Ασφάλεια Δικτύων και Επικοινωνιών

(Network and Communication Security)

4. Ζητήματα Ασφάλειας στο Επίπεδο Μεταφοράς

5. Το πρωτόκολλο SSL/TLS

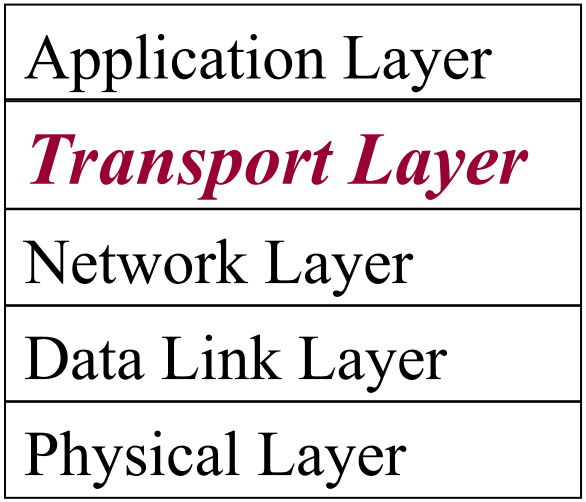
6. Εφαρμογή SSL/TLS σε apache web server με τη χρήση του openssl

Αν.Καθ. Παναγιώτης Κοτζανικολάου

ΠΜΣ ΚΥΒΕΡΝΟΑΣΦΑΛΕΙΑ ΚΑΙ ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ

4. Ασφάλεια στο επίπεδο μεταφοράς

Επίπεδο μεταφοράς (transport layer)



- **Επίπεδο Μεταφοράς**. Είναι υπεύθυνο για τη μεταφορά δεδομένων μεταξύ δύο οντοτήτων επιπέδου εφαρμογής.
  - Καθορίζει τους κανόνες βάσει των οποίων εξασφαλίζεται η ορθή λήψη δεδομένων.
  - Στην πλευρά του παραλήπτη, είναι υπεύθυνο για τη προώθηση των εισερχόμενων δεδομένων στη κατάλληλη διεργασία
- Εκτελείται στους «ακραίους» κόμβους (end systems)

Επίπεδο μεταφοράς

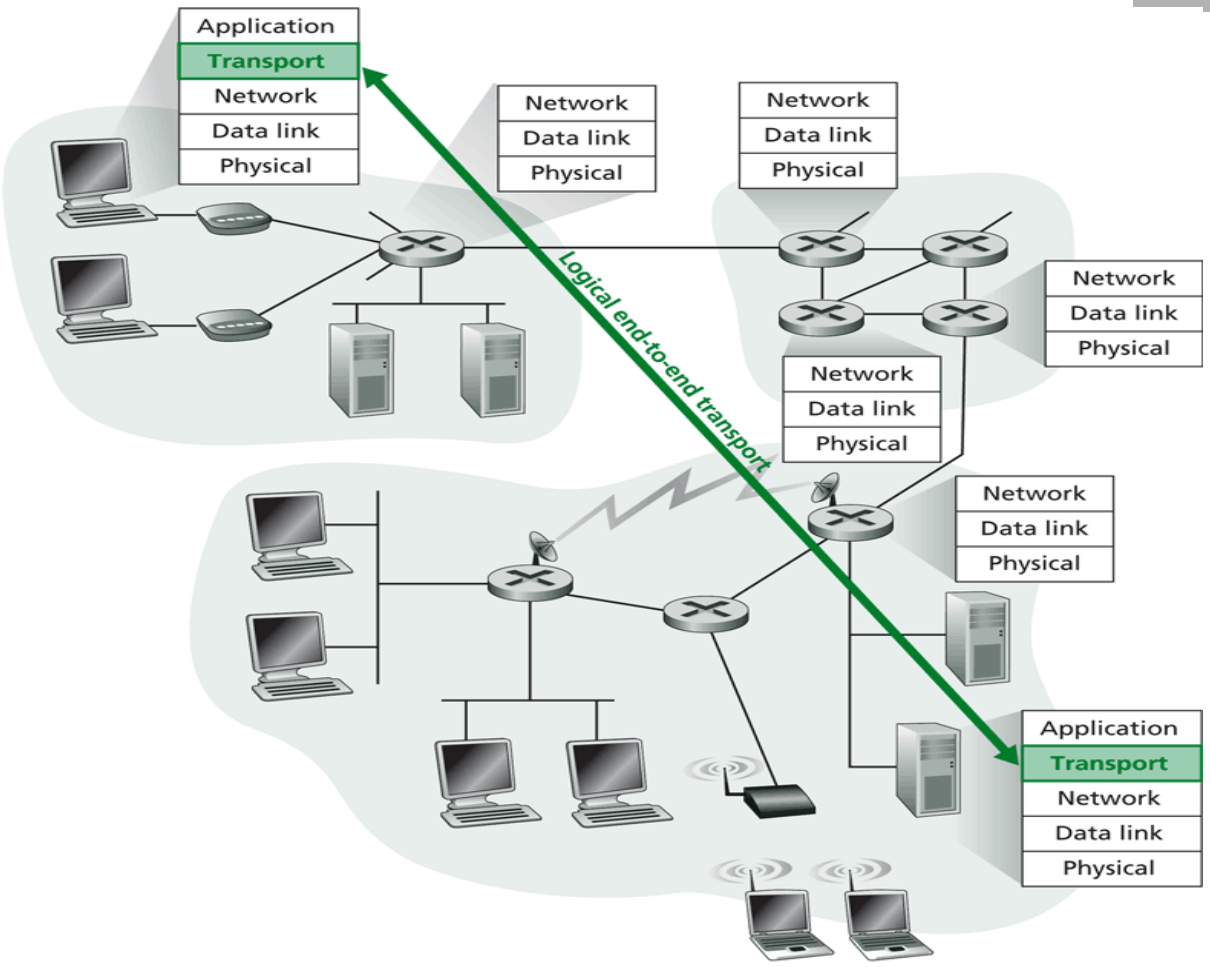
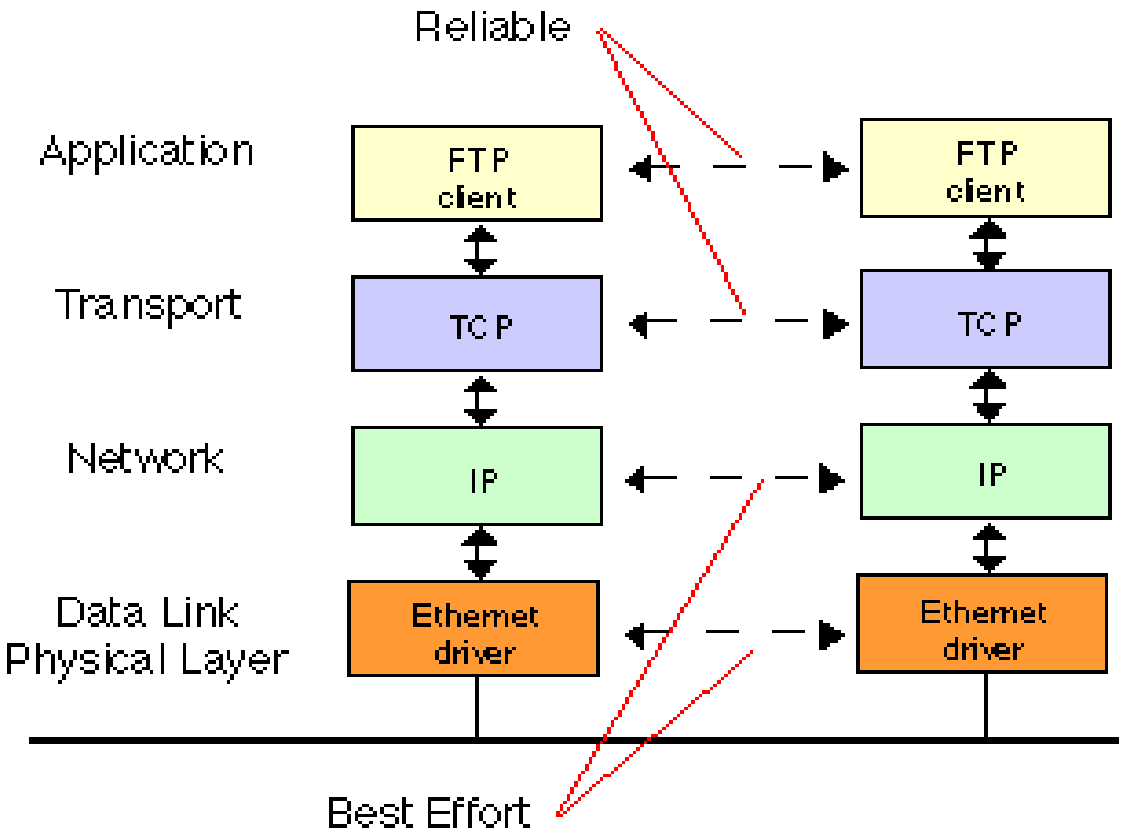


Figure 3.1 ♦ The transport layer provides logical rather than physical communication between application processes.

