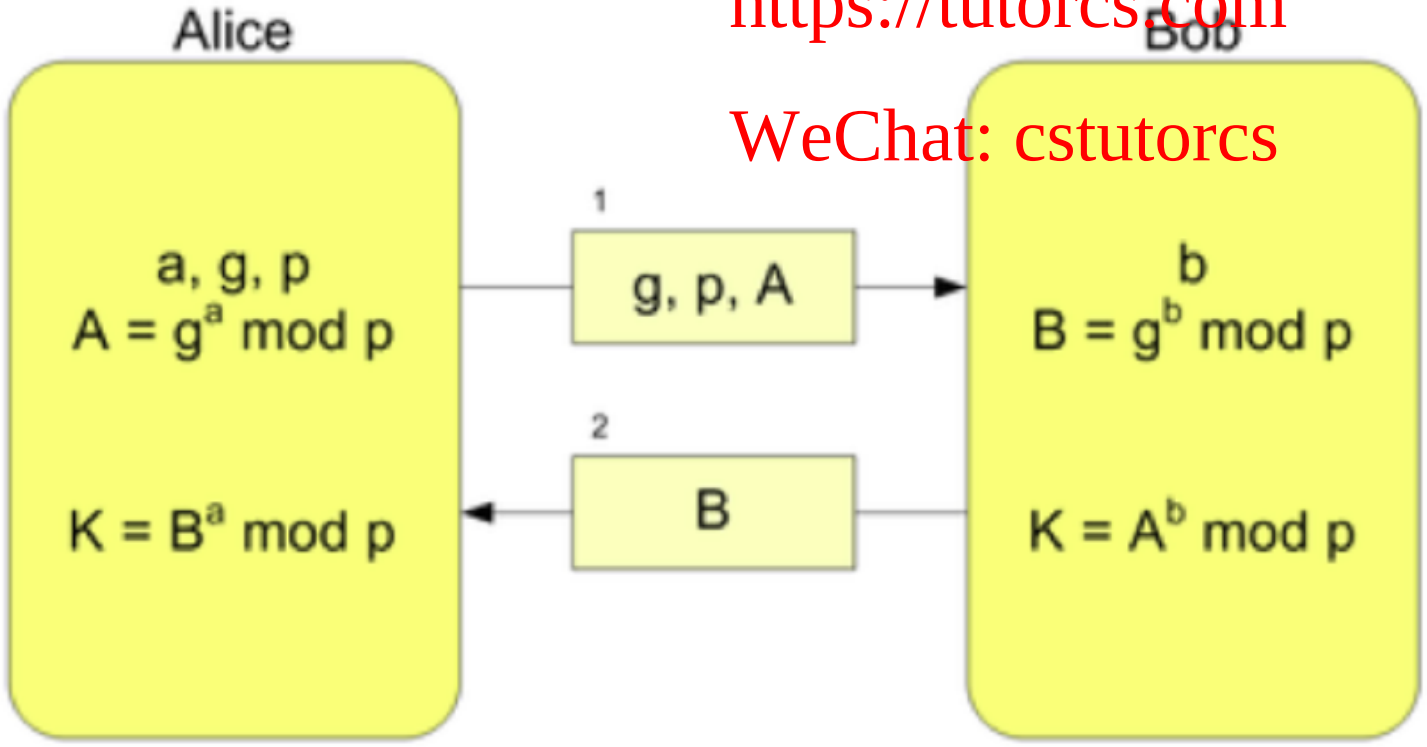
"If my security depends on the future, what's the point of doing security now? "

Perfect Forward Secrecy (PFS)

Security Protocols

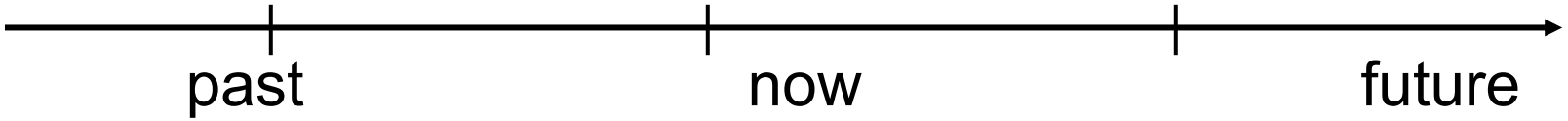


- Recall: a and b are long-term private keys of Alice and Bob
- if b is compromised in the future, is K_{now} still secure?



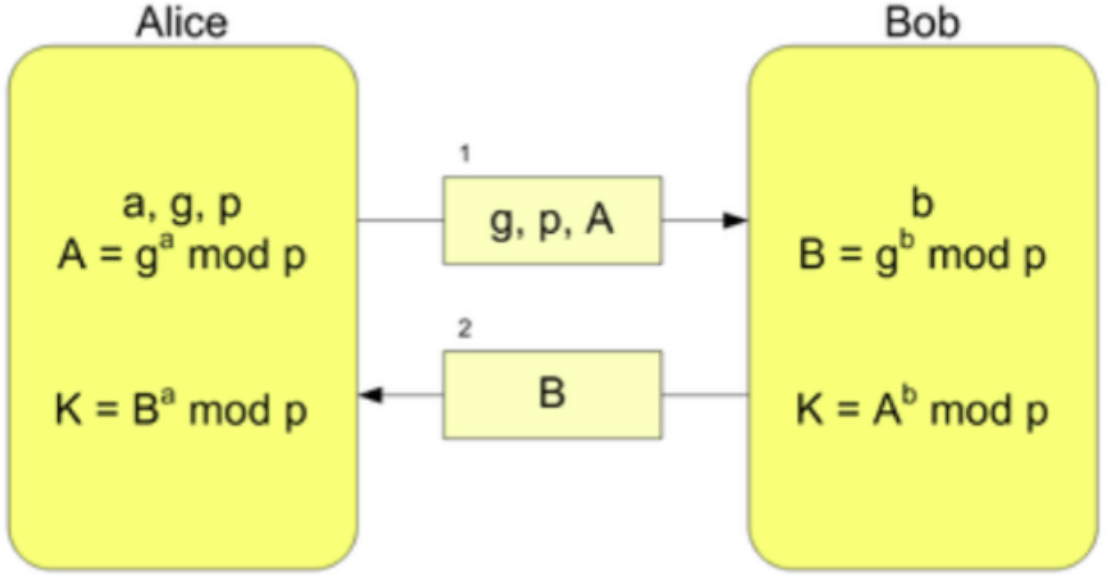
Perfect Forward Secrecy (PFS)

Security Protocols



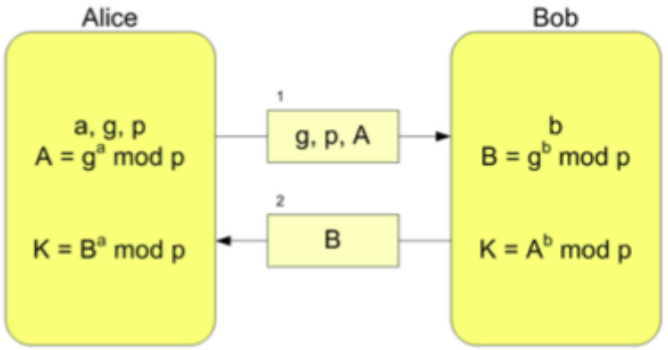
- **long-term** private keys: a and b, used in computation of K_{past} , K_{now} , K_{future}
- compromise of single long-term private key compromises all keys throughout time: K_{past} , K_{now} , K_{future}

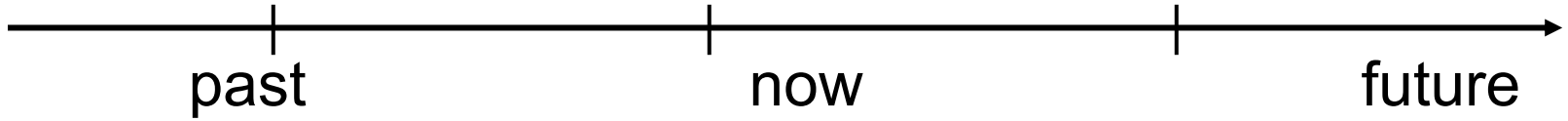
- all security lost, no option for recovery of security



Perfect Forward Secrecy (PFS)

Security Protocols





- **ephemeral** private keys: a_{past}, a_{now}, a_{future} and b_{past}, b_{now}, b_{future}, used in computation of K_{past}, K_{now}, K_{future}
  - use ephemeral keys for **encryption** key exchange
  - use long term key only for **signing** ephemeral keys

  → compromised long term key only affects future authentication (MITM attacks)

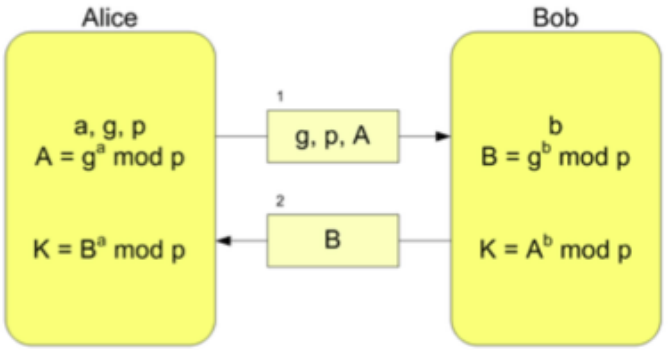
Q: With ephemeral encryption keys and long term keys for signing, why doesn't compromise of long term key now affect past security?