Αξιόπιστη και μη αξιόπιστη μεταφορά



Υπηρεσία με σύνδεση (Connection-Oriented Service)

- Ο πελάτης και ο απομακρυσμένος εξυπηρετητής ανταλλάσσουν πακέτα ελέγχου της σύνδεσης πριν την αποστολή των πραγματικών δεδομένων.
- Η διαδικασία αυτή είναι γνωστή ως πρωτόκολλο «χειραψίας» (handshaking)
- Μετά την λήξη της χειραψίας, λέμε ότι έχει εγκατασταθεί μία σύνδεση (connection) μεταξύ του πελάτη και του εξυπηρετητή.
- Ο όρος σύνδεση είναι αρκετά «χαλαρός». Μόνο τα ακραία συστήματα έχουν επίγνωση της σύνδεσης και όχι τα ενδιάμεσα (π.χ. Routers, switches κτλ).
- Μία TCP σύνδεση είναι στην πράξη η διατήρηση κάποιων δικτυακών πόρων (buffers, state variables) στα ακραία συστήματα για το σκοπό της μεταξύ τους επικοινωνίας.

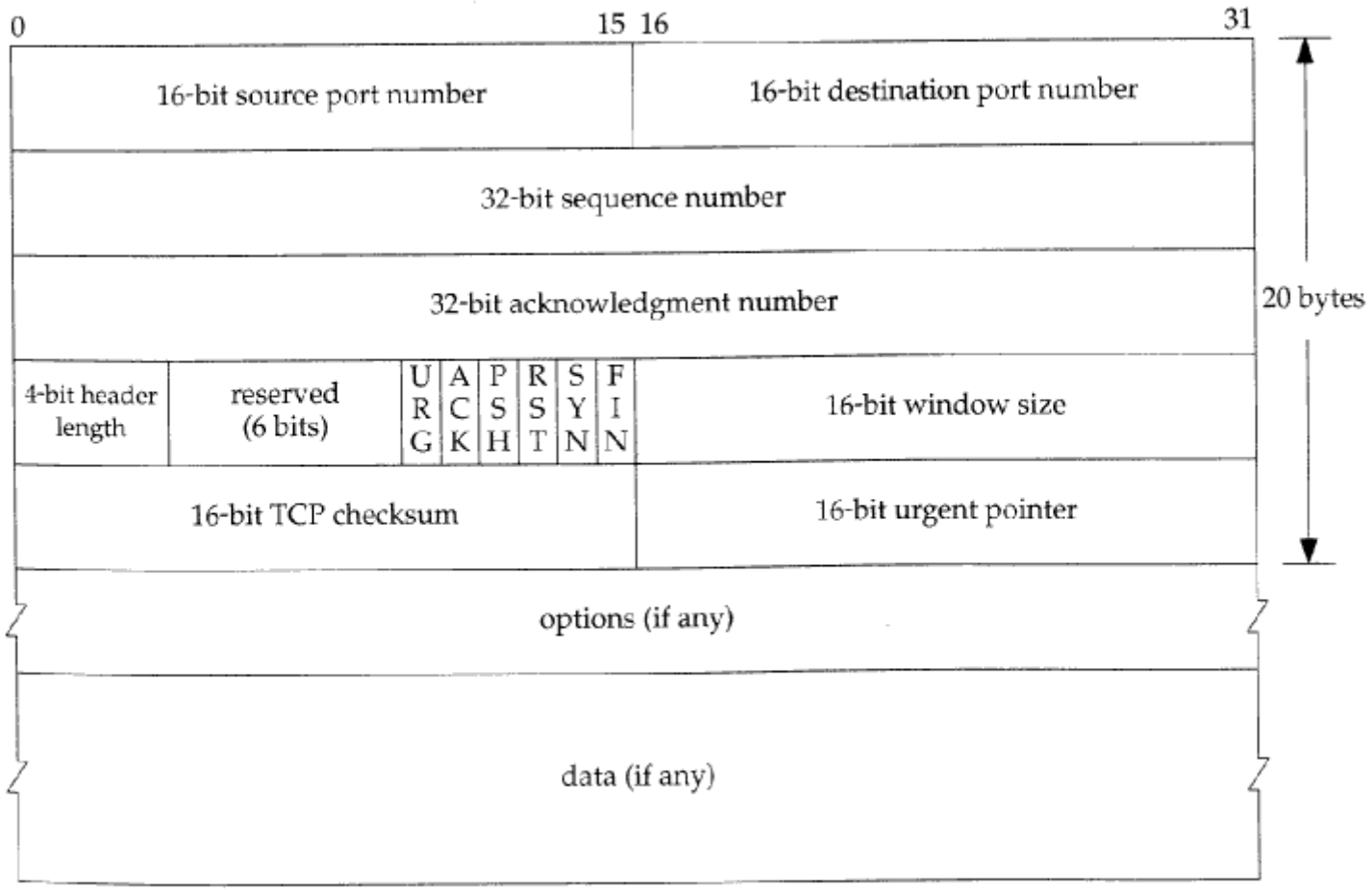
Υπηρεσία χωρίς σύνδεση (Connectionless Service)

- Δεν πραγματοποιείται χειραψία μεταξύ των ακραίων κόμβων (client – server).
- Όταν θέλει ο ένας κόμβος να στείλει δεδομένα στον άλλο, τότε η χρησιμοποιούμενη εφαρμογή απλώς στέλνει τα πακέτα.
- Εφόσον δεν προηγείται η φάση της χειραψίας, η ανταλλαγή δεδομένων είναι ταχύτερη.
- Χωρίς πρωτόκολλο χειραψίας δεν υπάρχει επιβεβαίωση της παραλαβής των πακέτων.
- Η υπηρεσία δεν ελέγχει τη ροή των δεδομένων ή την αποφυγή συμφόρησης.
- Το πρωτόκολλο UDP χρησιμοποιείται στο Internet για πρωτόκολλο μεταφοράς σε υπηρεσίες χωρίς σύνδεση.

Το πρωτόκολλο TCP (Transmission Control Protocol)

- Υπηρεσίες αξιόπιστης παράδοσης (reliable delivery)
  1. Επαληθεύει ότι τα πακέτα έφτασαν στον προορισμό τους
  2. Επαληθεύει ότι έφθασαν χωρίς σφάλματα.
  3. Εξασφαλίζει ότι τα πακέτα φθάνουν στη σωστή εφαρμογή με τη σωστή σειρά
  4. Εξασφαλίζει ότι ο αποστολέας δε δημιουργεί «συμφόρηση» στον παραλήπτη
- Ανίχνευση και διόρθωση λαθών από άκρο σε άκρο (end-to-end)
- Προσανατολισμένο στη σύνδεση (connection oriented)
  - Στο τέλος της χειραψίας το TCP κάθε κόμβου γνωρίζει ότι ο άλλος κόμβος είναι έτοιμος να λάβει δεδομένα

Επικεφαλίδα TCP (1)

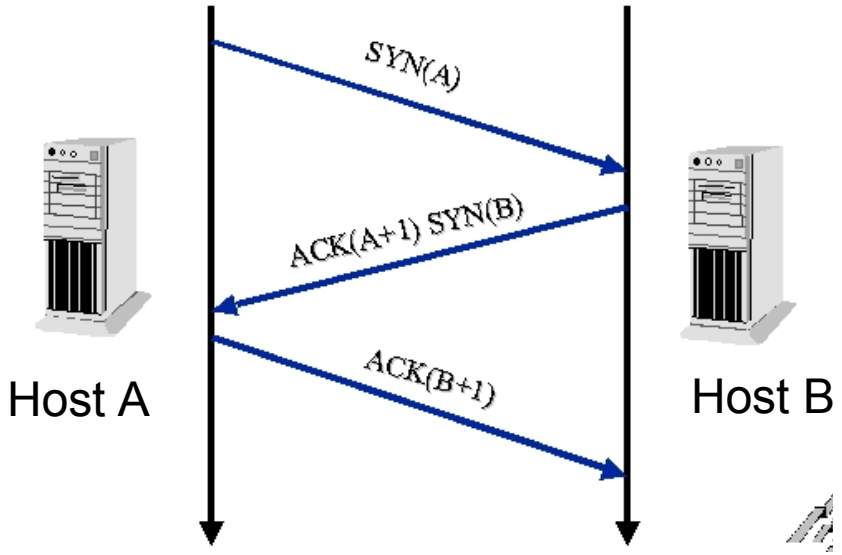


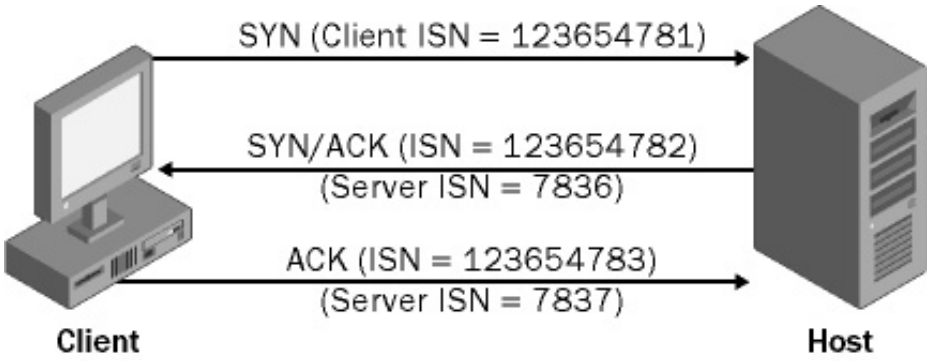
Επικεφαλίδα TCP (2)