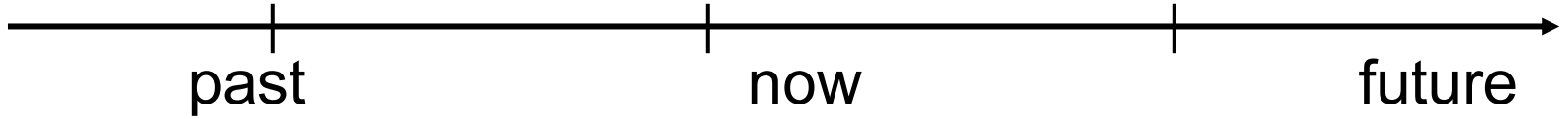
- compromise of a_t only compromises K_t for that period t

- so: not all is lost for security, still some remnant security
  - compromise resilience/tolerance

Perfect Forward Secrecy (PFS)

Security Protocols





- known since 1989 but industry not worried
- U.S. NSA PRISM:
  " *(encrypted) internet communications collected & stored from internet companies* …
- 2011: Google
- 2014: Microsoft, Facebook, Yahoo

Securing the Transport Layer: SSL/TLS

Security Protocols

- TWO Sub-protocols
  - *for 1st stage*
  - TLS **Handshake** protocol
    - use PKC: AUTH server & client
    - negotiate parameters
    - establish shared key K
      - based on Diffie-Hellman

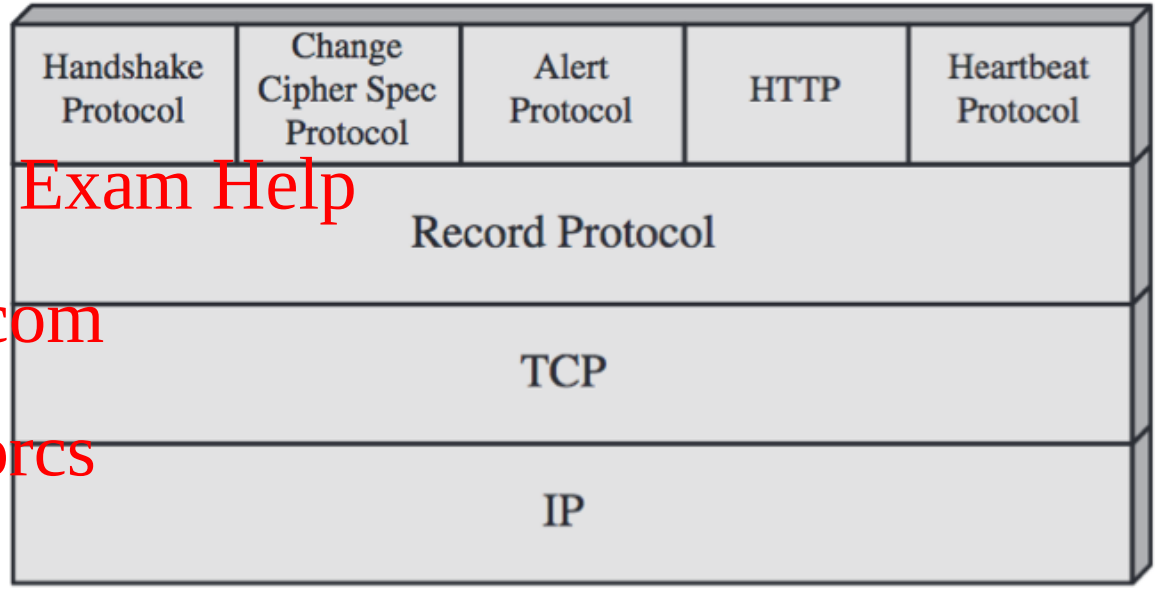
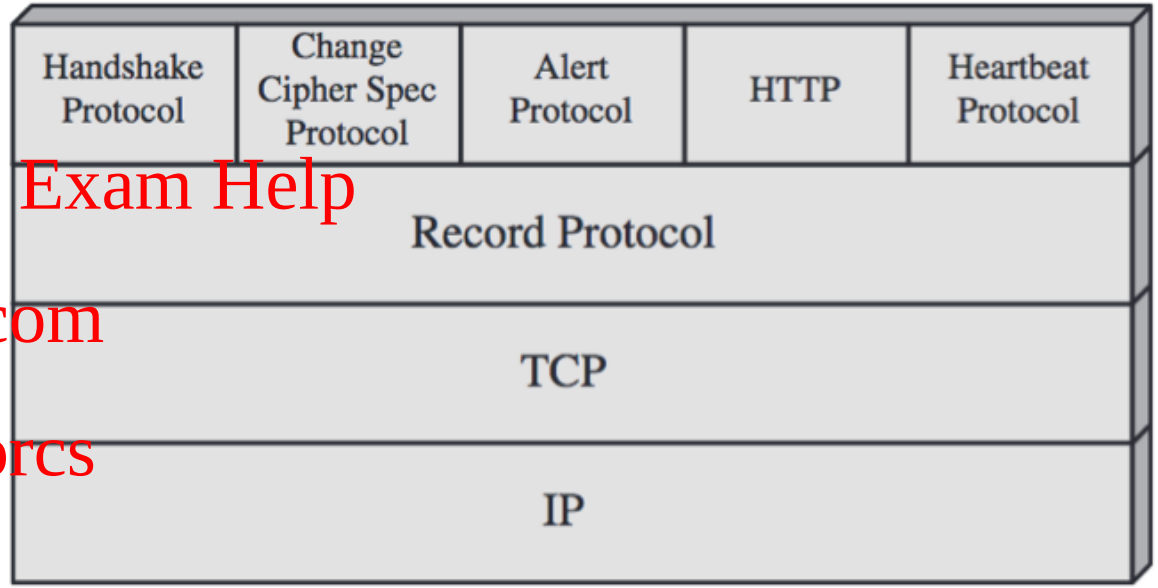


Figure 22.4 **SSL/TLS Protocol Stack**

Securing the Transport Layer: SSL/TLS

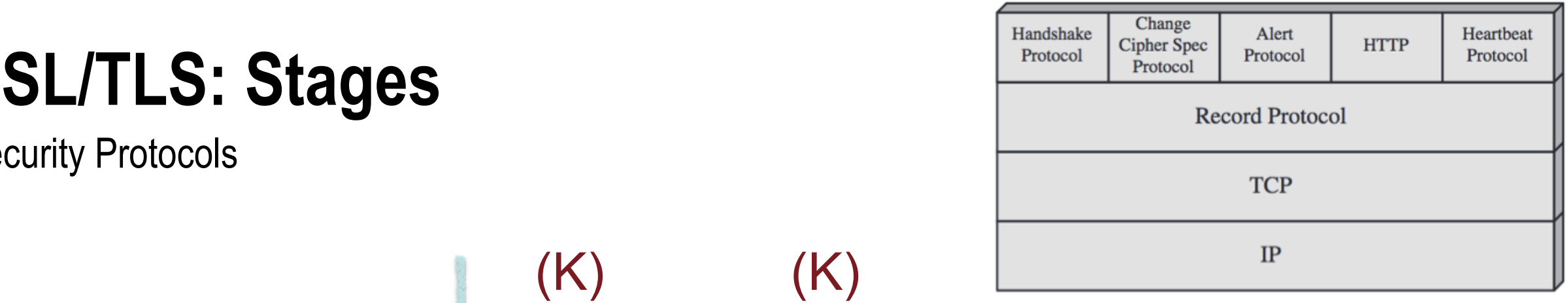
Security Protocols

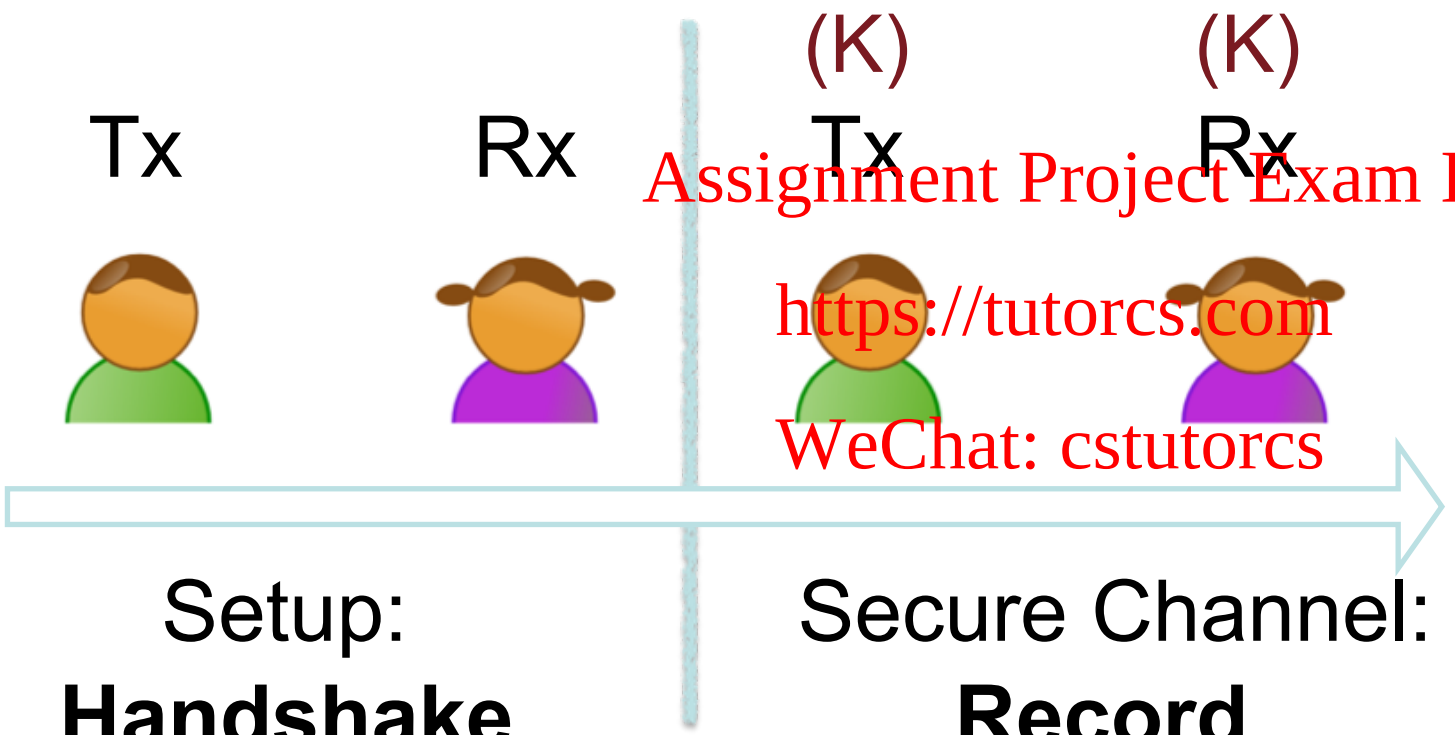
- TWO Sub-protocols
  - *2nd stage*
  - TLS **Record** protocol
    - use SKC
    - securely transport/encapsulate (*like envelope*) the data
      - CONF: encryption
      - INT: msg authentication (MAC)