- **Source Port .** Ο αριθμός θύρας της πηγής (16-bit).
- **Destination Port.** Ο αριθμός θύρας του προορισμού (16-bit).
- **Sequence Number.** Ο αριθμός ακολουθίας του 1ου byte δεδομένων στο συγκεκριμένο segment. Εάν το bit ελέγχου SYN είναι 1, ο αριθμός ακολουθίας είναι ο αρχικός αριθμός ακολουθίας (N) και το 1ο byte δεδομένων είναι N+1.
- **Acknowledgment Number.** Εάν το bit ελέγχου ACK είναι 1, το πεδίο αυτό περιλαμβάνει την τιμή του επόμενου αριθμού ακολουθίας που αναμένει να λάβει ο παραλήπτης.
- **Data Offset.** Ο αριθμός των 32-bit λέξεων στην TCP επικεφαλίδα. Δείχνει το σημείο όπου ξεκινούν τα δεδομένα.
- **Reserved.** Κρατημένο για μελλοντική χρήση (6-bit). Default τιμή: 000000.
- **URG.** Δείχνει τη σημαντικότητα του πακέτου.
- **ACK.** Bit ελέγχου επιβεβαίωσης.
- **PSH.** Push function.
- **RST.** Επανεκκίνηση της σύνδεσης.
- **SYN.** Συγχρονίζει τους αριθμούς ακολουθίας.
- **FIN.** Τερματισμός των δεδομένων από τον αποστολέα.
- **Window.** Χρησιμοποιείται στα τμήματα ACK. Καθορίζει τον αριθμό των byte δεδομένων που θα αποδεχθεί ο αποστολέας αυτού του τμήματος ως παραλήπτης μίας επικοινωνίας , ξεκινώντας από εκείνο που αναφέρεται στο πεδίο acknowledgment number.
- **Checksum.** Πεδίο ελέγχου της TCP επικεφαλίδας και των TCP δεδομένων . Κατά τον υπολογισμό του, το πεδίο αυτό λαμβάνει την τιμή 0.

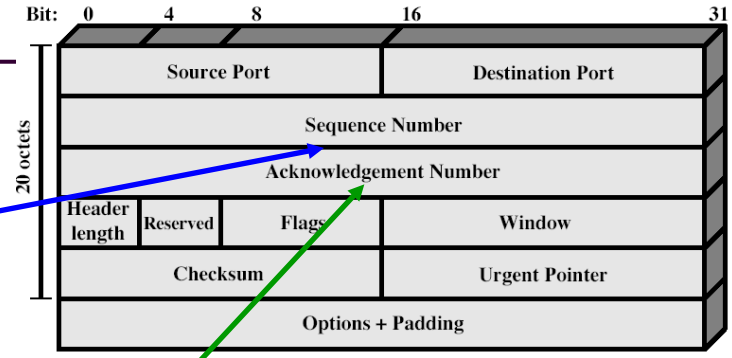
Χειραψία στο TCP

1. Ο Α αρχίζει τη χειραψία στέλνοντας στον Β ένα πακέτο με το πεδίο "Synchronize sequence numbers" SYN = 1. Αυτό λέει στον Β ότι:
   - Ο Α θέλει να ξεκινήσει μία σύνδεση,
   - Ο Α θα χρησιμοποιήσει το συγκεκριμένο Sequence number ως αριθμό εκκίνησης μέτρησης των πακέτων (ώστε να διατηρηθεί η σωστή σειρά των πακέτων).
2. Ο Β απαντά στον *Α* με ένα πακέτο όπου τα bit ACK και SYN είναι 1. Αυτό λέει στον Α ότι ο Β :
   - Έλαβε την αίτηση του Α
   - Ο Β θα χρησιμοποιήσει το συγκεκριμένο sequence number ως αριθμό εκκίνησης μέτρησης των πακέτων.
3. Τελικά ο Α στέλνει στον Β ένα πακέτο που επιβεβαιώνει τη λήψη του προηγούμενου από τον Β και ξεκινά την αποστολή των δεδομένων





Αρίθμηση πακέτων στο TCP

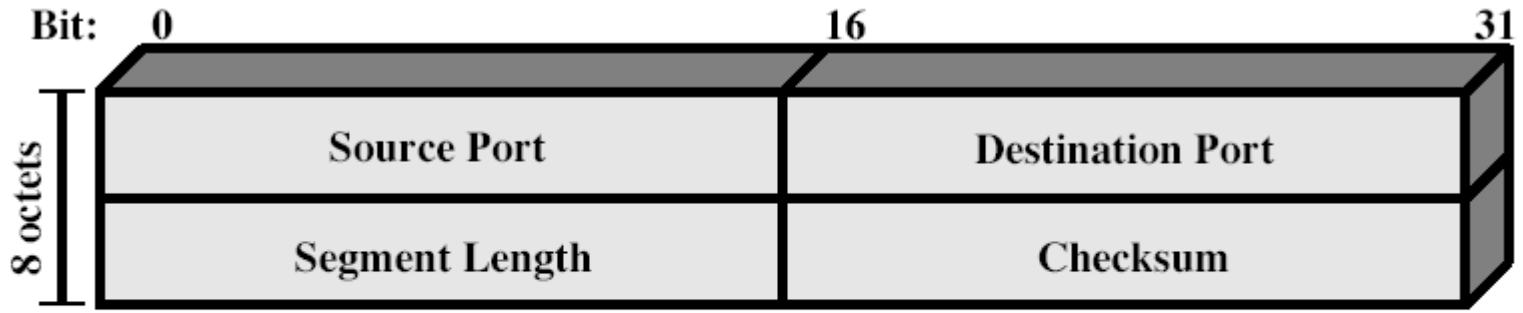


- Σε κάθε πακέτο δεδομένων (segment) εισάγεται **Ακολουθίας** (Sequence Number).
  - Οι αριθμοί ακολουθίας εξασφαλίζουν τη «σωστή σειρά» των πακέτων
- Κάθε πακέτο επιβεβαίωσης περιλαμβάνει έναν **Αριθμό Επιβεβαίωσης** (Acknowledgement Number). Ο αριθμός αυτός
  1. Επιβεβαιώνει τα πακέτα που έχουν ληφθεί σωστά έως τη στιγμή της επιβεβαίωσης
  2. Αναφέρει τον Αριθμό Ακολουθίας του επόμενου πακέτου που αναμένεται
- Συνδέσεις full-duplex
  - Δύο εφαρμογές μπορούν να στέλνουν δεδομένα ταυτόχρονα η μία στην άλλη

Το πρωτόκολλο UDP (User Datagram Protocol)

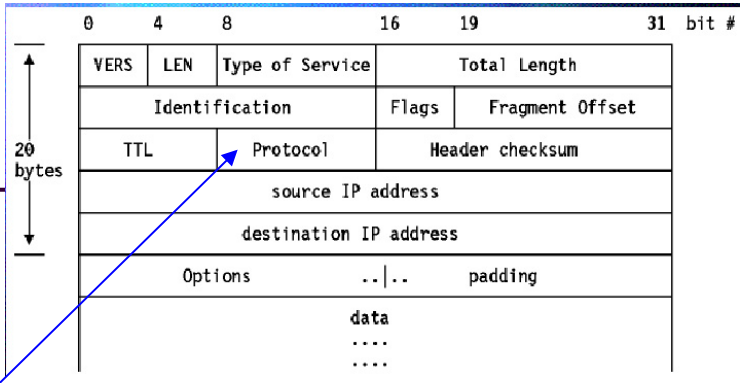
- Μη αξιόπιστη παράδοση
- Χωρίς διόρθωση λαθών
- Χωρίς προσανατολισμό στη σύνδεση (connectionless)
- Κατάλληλο για μεταφορά:
  - Μικρής ποσότητας δεδομένων
  - Εάν η επιβάρυνση για τη διατήρηση της σύνδεσης είναι μεγαλύτερη από το κόστος επαναμετάδοσης.

Επικεφαλίδα UDP

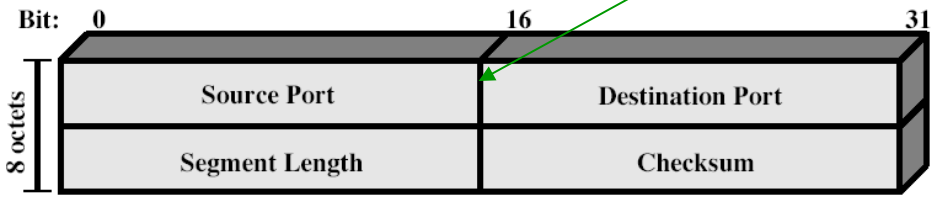


(b) UDP Header

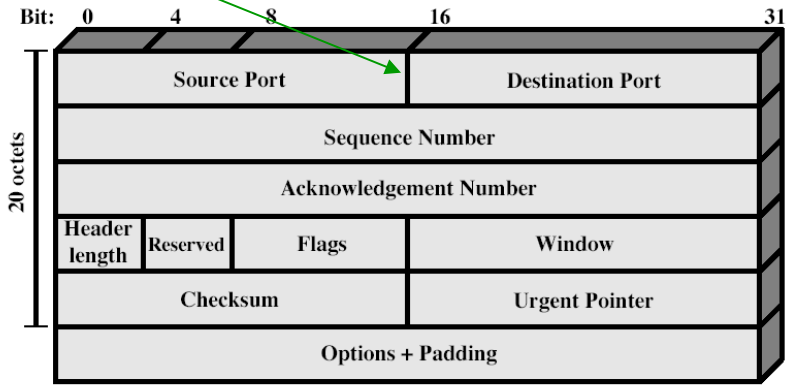
Πρωτόκολλα (Protocols), Θύρες (Ports) και Sockets



- Δύο πρωτόκολλα μεταφοράς (TCP & UDP), πολλές εφαρμογές!
  - Πως βρίσκουν το δρόμο τους τα δεδομένα στον σωρό (stack) των πρωτοκόλλων;
- Το IP χρησιμοποιεί αριθμούς πρωτοκόλλων (protocol numbers) ώστε να καθορίσει το σωστό πρωτόκολλο μεταφοράς
- Τα TCP και UDP χρησιμοποιούν αριθμούς θυρών (port numbers) ώστε να καθορίσουν τη σωστή εφαρμογή



(b) UDP Header



(a) TCP Header

Θύρες (Ports) και Sockets

- Το TCP είναι υπεύθυνο για την πολυπλεξία και κατανομή της εισερχόμενης πληροφορίας στα αντίστοιχα προγράμματα.
- Ο τελικός παραλήπτης των μηνυμάτων κάθε κόμβου είναι ένα 16-bit αφηρημένο σημείου προορισμού – θύρα πρωτοκόλλου – port number)
- Η εκχώρηση θυρών ελέγχεται από την IANA
  - Δεσμευμένες (Reserved) / 0-1024. δημοφιλείς (well-known) εφαρμογές
    - Π.χ. web, mail, chat, ftp, telnet,… (http://www.iana.org/assignments/port-numbers)
  - Καταχωρημένες (Registered) / 1024-49151
  - Ιδιωτικές (Private) / 49152-65535
- **SOCKET = Διεύθυνση IP & port number**
  - Προσδιορίζει μοναδικά μια δικτυακή εφαρμογή σε ολόκληρο το Internet

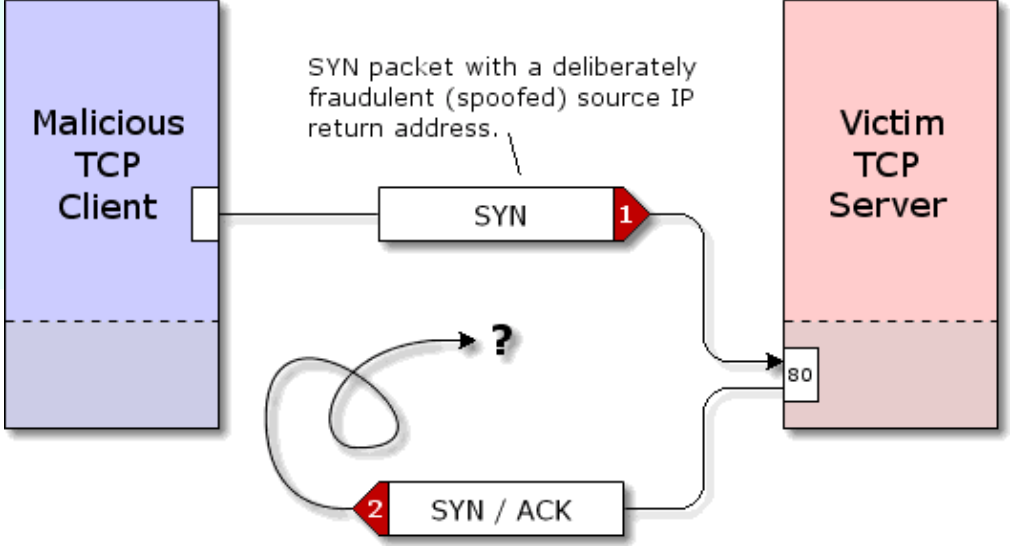
Επιθέσεις στο πρωτόκολλο TCP

TCP Hijacking

Κατάληψη της συνόδου (session) μεταξύ δύο κόμβων.

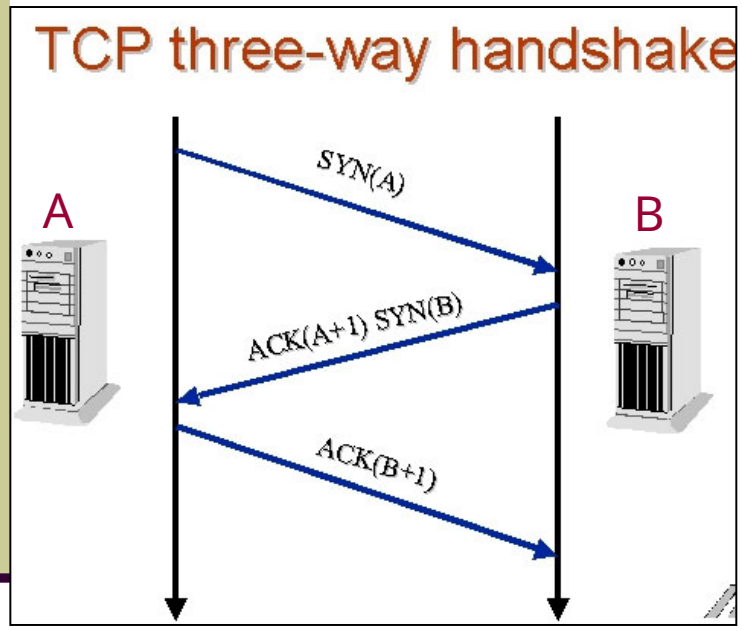
1. Στην αρχή της σύνδεσης:
   - Πριν την ολοκλήρωση της (όποιας) αυθεντικοποίησης
2. Κατά τη διάρκεια της σύνδεσης – Επιθέσεις Man-In-the-Middle (MITM):
   - Ο επιτιθέμενος περιμένει να εγκατασταθεί η σύνδεση μεταξύ του Client και του Server
   - Ο επιτιθέμενος θέτει εκτός λειτουργίας τον client (DoS attack)
   - Ο επιτιθέμενος προσποιείται στον Server ότι είναι ο πραγματικός client.

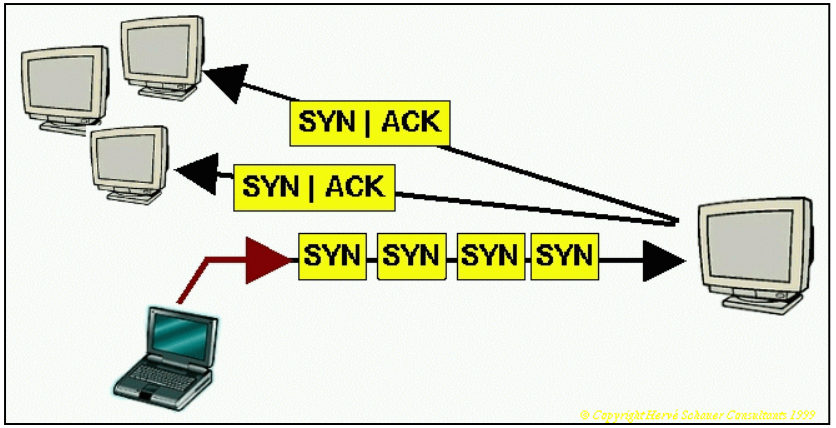
Ασφάλεια σε Δίκτυα TCP/IP- Άρνηση Εξυπηρέτησης



- TCP (SYN) flooding
  - Ο Α στέλνει στον Β συνεχόμενα πακέτα “hello” (TCP SYN))
  - Η διεύθυνση IP αποστολέα στα πακέτα που στέλνει ο Α είναι πλαστή (IP spoofing) – ανύπαρκτη (unreachable)
  - Ο Β απαντά σύμφωνα με το πρωτόκολλο & περιμένει το τρίτο βήμα
    - Δεσμεύοντας έναν χώρο στη μνήμη για την εκκρεμή σύνδεση
    - Χρόνος πριν εκκαθαριστούν οι εκκρεμείς συνδέσεις: 75 sec έως 23 min. (ανάλογα με την υλοποίηση του TCP/IP)
  - Επίθεση: αποστολή εκκρεμών SYN στη συγκεκριμένη θύρα με ρυθμό μεγαλύτερο από το ρυθμό εκκαθάρισης !