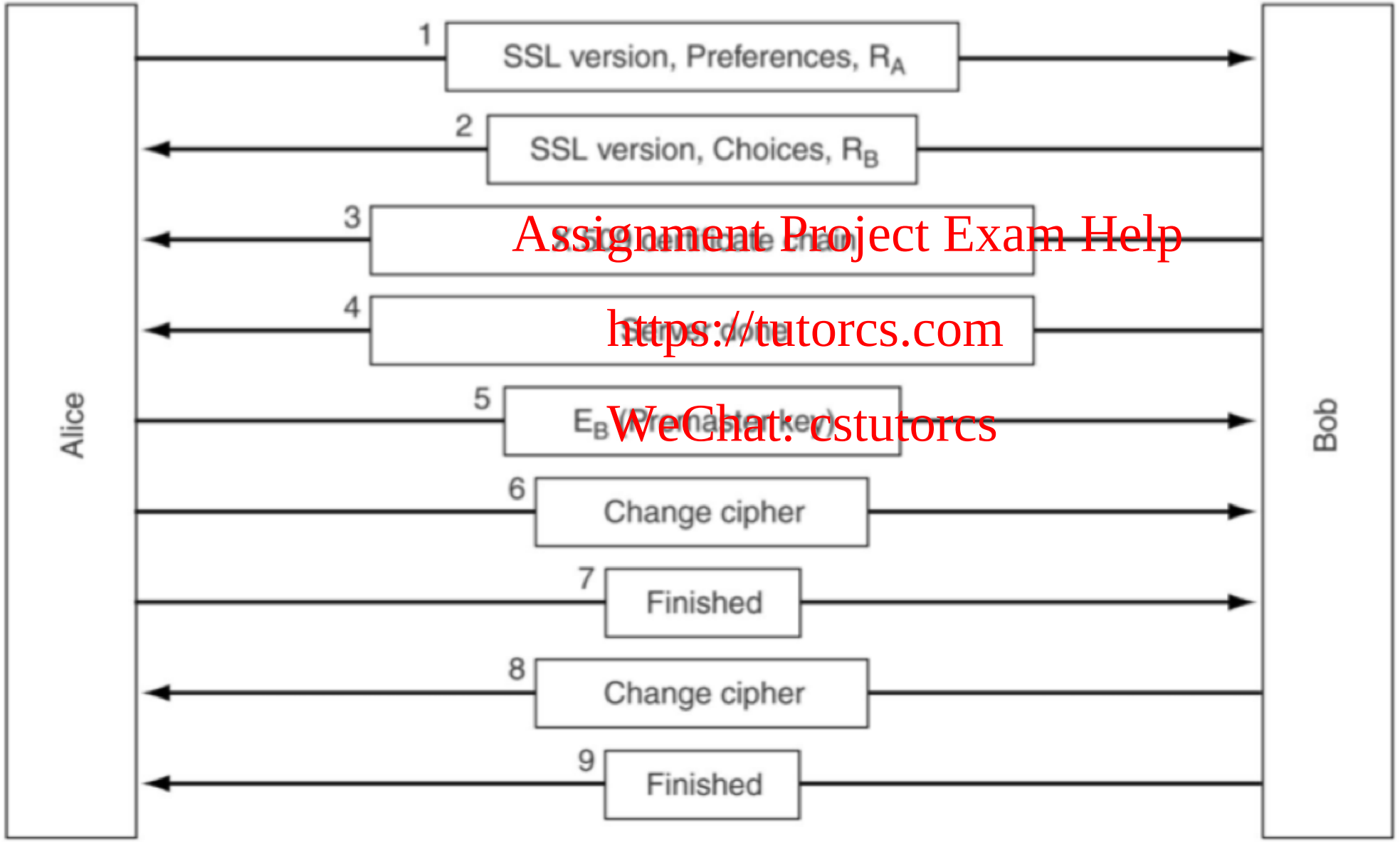
SSL/TLS: Stages

Security Protocols



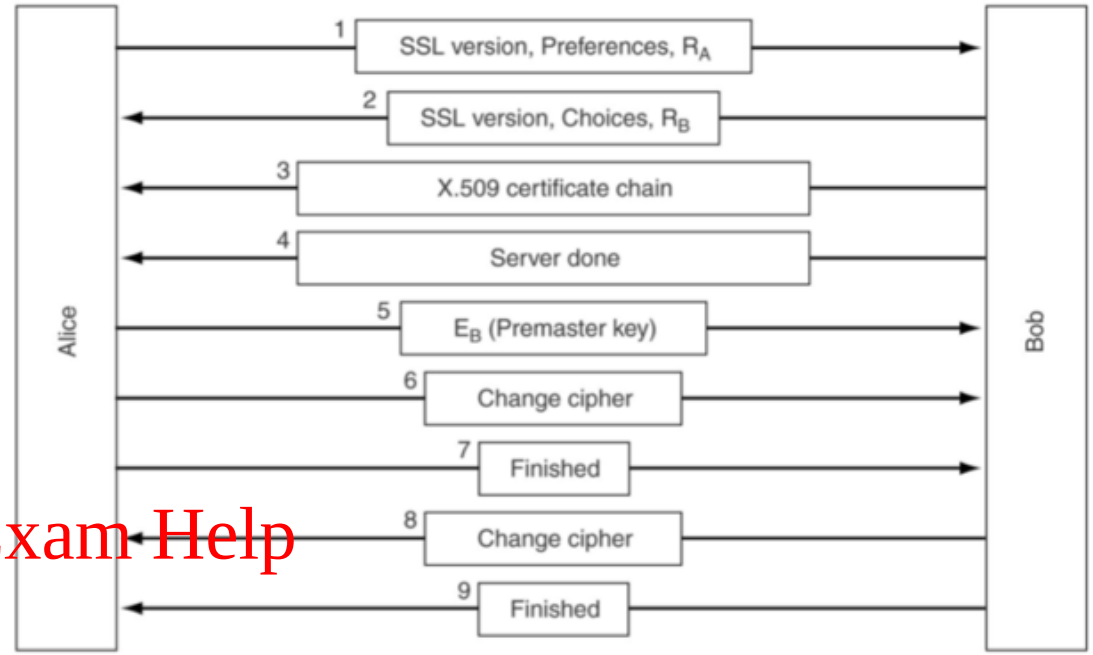


SSL/TLS: Handshake Protocol



SSL/TLS: Handshake Protocol

Security Protocols



- Gist: solve the key distribution problem
  - 1,2: nonces R_A , R_B for **freshness**
  - 3: ensure public key integrity (**digital certificates**) vs key replacement attacks
  - 5: using **PKC** for **key transport**

Q: What kind of attacks are prevented using random nonces R_A , R_B?

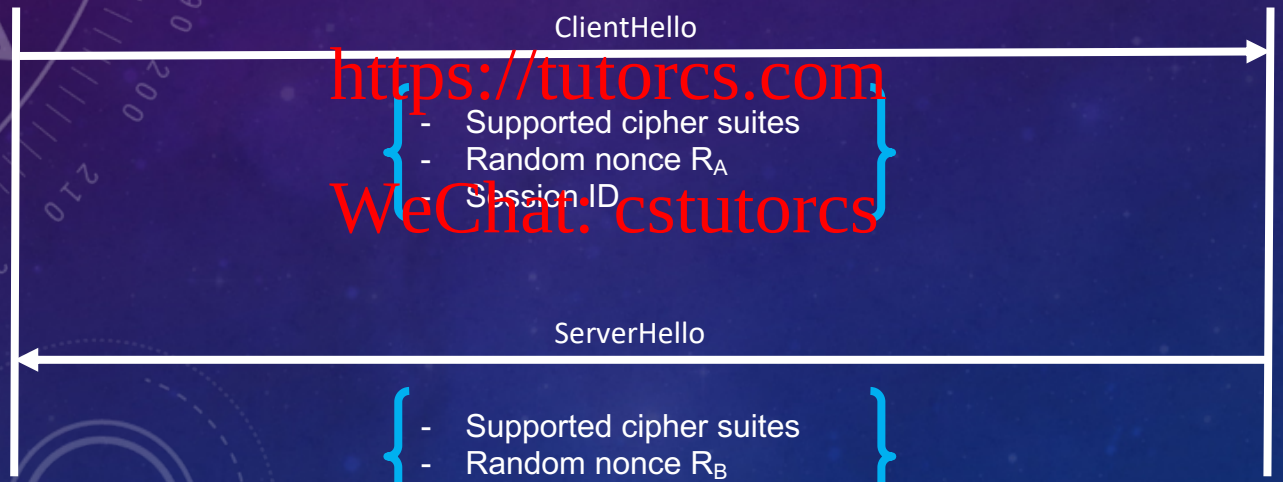
Activity (2 mins)

1) Click the latest link in the Zoom chat
2) Add your question response to the Ed forum

#TLS HANDSHAKE: A CLOSER LOOK

Handshake Part 1 – Establishing Security Capabilities

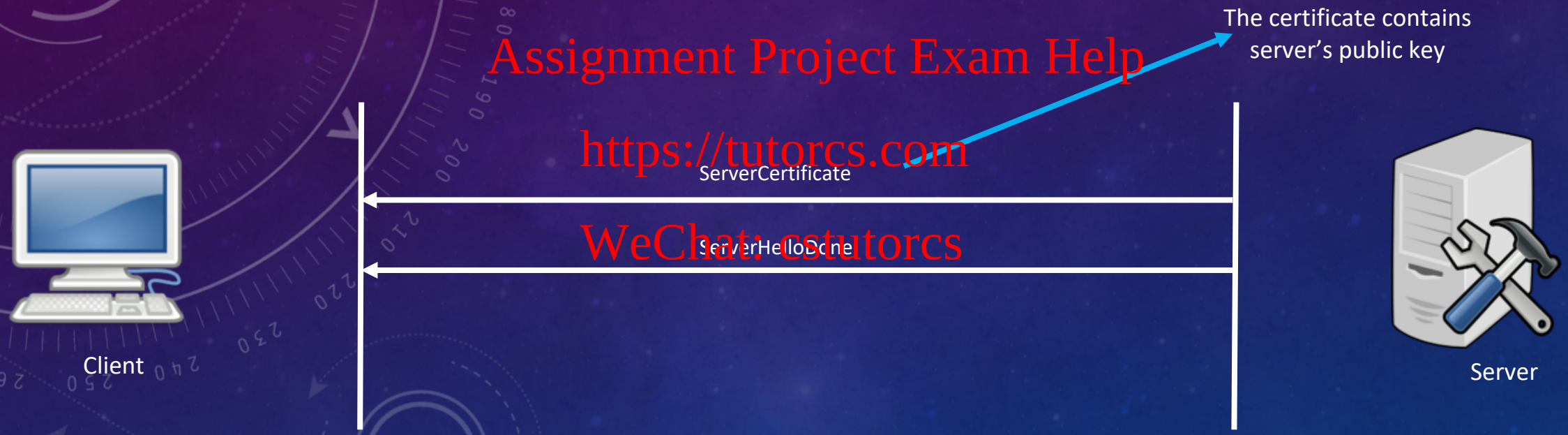
Client



Server

#TLS HANDSHAKE: A CLOSER LOOK

Handshake Part 2 – Server Authentication and Key Exchange



#TLS HANDSHAKE: A CLOSER LOOK

Handshake Part 3 (if client authentication used) –
Client Authentication and Key Exchange

Client

Server

#TLS HANDSHAKE: A CLOSER LOOK

Handshake Part 4 – Key Generation

Client will create pre-master secret and encrypt it by using the public key contained in the server's certificate

Client

ServerHelloDone

Client Certificate (Key exchange)

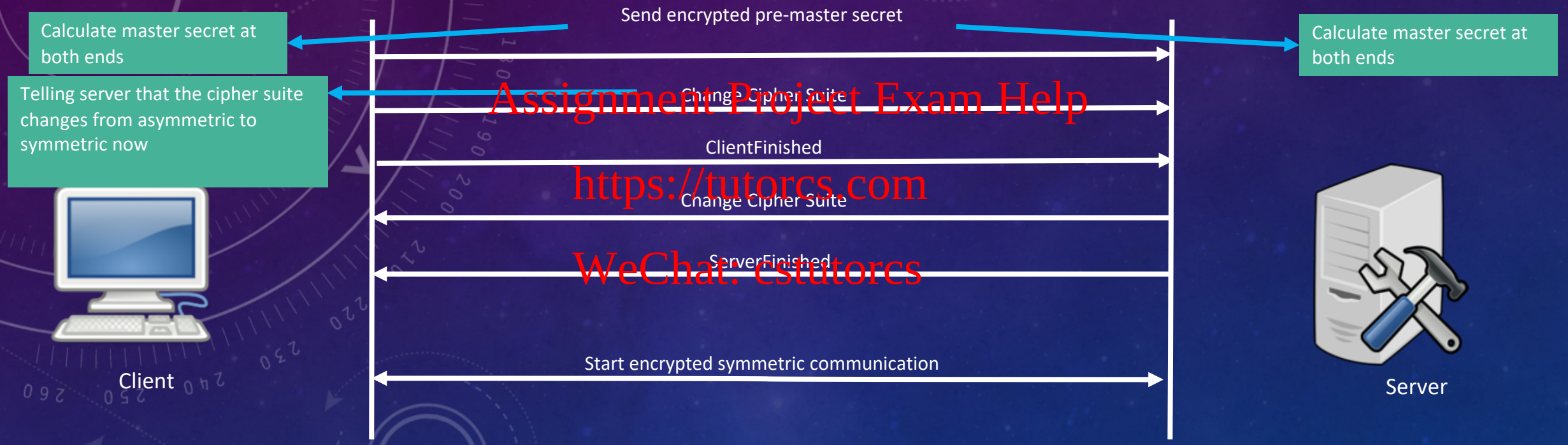
Client Certificate

Send encrypted pre-master secret

Server

#TLS HANDSHAKE: A CLOSER LOOK

Handshake Part 5 – Finish



Digital Signature in SSL/TLS

Security Protocols

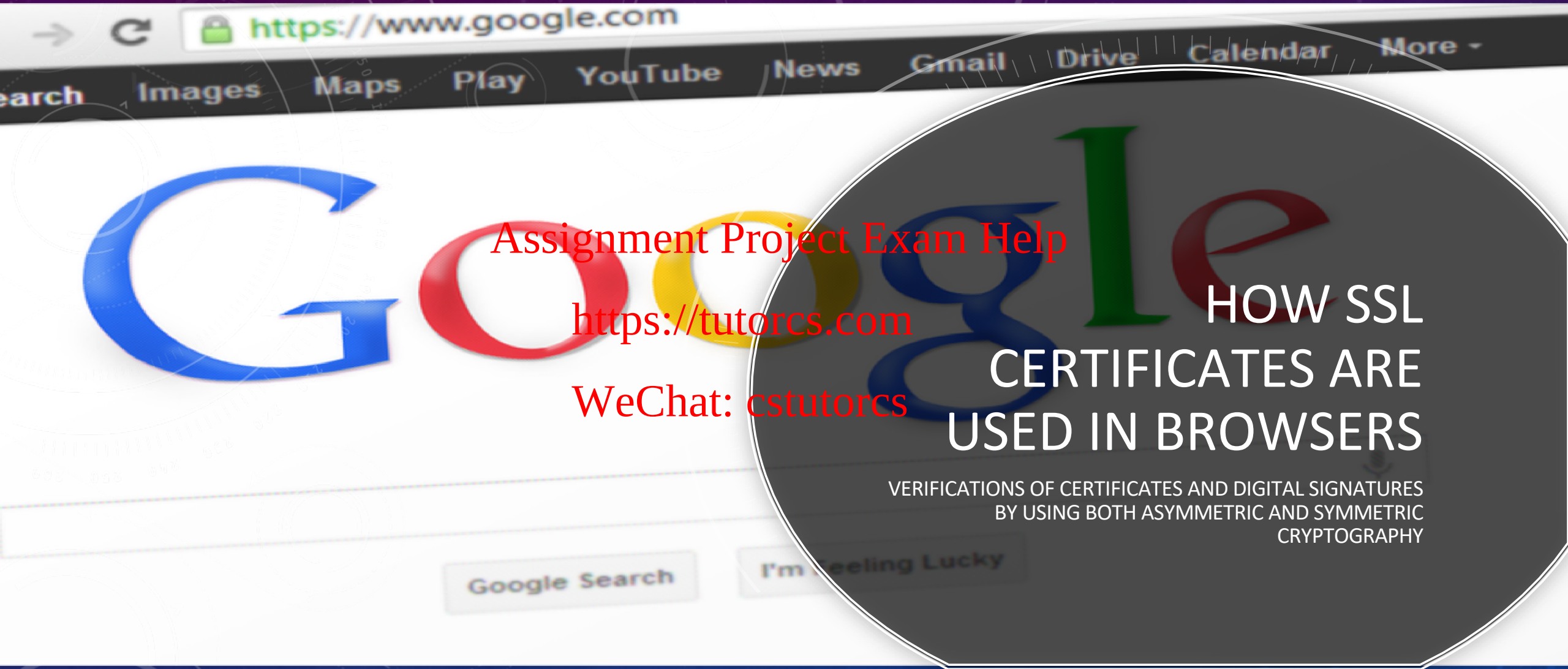
- A **certificate** provides additional information for a **public key**:
  - Owner **identity** (e.g. URL) of the matching private key
  - **Validity** (expiration date and time)
  - Subject name
  - Issuer name
  - other parameters
  - **digital signature** by a trusted **certification authority** (CA)

Trust of Digital Signature in SSL/TLS

Security Protocols

- A **trusted certificate** is digitally signed by a known CA
- Browsers (Chrome, Firefox, IE, Safari, etc.) come **pre-installed** with a list of these CAs and their authentic **public keys**
- Browsers can then **verify** certificates as coming from these CAs

- Note: Do not trust ”self-signed” certificates! Anyone can “self-sign” a fake certificate



HOW SSL CERTIFICATES ARE USED IN BROWSERS

VERIFICATIONS OF CERTIFICATES AND DIGITAL SIGNATURES BY USING BOTH ASYMMETRIC AND SYMMETRIC CRYPTOGRAPHY

1. Google server obtains SSL certificate which is digitally signed by 3rd party called CA

Certificate Authority

2. CA's public keys will be preinstalled inside the client's PC

3. When you type a URL in the web browser e.g. www.google.com , a request will be sent from your browser to a Google web server for secure pages

** Note that SSL certificate is created by using CA's private key

Client

4. Google server sends its public key with SSL certificate

Web Server
www.google.com

5. The green padlock verifies that the certificate indeed belongs to a www.google.com server and has a valid CA signature by one of the preinstalled CA's

Public Key

Private Key

7. However, the client would not want to send the secret key in plain text. Thus, it uses Google server's public key to encrypt the secret key first.

8. When Google server receives the shared key encrypted by using its public key, it can decrypt it by using its private key.