SYN FLOODING





Η χειραψία (handshake) στο πρωτόκολλο TCP

Η επίθεση SYN Flooding

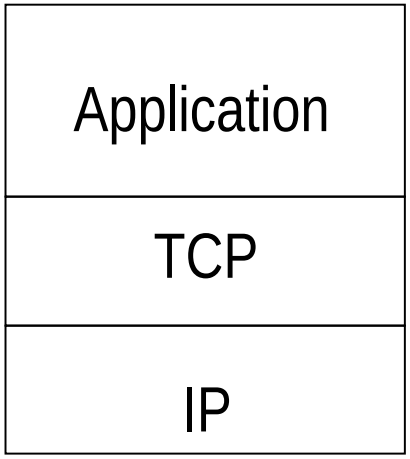
Τα Πρωτόκολλα **Secure Socket Layer (SSL)** και **Transmission Layer Security (TLS)**

5. Το πρωτόκολλο SSL/TLS

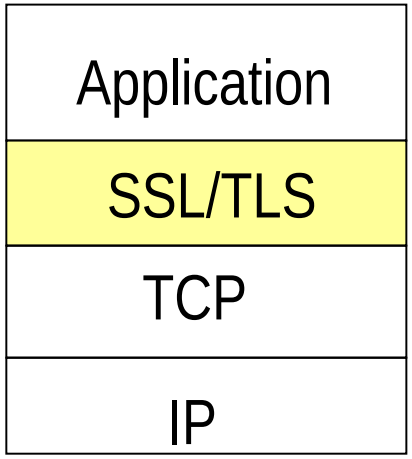
SSL / TLS protocol suites

- Το SSL (v1.0) σχεδιάστηκε αρχικά τη Netscape το 1993.
  - SSL v2.0 (1995)
  - SSL v3.0 (1996). Δημοσιεύτηκε από τον IETF (RFC 6101)
  - TLS v1.0 (1999), RFC 2246.
  - TLS v1.1 (2006), RFC 4346.
  - TLS v1.2 (2008), RFC 4346.
  - TLS v1.3 (2018), RFC 8446.
- Από το 2011, το RFC 6176 ακυρώνει όλες τις προηγούμενες εκδόσεις και δεν επιτρέπει backward compatibility.
- Υποστηρίζεται από τους γνωστούς web browsers / web servers
- Υπηρεσίες που παρέχει:
  - Εμπιστευτικότητα κατά τη μεταφορά (Confidentiality).
  - Ακεραιότητα δεδομένων (Integrity).
  - Αυθεντικοποίηση προορισμού και (προαιρετικά) πηγής (Authentication).

TLS και TCP/IP



Κανονική εφαρμογή



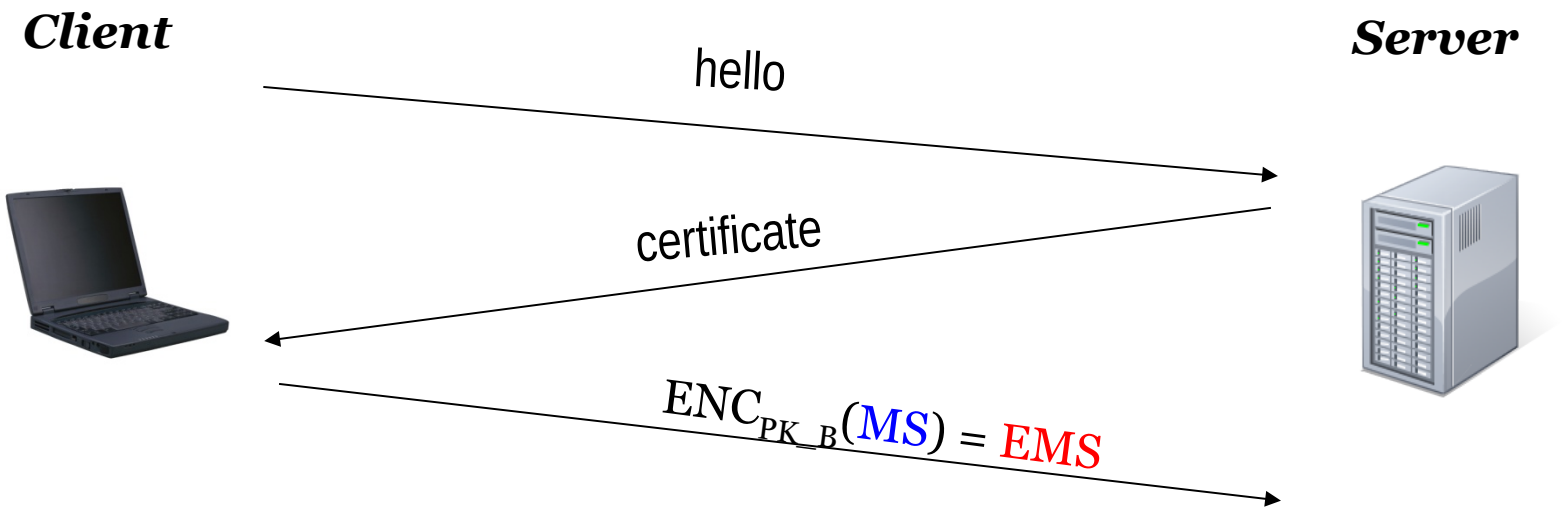
Εφαρμογή με χρήση SSL/TLS

- Το SSL παρέχει προγραμματιστικές διεπαφές (application programming interface – API) στις εφαρμογές
- Διαθέσιμες βιβλιοθήκες/κλάσεις για SSL σε C και Java

Βασικές λειτουργίες

- Χειραψία (Handshake): Η Alice και ο Bob χρησιμοποιούν τα πιστοποιητικά τους και τα αντίστοιχα ιδιωτικά κλειδιά τους για αμοιβαία αυθεντικοποίηση και για ανταλλαγή κοινού μυστικού κλειδιού
- Παραγωγή κλειδιών (Key Derivation): Η Alice και ο Bob χρησιμοποιούν το κοινό κλειδί για να δημιουργήσουν ένα σύνολο από κλειδιά για όλη την επικοινωνία
- Μεταφορά δεδομένων (Data Transfer): Τα δεδομένα που θα μεταφερθούν χωρίζονται σε μία σειρά εγγραφών (data records)
- Τερματισμός σύνδεσης (Connection Closure): Χρησιμοποιούνται ειδικά μηνύματα για τον ασφαλή τερματισμό της σύνδεσης

Ένα απλό πρωτόκολλο χειραψίας



MS = master secret key

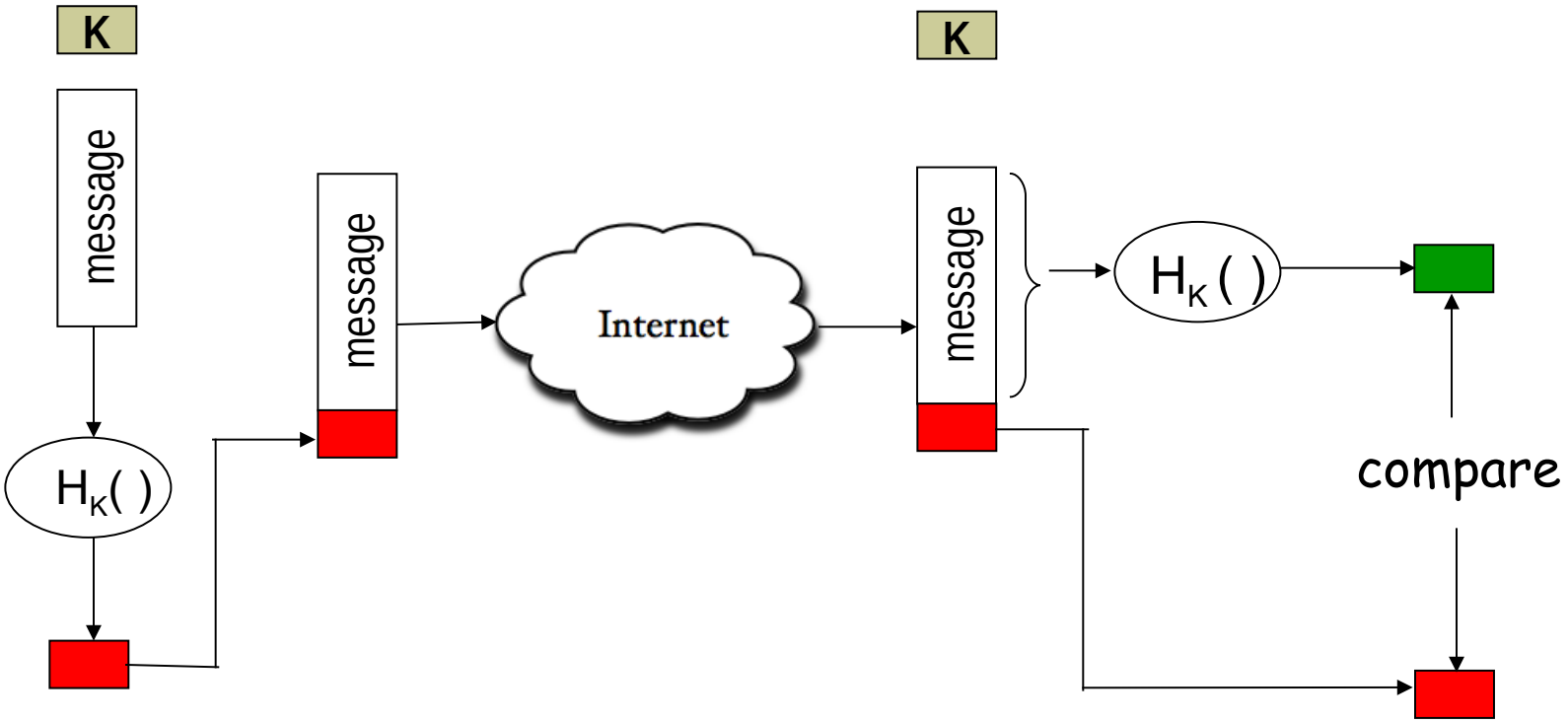
EMS = encrypted master secret key

Ένα απλό παράδειγμα παραγωγής κλειδιών

- Θεωρείται μη ασφαλής η χρήση του ίδιου κλειδιού για πολλαπλές κρυπτογραφικές πράξεις
  - Χρήση διαφορετικού κλειδιού για έλεγχο ακεραιότητας μηνύματος (message authentication code – MAC) και για κρυπτογράφηση
- Από το Master Secret παράγονται τέσσερα κλειδιά:
  - K_c = κλειδί κρυπτογράφησης των δεδομένων που στέλνει ο client προς τον server
  - M_c = κλειδί ελέγχου ακεραιότητας (MAC key) των δεδομένων που στέλνει ο client προς τον server
  - K_s = κλειδί κρυπτογράφησης των δεδομένων που στέλνει ο server προς τον client
  - M_s = κλειδί ελέγχου ακεραιότητας (MAC key) των δεδομένων που στέλνει ο server προς τον client

- Τα κλειδιά παράγονται με τη βοήθεια μίας συνάρτησης παραγωγής κλειδιών (key derivation function – KDF)
  - Λαμβάνει στην είσοδο το διαμοιραζόμενο μυστικό κλειδί (Master Secret) και κάποια επιπλέον τυχαία δεδομένα και παράγει στην έξοδο τα κλειδιά

Έλεγχος ακεραιότητας μηνύματος με τη χρήση Message Autentication Code (MAC)



- Έστω K το μυστικό κλειδί
- Ο αλγόριθμος HMAC είναι η προτυποποιημένη (standarized) εκδοχή του αλγορίθμου MAC
- Το SSL χρησιμοποιεί μία παραλλαγή του αλγορίθμου HMAC
- Το TLS χρησιμοποιεί τον αλγόριθμο HMAC

Εγγραφές δεδομένων (Data Records)

- Κρυπτογράφηση των δεδομένων σε σταθερή ροή, όπως εγγράφονται στο TCP
  - Πού θα τοποθετηθεί το MAC; Αν πάει στο τέλος, τότε ο έλεγχος ακεραιότητας δεν μπορεί να ολοκληρωθεί μέχρι να παραληφθούν όλα τα δεδομένα
- Διάσπαση της ροής των δεδομένων σε μία ακολουθία εγγραφών (data records):
  - Κάθε εγγραφή έχει το δικό της MAC
  - Ο παραλήπτης μπορεί να ελέγξει κάθε εγγραφή δεδομένων ξεχωριστά, αμέσως μετά τη λήψη της
- Πρόβλημα: σε κάθε εγγραφή ο παραλήπτης πρέπει να μπορεί να διαχωρίσει το MAC από τα πραγματικά δεδομένα
  - Χρήση μεταβλητής για το μήκος κάθε εγγραφής

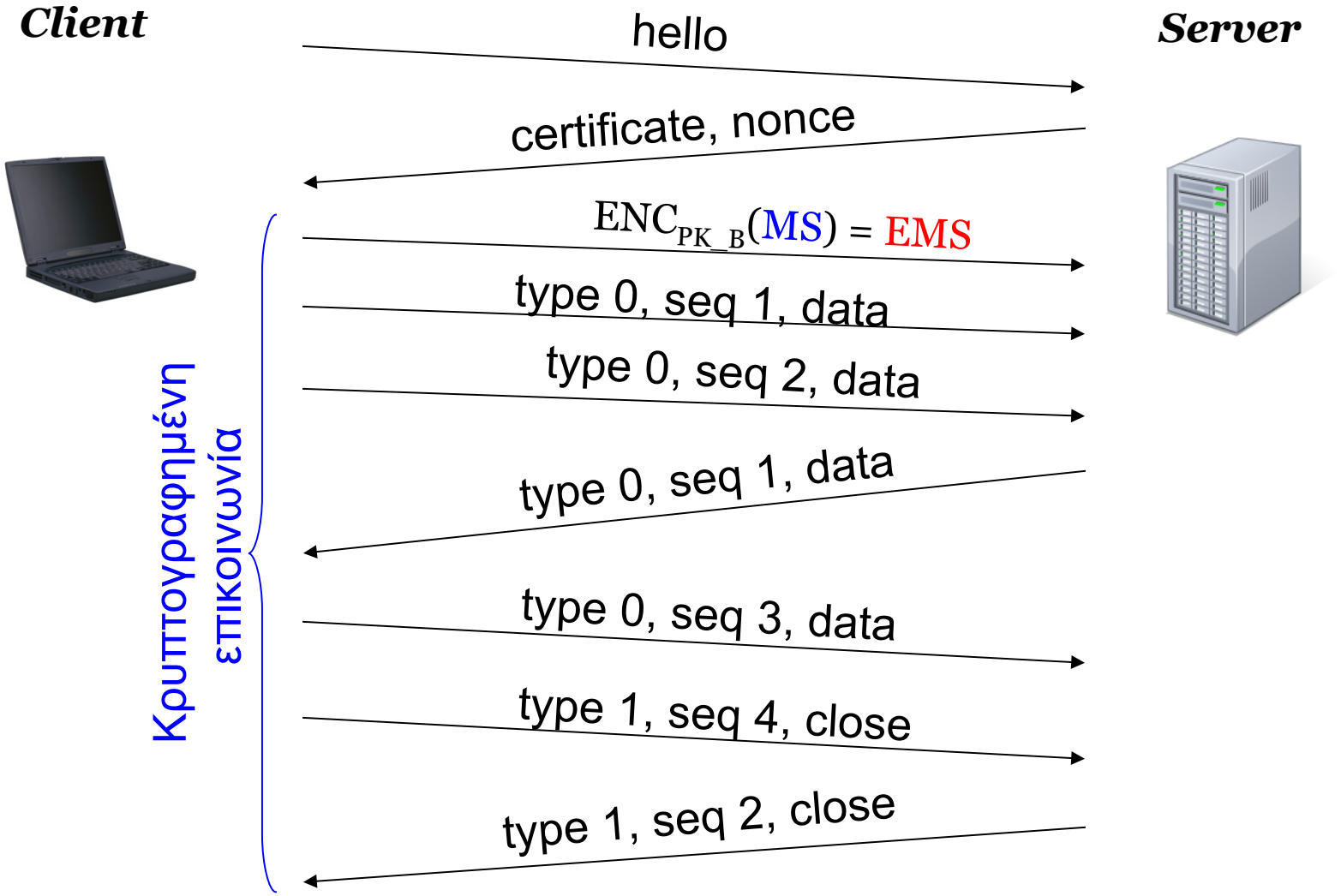
Χρήση αριθμών ακολουθίας (Sequence Numbers)

- Επίθεση επανάληψης (replay attack) : Ένας επιτιθέμενος μπορεί να αντιγράψει και να επαναστείλει μία ή περισσότερες εγγραφές δεδομένων ή/και να αλλάξει τη σειρά τους
- Λύση: Χρήση αριθμών ακολουθίας (sequence numbers) μέσα στο MAC
  - $MAC = H_K(K \,||\, seq\_no \,||\, data)$
  - **Προσοχή**: δεν υπάρχει πεδίο sequence number
- Ο επιτιθέμενος μπορεί ακόμα να επαναστείλει όλες τις εγγραφές δεδομένων
- Λύση: χρήση τυχαίων αριθμών (random nonces)