Client

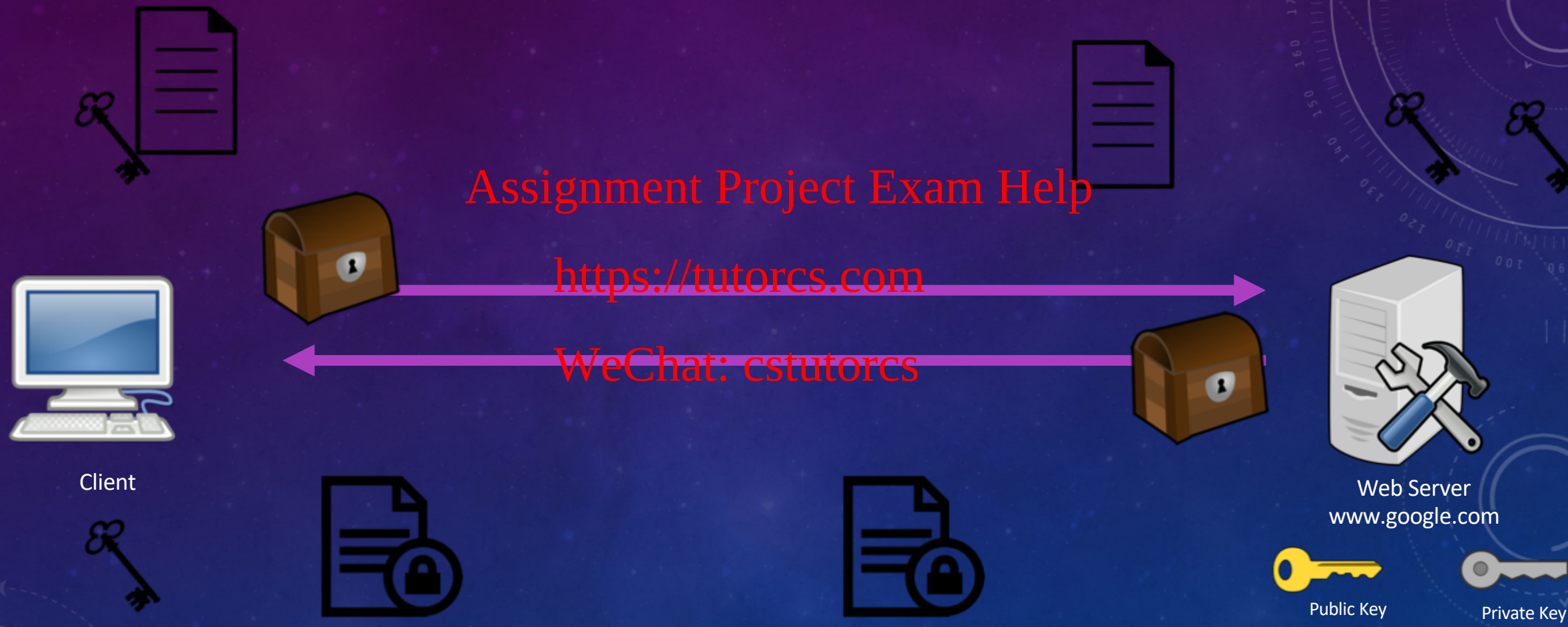
Web Server
www.google.com

Symmetric Key

6. Verification done. Client will generate a secret shared (symmetric) key

Public Key

Private Key



TLS EXAMPLE: HOW ATTACKS ARE PREVENTED

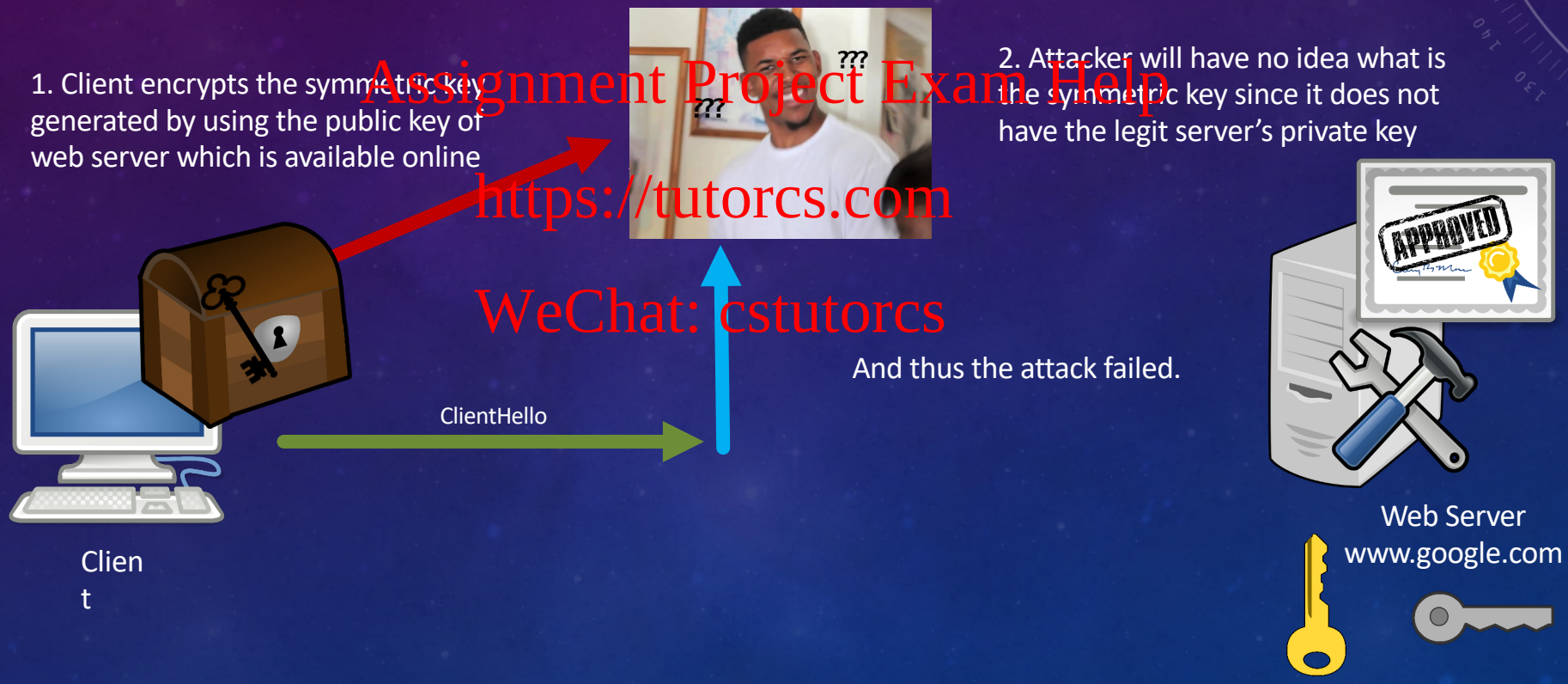
EXAMPLE 1: MODIFICATION ATTEMPT BY USING REAL CERTIFICATE

But what if attacker imitated themselves as both legit server/receiver and client / sender?

TLS EXAMPLE: HOW ATTACKS ARE PREVENTED

EXAMPLE 1: MODIFICATION ATTEMPT BY USING REAL CERTIFICATE

Sounds like the attacker will succeed, right? However with TLS we are using RSA public key exchange upon symmetric key exchange, thus even the attacker intercepted the information, they have no way to access/decrypt it.
Now assume the public key exchange is done, the client is now going to send server the symmetric key that they are going to use for further connection.



TLS EXAMPLE: HOW ATTACKS ARE PREVENTED

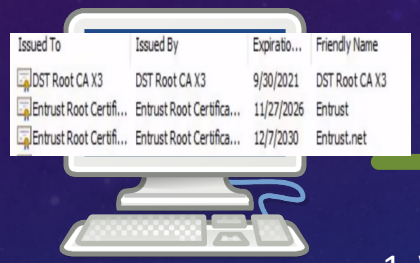
EXAMPLE 2: MAN-IN-THE-MIDDLE ATTACK WITH FAKE CERTIFICATE

Now consider another scenario, what if attacker imitated themselves as both legit server/receiver and client / sender by using a fake certificate for google.com with attacker's key?

2. Then the attacker itself act as the client and send the fake certificate to the client

4. But obviously in this case, the certificate generated by attacker is not certified by authorised CA, thus the certificate will not be approved by the client

3. Client will look up from the online public database for the public key of the certificate received to verify the certificate



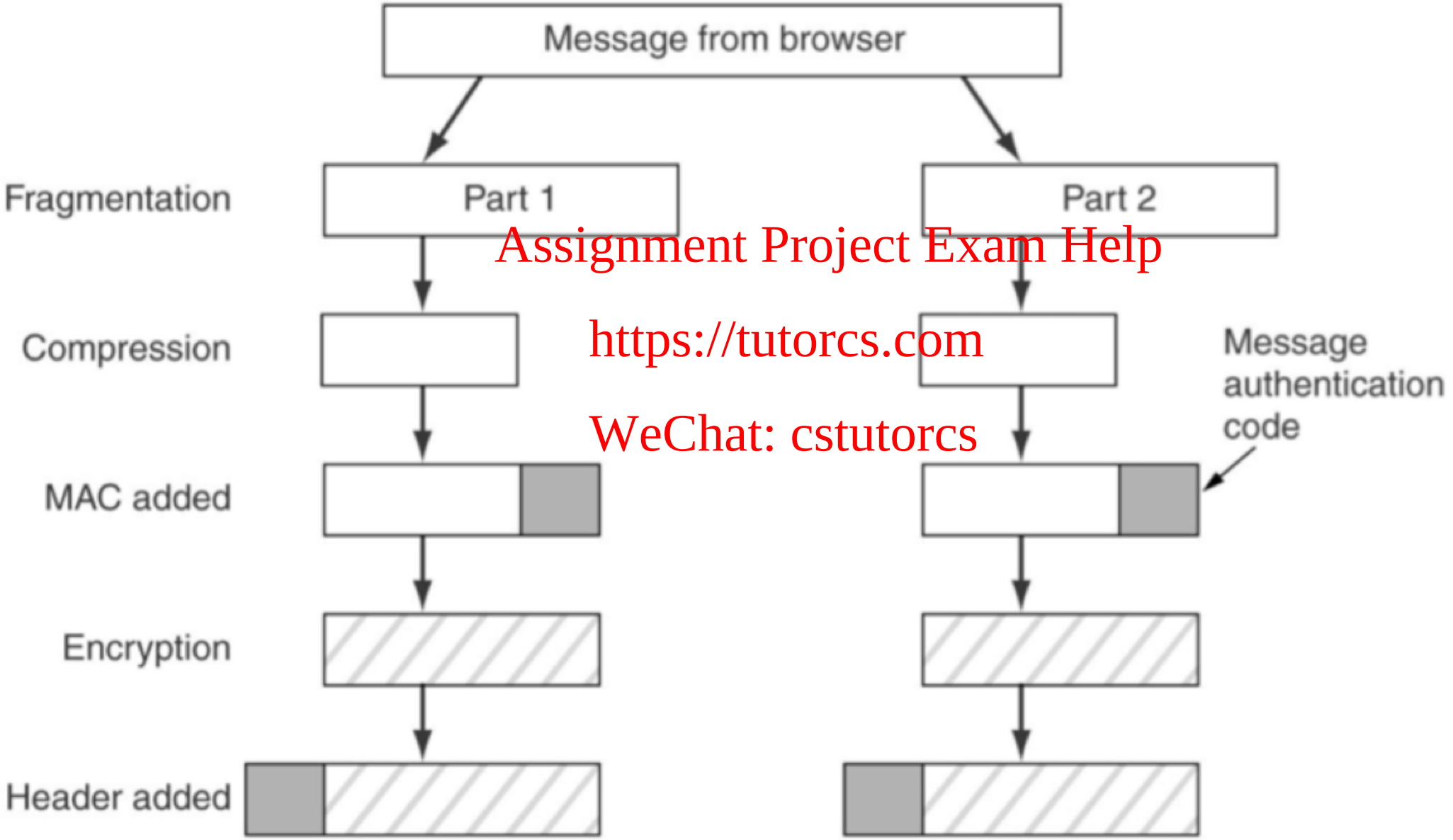
ClientHello

5. The website will then be labelled as not secure and again the attack failed

Clien
t

1. When client requested for a webpage from server, the attacker intercept the message from the network and direct it to themselves so that the legit server will not receive the message

SSL/TLS: Record Protocol



SSL/TLS: Record Protocol

Security Protocols

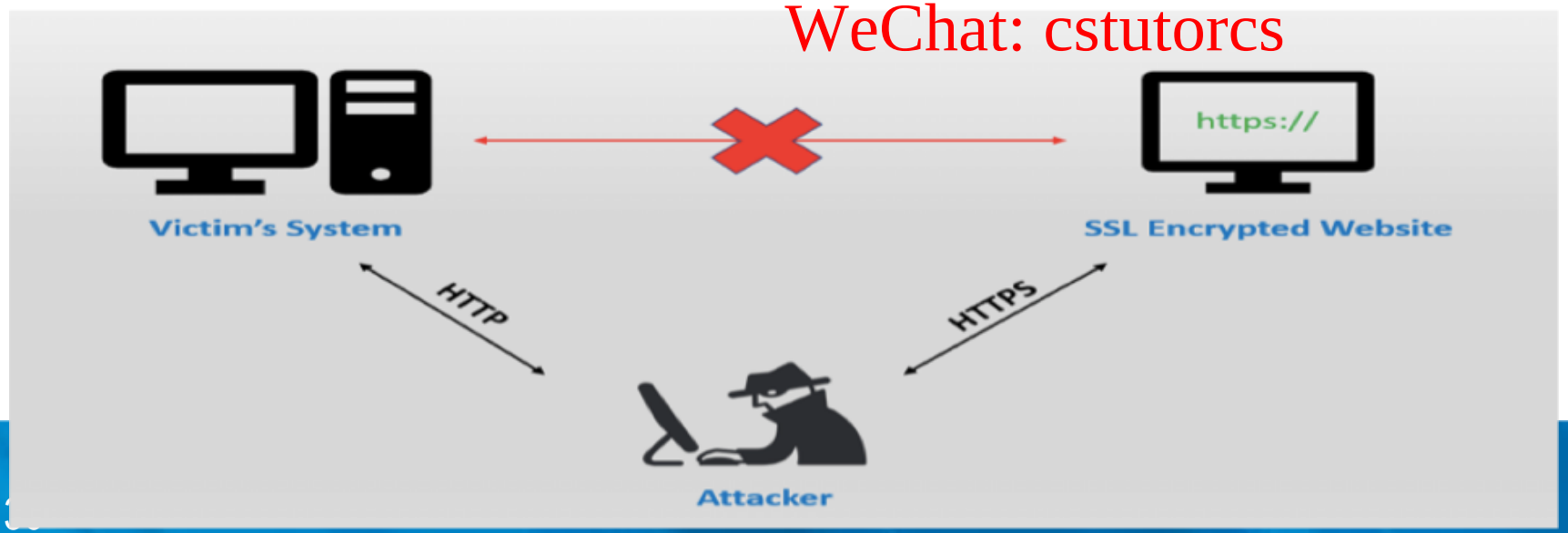
- Gist:
  - after handshake, virtual secure channel between browser and server
  - all data will be secured using authenticated encryption

  - **authenticated encryption**
    - integrity via **message authentication**
    - confidentiality via **encryption**

SSL Stripping/Downgrade Attack


Security Protocols

1. User types into browser a URL address (eg. www.amazon.com)
2. Browser requests secure connection: https://www.amazon.com
3. MITM attacker intercepts request and answers with **downgrade request**
4. Browser sends a downgraded HTTP request: http://www.amazon.com
5. Attacker forwards request to real Amazon server as https://www.amazon.com – MITM attacker can see all plaintext in http msgs!



Q: How can servers and browsers defend against SSL Stripping downgrade attacks while still supporting http websites?

Attacks on SSL/TLS

Security Protocols

- SSL Stripping: Attacker modifies web traffic to trick a client into accepting http (unencrypted) connection instead of https
  - Countermeasure: Server tells browser upon first visit to only accept HTTPS (not HTTP) for future requests to this domain

- BEAST (Browser Exploit Against SSL/TLS)/POODLE (Padding Oracle On Downgraded Legacy Encryption): Attacker exploits bugs in encryption/MAC mode of operation algorithm/implementation in TLS **v1.0** to extract information on secret key
  - Countermeasure: patches to TLS algorithms/implementations

- Compression Ratio Info-leak Made Easy (CRIME): TLS has a compress-then-encrypt mode; **compressed data length** not hidden by encryption, reveals information to attacker on secret web site cookies
  - Countermeasure: disable TLS compression, avoid if possible record layer compression

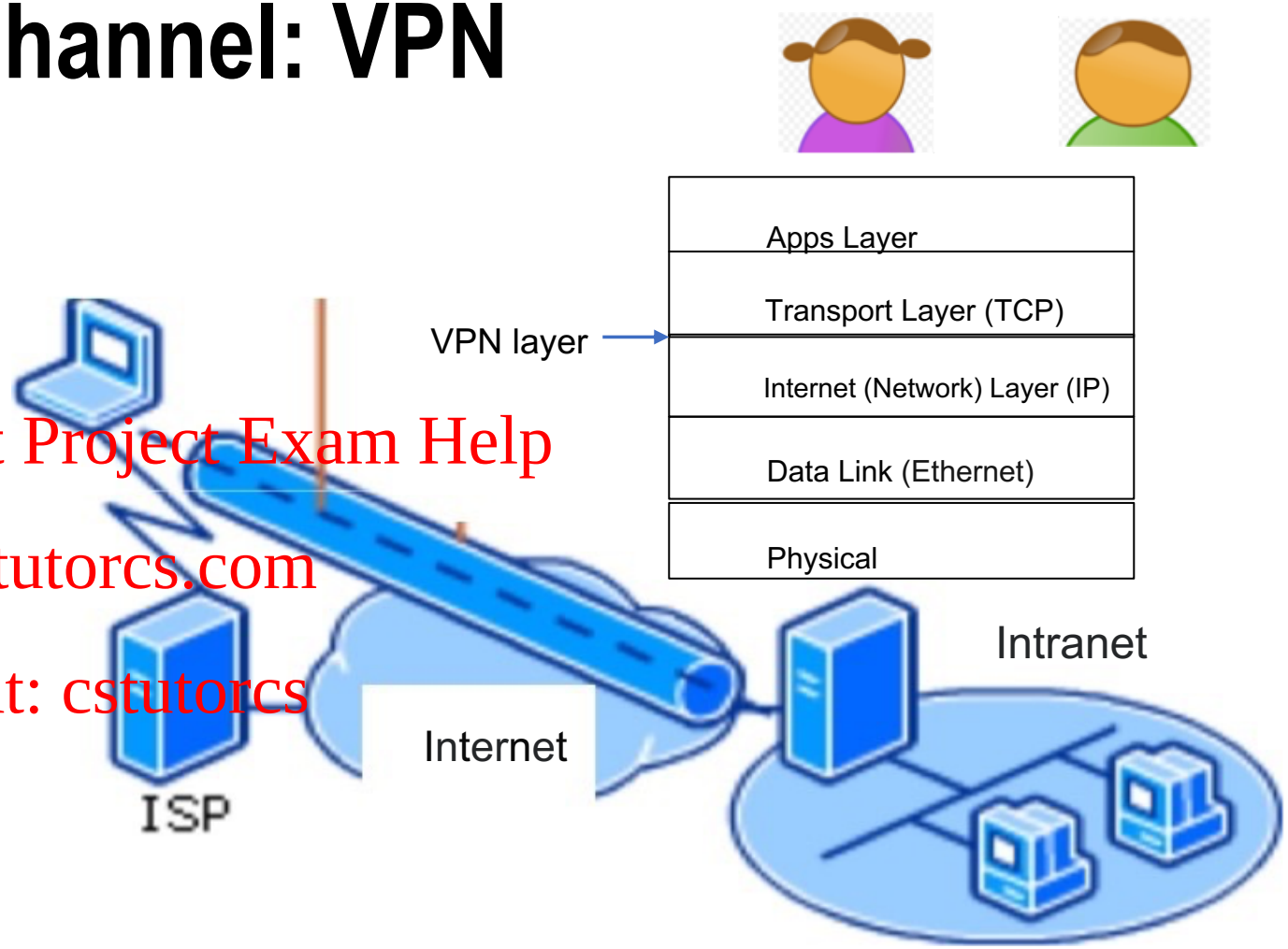
MONASH University

IPsec: Securing the Network Layer

Securing your Comms Channel: VPN

Security Protocols

- VPN (Virtual Private Network)
  - encrypted channel / tunnel
- routes packets between different networks
- secure channel via IPsec



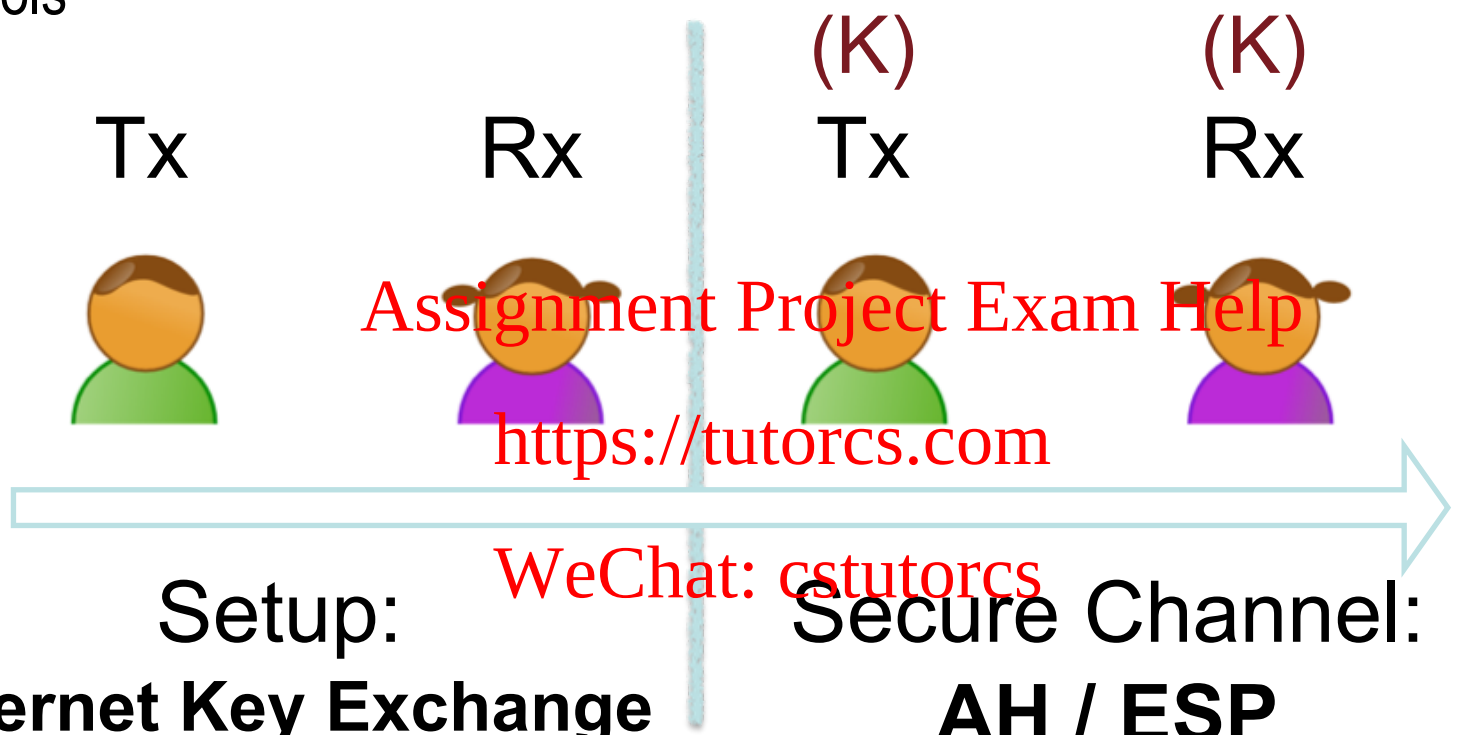
Q: Why should VPN work between Transport and Network layers (rather than between App and Transport layer)?

Activity (2 mins)

1) Click the latest link in the Zoom chat
2) Add your question response to the Ed forum

IPsec: Stages

Security Protocols



Setup:

Internet Key Exchange

(IKE) protocol

(*based on Diffie-Hellman*)

Secure Channel:

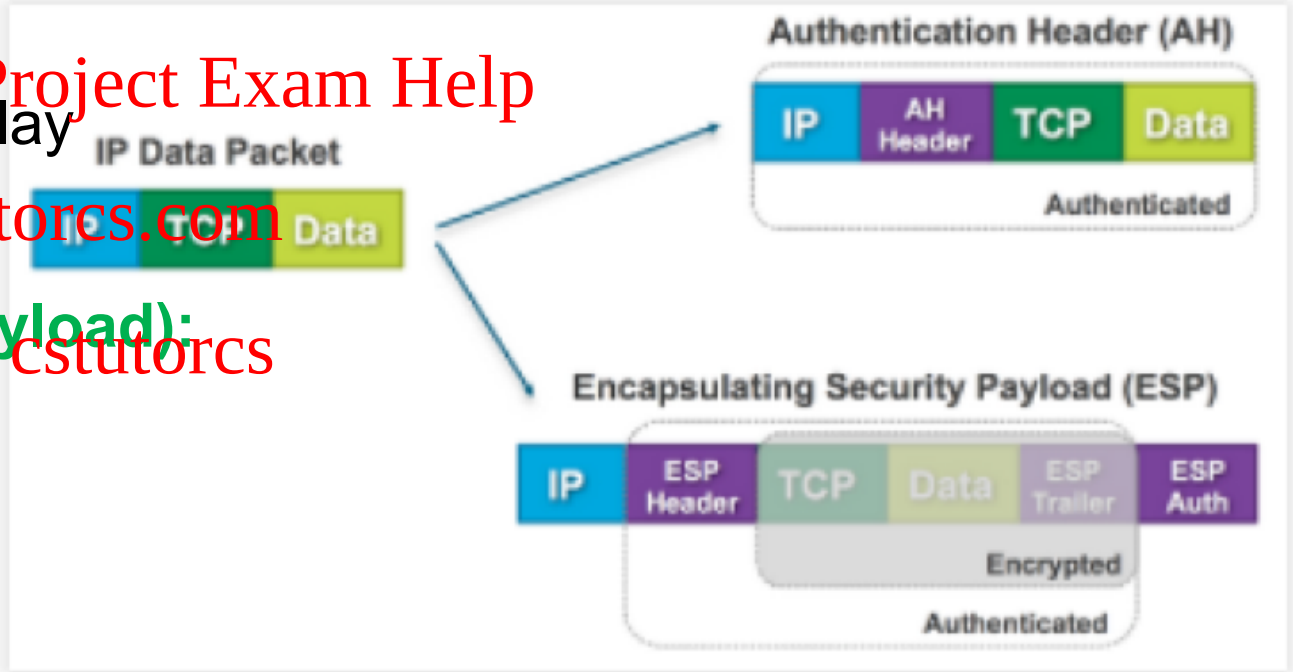
AH / ESP

methods

IPsec: Methods

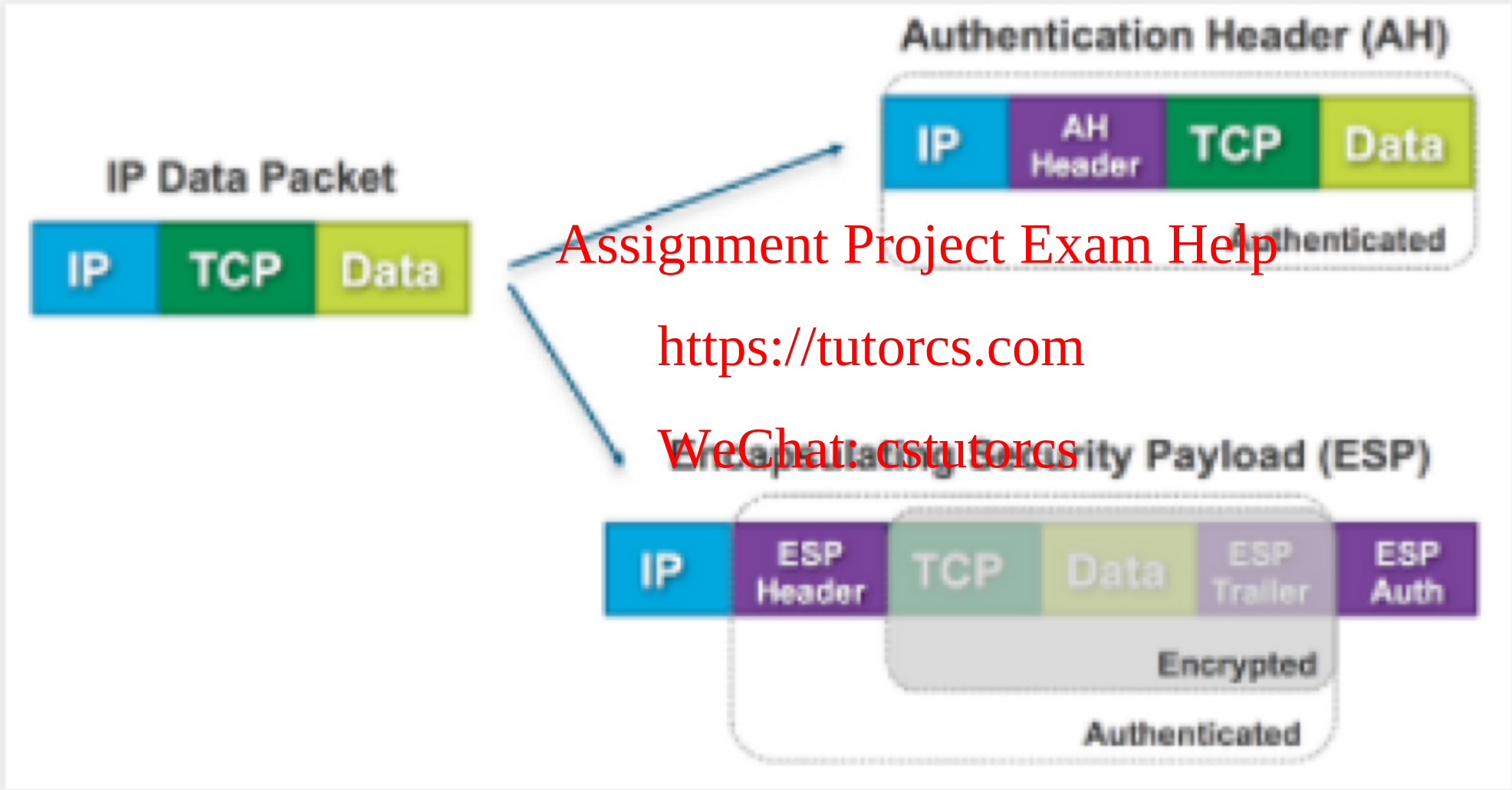
Security Protocols

- Two IPSec Methods:
  - **AH (Authentication Header):**
    - INT, AUTH, no CONF, anti-replay
  - **ESP (Encapsulating Security Payload):**
    - INT, AUTH, CONF, anti-replay



Source: http://jazier.blogspot.com.au/2015/08/ipsec-vpn-theory.html

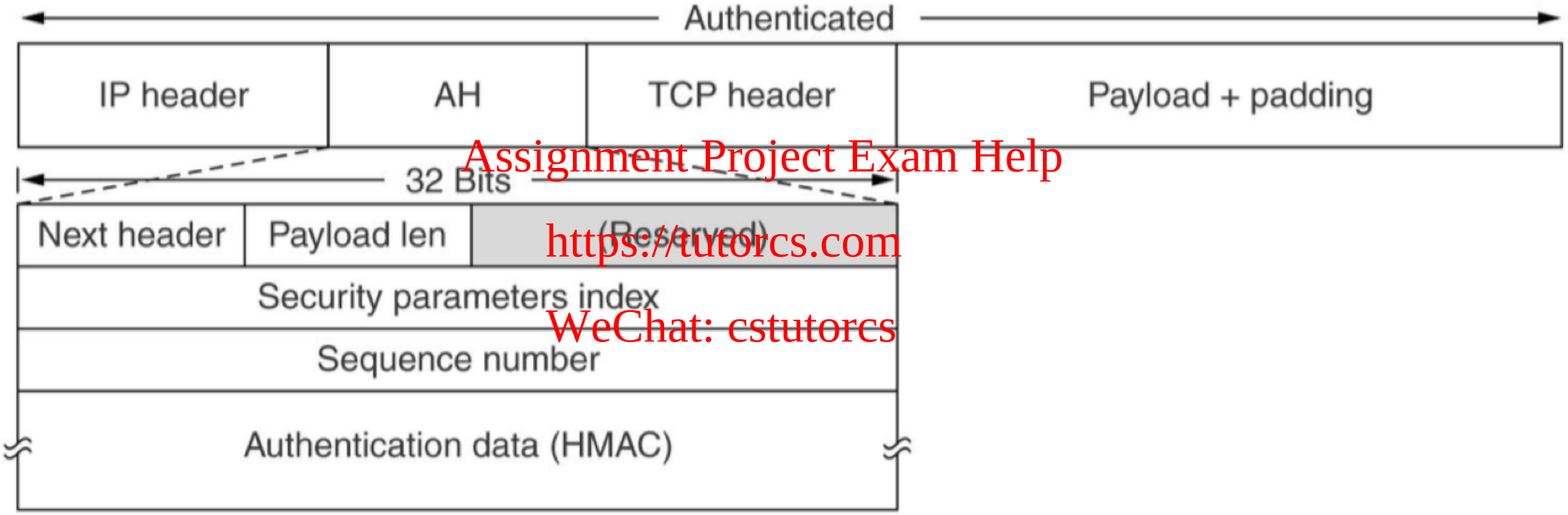
IPsec: Two Methods (AH / ESP)