Έλεγχος πληροφορίας

- Truncation attack:
  - Ο επιτιθέμενος αλλοιώνει την TCP σύνδεση και τερματίζει τη σύνδεση (socket)
  - Η μία ή και οι δύο πλευρές θεωρούν ότι υπάρχουν λιγότερα από τα πραγματικά δεδομένα
- Λύση: χρήση πεδίου record type
  - type 0, για αποστολή δεδομένων
  - type 1, για τερματισμό
- MAC = H_K(K || seq_no || type||data)

Σύνοψη απλοποιημένου πρωτοκόλλου



Προβλήματα απλού παραδείγματος

- Μήκος πεδίων
- Πρωτόκολλα κρυπτογράφησης
- Διαπραγμάτευση μεταξύ client και server
  - Επιτρέπει στον client και το server να υποστηρίζουν διάφορους αλγόριθμους κρυπτογράφησης
  - Επιτρέπει στον client και τον server να επιλέξουν συγκεκριμένο αλγόριθμο κρυπτογράφησης πριν τη μεταφορά των δεδομένων

Πακέτο κρυπτογραφίας για το TLS (TLS Cipher Suite)

- Cipher Suite
  - Αλγόριθμος δημόσιου κλειδιού
  - Συμμετρικός κρυπτογραφικός αλγόριθμος
  - Αλγόριθμος ελέγχου ακεραιότητας (MAC)
- Το TLS υποστηρίζει πολλά κρυπτογραφικά πακέτα
- Η χρήση ενός συγκεκριμένου cipher suite από τον client και τον server βασίζεται στη μεταξύ τους διαπραγμάτευση-συμφωνία
  - Ο client προτείνει εναλλακτικά cipher suites
  - Ο server επιλέγει ένα από αυτά

Αλγόριθμοι συμμετρικής κρυπτογράφησης (ανά έκδοση)

Cipher			Protocol version						Status
Type	Algorithm	Nominal strength (bits)	SSL 2.0	SSL 3.0[n 1][n 2][n 3][n 4]	TLS 1.0[n 1][n 3]	TLS 1.1[n 1]	TLS 1.2[n 1]	TLS 1.3	
Block cipher with mode of operation	AES GCM[75][n 5]	256, 128	—	—	—	—	Secure	Secure	Defined for TLS 1.2 in RFCs
	AES CCM[76][n 5]		—	—	—	—	Secure	Secure	
	AES CBC[n 6]		—	Insecure	Depends on mitigations	Depends on mitigations	Depends on mitigations	—	
	Camellia GCM[77][n 5]	256, 128	—	—	—	—	Secure	—	
	Camellia CBC[78][n 6]		—	Insecure	Depends on mitigations	Depends on mitigations	Depends on mitigations	—	
	ARIA GCM[79][n 5]	256, 128	—	—	—	—	Secure	—	
	ARIA CBC[79][n 6]		—	—	Depends on mitigations	Depends on mitigations	Depends on mitigations	—	
	SEED CBC[80][n 6]	128	—	Insecure	Depends on mitigations	Depends on mitigations	Depends on mitigations	—	
	3DES EDE CBC[n 6][n 7]	112[n 8]	Insecure	Insecure	Insecure	Insecure	Insecure	—	
	GOST R 34.12-2015 Magma CTR[74][n 7]	256	—	—	Insecure	Insecure	Insecure	—	Defined in RFC 4357, 9189
	GOST R 34.12-2015 Kuznyechik CTR[74]	256	—	—	—	—	Secure	—	Defined in RFC 9189
	GOST R 34.12-2015 Magma MGM[74][n 5][n 7]	256	—	—	—	—	—	Insecure	Defined in RFC 9367
	GOST R 34.12-2015 Kuznyechik MGM[74][n 5]	256	—	—	—	—	—	Secure	Defined in RFC 9367
	IDEA CBC[n 6][n 7][n 9]	128	Insecure	Insecure	Insecure	Insecure	—	—	Removed from TLS 1.2
	DES CBC[n 6][n 7][n 9]	56	Insecure	Insecure	Insecure	Insecure	—	—	
		40[n 10]	Insecure	Insecure	Insecure	—	—	—	Forbidden in TLS 1.1 and later
	RC2 CBC[n 6][n 7]	40[n 10]	Insecure	Insecure	Insecure	—	—	—	
Stream cipher	ChaCha20-Poly1305[83][n 5]	256	—	—	—	—	Secure	Secure	Defined for TLS 1.2 in RFCs
	RC4[n 11]	128	Insecure	Insecure	Insecure	Insecure	Insecure	—	Prohibited in all versions of TLS by RFC 7465
		40[n 10]	Insecure	Insecure	Insecure	—	—	—	
None	Null[n 12]	-	Insecure	Insecure	Insecure	Insecure	Insecure	—	Defined for TLS 1.2 in RFCs

Αλγόριθμοι ελέγχου ακεραιότητας (ανά έκδοση)

Data integrity

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Status
HMAC-MD5	Yes	Yes	Yes	Yes	Yes	No	Defined for TLS 1.2 in RFCs
HMAC-SHA1	No	Yes	Yes	Yes	Yes	No	
HMAC-SHA256/384	No	No	No	No	Yes	No	
AEAD	No	No	No	No	Yes	Yes	
GOST 28147-89 IMIT[74]	No	No	No	No	Yes	No	Defined for TLS 1.2 in RFC 9189.
GOST R 34.12-2015 AEAD[74]	No	No	No	No	No	Yes	Defined for TLS 1.3 in RFC 9367.

Αλγόριθμοι αυθεντικοποίησης και ανταλλαγής κλειδιού (ανά έκδοση)

Key exchange/agreement and authentication

Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Status
RSA	Yes	Yes	Yes	Yes	Yes	No	Defined for TLS 1.2 in RFCs
DH-RSA	No	Yes	Yes	Yes	Yes	No	
DHE-RSA (forward secrecy)	No	Yes	Yes	Yes	Yes	Yes	
ECDH-RSA	No	No	Yes	Yes	Yes	No	
ECDHE-RSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes	
DH-DSS	No	Yes	Yes	Yes	Yes	No	
DHE-DSS (forward secrecy)	No	Yes	Yes	Yes	Yes	No[72]	
DHE-ECDSA (forward secrecy)	No	No	No	No	No	Yes	
ECDH-ECDSA	No	No	Yes	Yes	Yes	No	
ECDHE-ECDSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes	
DHE-EdDSA (forward secrecy)	No	No	No	No	No	Yes	
ECDH-EdDSA	No	No	Yes	Yes	Yes	No	
ECDHE-EdDSA (forward secrecy)[73]	No	No	Yes	Yes	Yes	Yes	
PSK	No	No	Yes	Yes	Yes	Yes	
PSK-RSA	No	No	Yes	Yes	Yes	No	
DHE-PSK (forward secrecy)	No	No	Yes	Yes	Yes	Yes	
ECDHE-PSK (forward secrecy)	No	No	Yes	Yes	Yes	Yes	
SRP	No	No	Yes	Yes	Yes	No	
SRP-DSS	No	No	Yes	Yes	Yes	No	
SRP-RSA	No	No	Yes	Yes	Yes	No	
Kerberos	No	No	Yes	Yes	Yes	?	
DH-ANON (insecure)	No	Yes	Yes	Yes	Yes	No	
ECDH-ANON (insecure)	No	No	Yes	Yes	Yes	No	
GOST R 34.10-2012[74]	No	No	No	No	Yes	Yes	Defined for TLS 1.2 and for TLS 1.3 in RFC 9189, 9367.

Πρωτόκολλο χειραψίας (handshake protocol) (1)

- Στόχοι πρωτοκόλλου
  1. Αυθεντικοποίηση server (server authentication)
  2. Διαπραγμάτευση για κρυπτογραφικούς αλγορίθμους
  3. Ανταλλαγή/συμφωνία κλειδιών
  4. Αυθεντικοποίηση client (προαιρετικά)

Πρωτόκολλο χειραψίας (handshake protocol) (2)

1. Ο client στέλνει μία λίστα από τους αλγόριθμους κρυπτογράφησης που υποστηρίζει, μαζί με ένα τυχαίο αριθμό μίας χρήσης (client nonce)
2. Ο server επιλέγει από τη λίστα και επιστρέφει:
   - choice + certificate + server nonce
3. Ο client:
   - επαληθεύει το πιστοποιητικό
   - Εξάγει το δημόσιο κλειδί του server
   - Παράγει το pre_master_secret
   - Το κρυπτογραφεί με το δημόσιο κλειδί του server
   - Το στέλνει στο server
4. Οι client και server υπολογίζουν ο καθένας τα κλειδιά κρυπτογράφησης και ελέγχου ακεραιότητας (MAC key) από το pre_master_secret και τα δύο nonces
5. Ο client στέλνει ένα MAC όλων των μηνυμάτων της χειραψίας
6. Ο server στέλνει ένα MAC όλων των μηνυμάτων της χειραψίας

Πρωτόκολλο χειραψίας (handshake protocol) (3)

- Τα τελευταία δύο βήματα προστατεύουν από πιθανή απόπειρα αλλοίωσης
- Ο client προτείνει διάφορους αλγόριθμους κρυπτογράφησης με διαφορετικό επίπεδο ασφάλειας
- Ένας κακόβουλος ενδιάμεσος (Man-in-the middle) θα μπορούσε να διαγράψει τους ισχυρότερους αλγόριθμους από τη λίστα (downgrading attack)
- Αυτό προλαμβάνεται με τα τελευταία 2 βήματα (MAC)

Πρωτόκολλο χειραψίας (handshake protocol) (4)

- Γιατί χρησιμοποιούνται τα random nonces?