Source: http://jazier.blogspot.com.au/2015/08/ipsec-vpn-theory.html

IPsec: Header

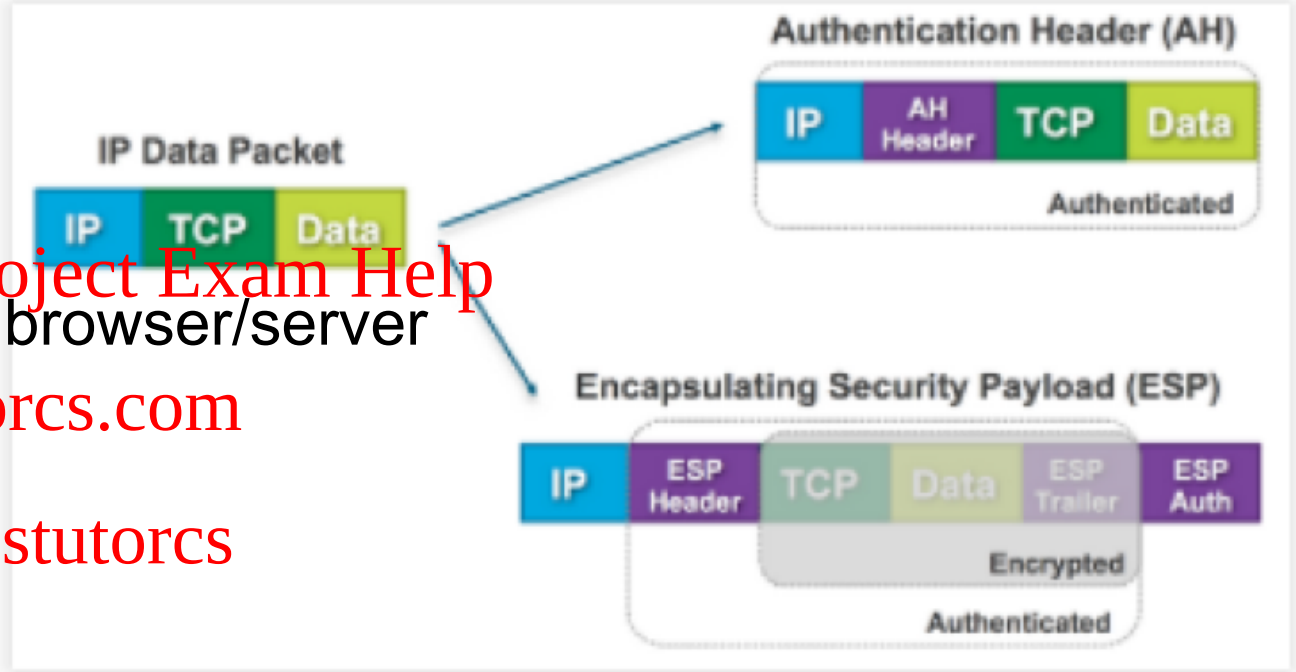
Security Protocols



IPsec: Header

Security Protocols

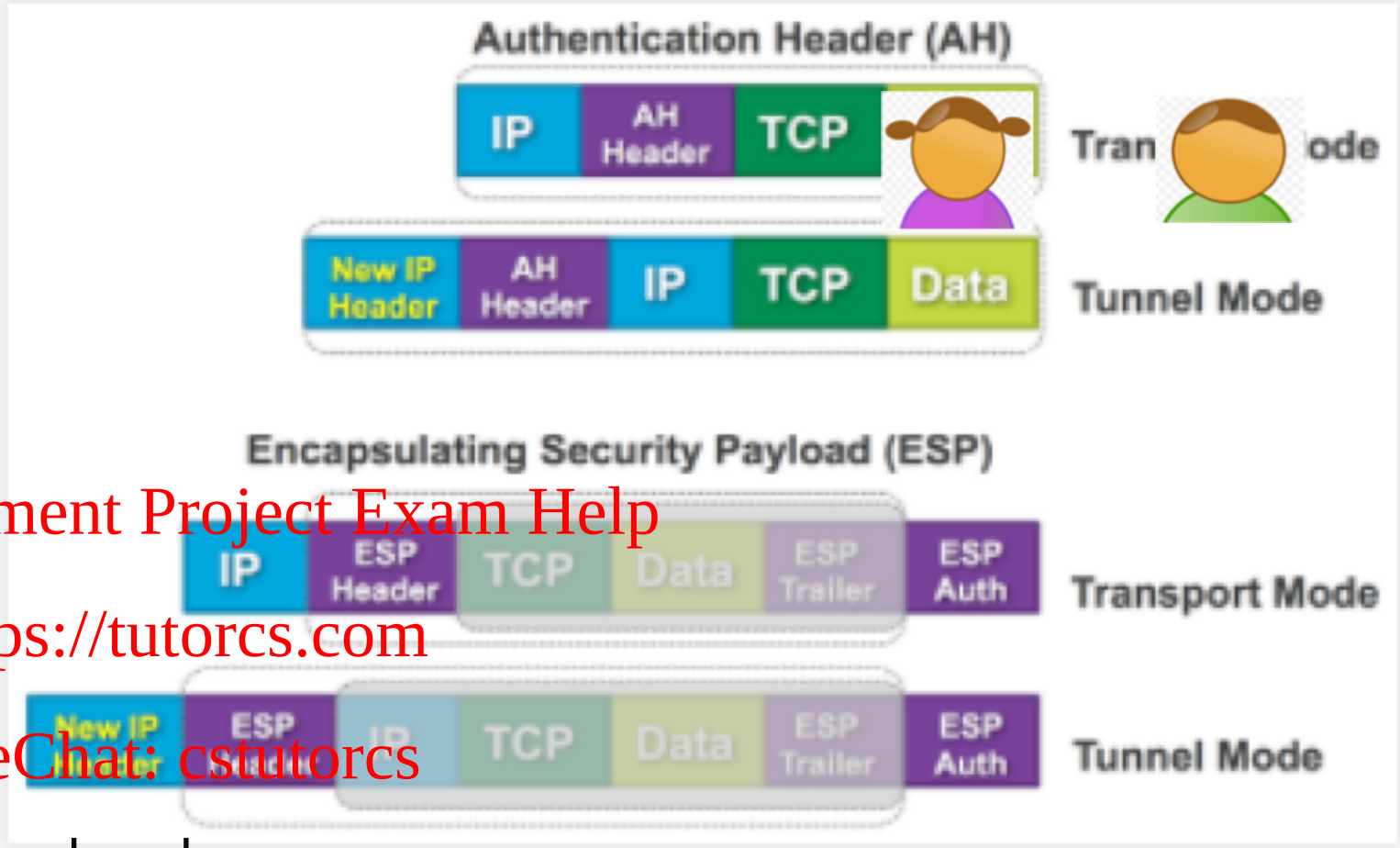
- extra fields for security purposes
  - security parameters index
    - point to entry corresponding to browser/server
      - incl info on shared key
  - sequence number
    - for freshness vs replay attacks
  - MAC output field



Source: http://jazier.blogspot.com.au/2015/08/ipsec-vpn-theory.html

IPsec: Modes

Security Protocols



- **Transport** Mode
  - IP packet **inserted** with IPsec header
- **Tunnel** Mode
  - **original packet preserved** incl original header, new header added/prepended

Source: http://jazier.blogspot.com.au/2015/08/ipsec-vpn-theory.html

MONASH University

Bluetooth: Securing the Personal Area Network (PAN)

Securing the Personal Area Network (PAN)

Security Protocols

- Personal Area Network (PAN)
  - vs LAN / WAN / MAN
- network of personal devices around human's personal space
  - e.g. mobile phone, ear phone, tablet
  - connected via Bluetooth, ZigBee, IrDA

Personal Area Network Security: Bluetooth

Security Protocols

- Bluetooth is a popular PAN technology
  - Apple **AirPods**
  - keyboards, mice, trackpads
  - **fitness** watches
  - **helmet** cameras
  - **wearables**: Jacguard by Google with Levi's / Adidas
    https://youtube.com/watch?v=xwuSGTGYdO0
    https://www.youtube.com/watch?v=qObSFfdfe7I

Security your PAN

Security Protocols

- much hype about internet of things (IoT), of everything (IoE)
- Q: What if your things got hijacked?

- Bluetooth standard
  - latest version 5.2 [2020]
- Bluetooth security
  - Low Energy **Secure Connections** [v4.2]
  - uses ECDH (elliptic curve Diffie-Hellman)
  - supports 4 association models
    - ...

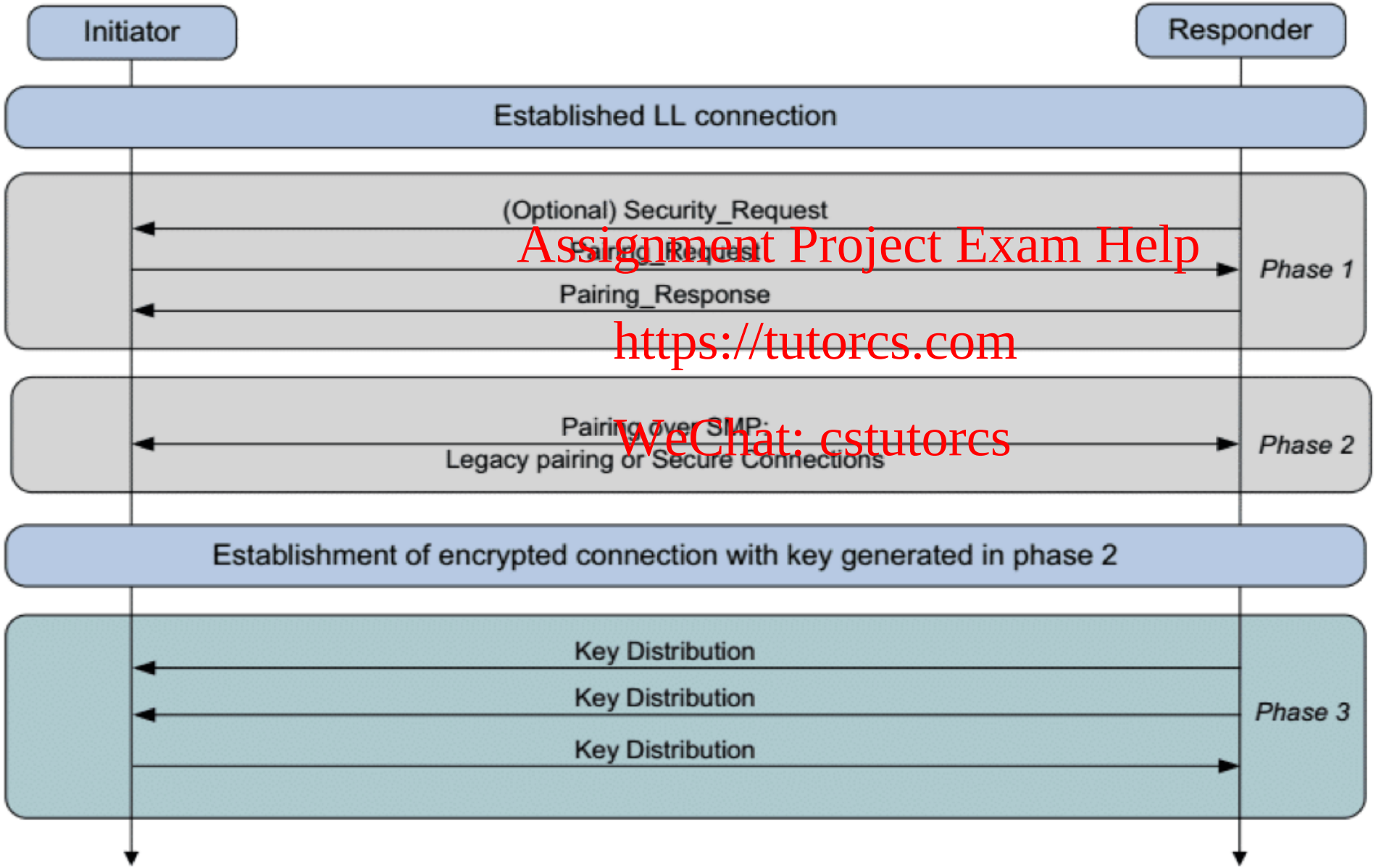
Security your PAN: Bluetooth

Security Protocols

- Bluetooth security
  - LE Secure Connections [v4.2]
  - supports 4 association models
    - Just Works
    - Numeric Comparison
    - Passkey Entry
    - Out of Band

to **pair** two Bluetooth devices securely, i.e. establish a shared key,
based on device I/O capabilities,
for subsequent secure data transfer between
paired Bluetooth devices

Security your PAN: Bluetooth



Bluetooth's Secure Connections Protocol

Security Protocols

- Gist of Bluetooth's Secure Connections [v4.2]
  - mainly Phase 2
    - legacy pairing (Secure Simple Pairing), or
    - Secure Connections
  - novel way of AUTH
    - leverage on **human channel** during pairing, e.g.
      - compare passkey on both devices
      - see passkey on one, type into the other
      - type same passkey into both
      - which option, depends on device I/O capability
    - assumption
      - harder for attacker to attack both the wireless channel and also control the human owner

Q: What kind of attack is prevented by passkey-based AUTH pairing?

Activity (2 mins)

1) Click the latest link in the Zoom chat
2) Add your question response to the Ed forum

Further Reading

Security Protocols

- Chapters 9 & 22 of the textbook: *Computer Security: Principles and Practice" by William Stallings & Lawrie Brown,* Prentice Hall, 2015

- Optional:
  - IETF RFC5246: TLS 1.2 Protocol Specification: *https://tools.ietf.org/html/rfc5246*
  - IETF TLS 1.3 draft specification: *https://tools.ietf.org/html/draft-ietf-tls-tls13-28*
  - IETF RFC7457: Summarizing Known Attacks on Transport Layer Security (TLS) *and Datagram TLS (DTLS): https://tools.ietf.org/html/rfc7457*
  - Bluetooth Blog: *https://bluetooth.com*