- Προστασία από επιθέσεις επανάληψης (replay attacks)
  - Ο επιτιθέμενος υποκλέπτει όλα τα μηνύματα μεταξύ του client και server
  - Ο επιτιθέμενος ξεκινά μία νέα TCP σύνδεση με το server και στέλνει τα ίδια ακριβώς μηνύματα με την ίδια σειρά (και πιθανώς το πιστοποιητικό του client)
  - Ο server νομίζει ότι πρόκειται για δύο ανεξάρτητες συνδέσεις από τον ίδιο client (π.χ. δύο παραγγελίες)

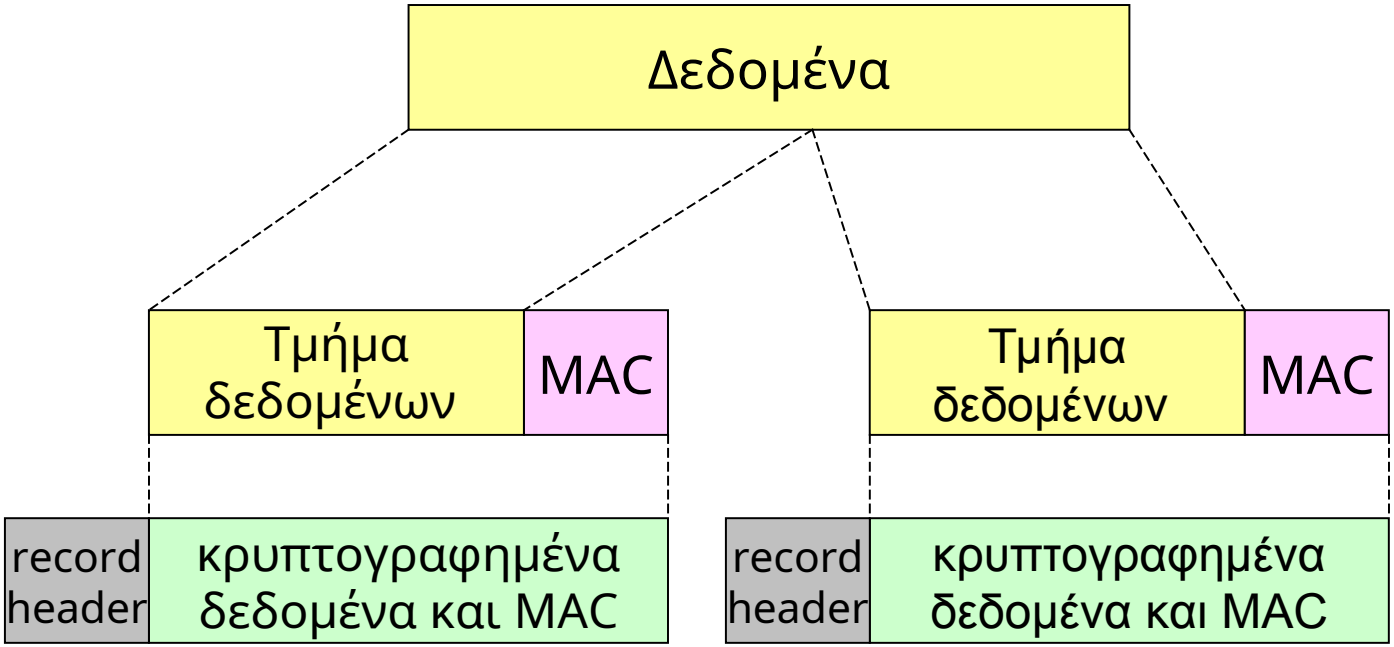
- Λύση: Ο server χρησιμοποιεί διαφορετικό random nonce για κάθε σύνδεση.
  - Το ίδιο και ο client

Τύποι χειραψίας

- Όλα τα μηνύματα χειραψίας της TLS επικεφαλίδας έχουν ένα πεδίο που περιγράφει τον τύπο της χειραψίας (1 byte)

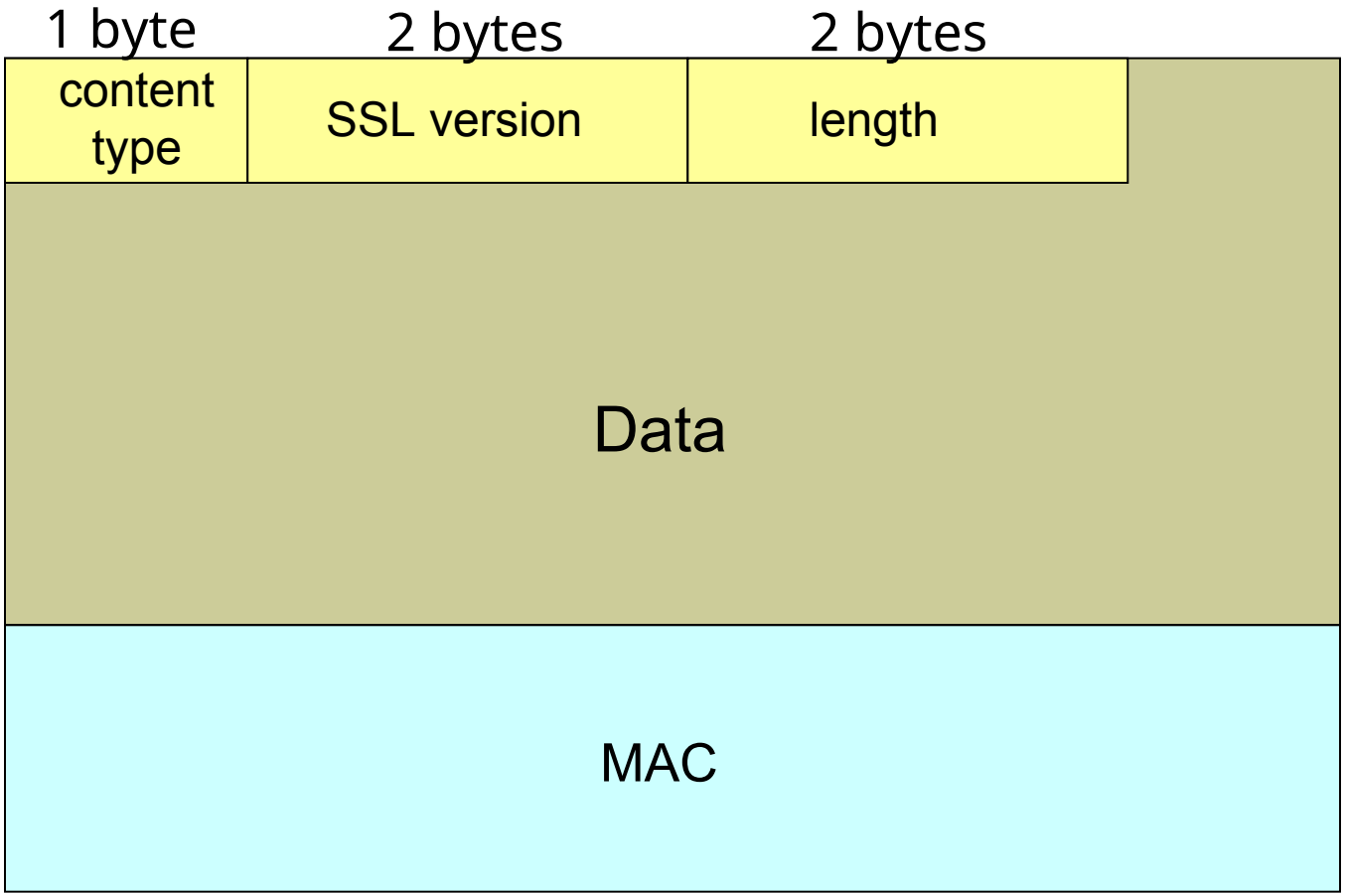
- Πιθανοί τύποι TLS χειραψίας
  - ClientHello
  - ServerHello
  - Certificate
  - ServerKeyExchange
  - CertificateRequest
  - ServerHelloDone
  - CertificateVerify
  - ClientKeyExchange
  - Finished

Πρωτόκολλο εγγραφής TLS (TLS Record Protocol)



Επικεφαλίδα TLS (record header): content type; version; length

Μορφοποίηση TLS Record

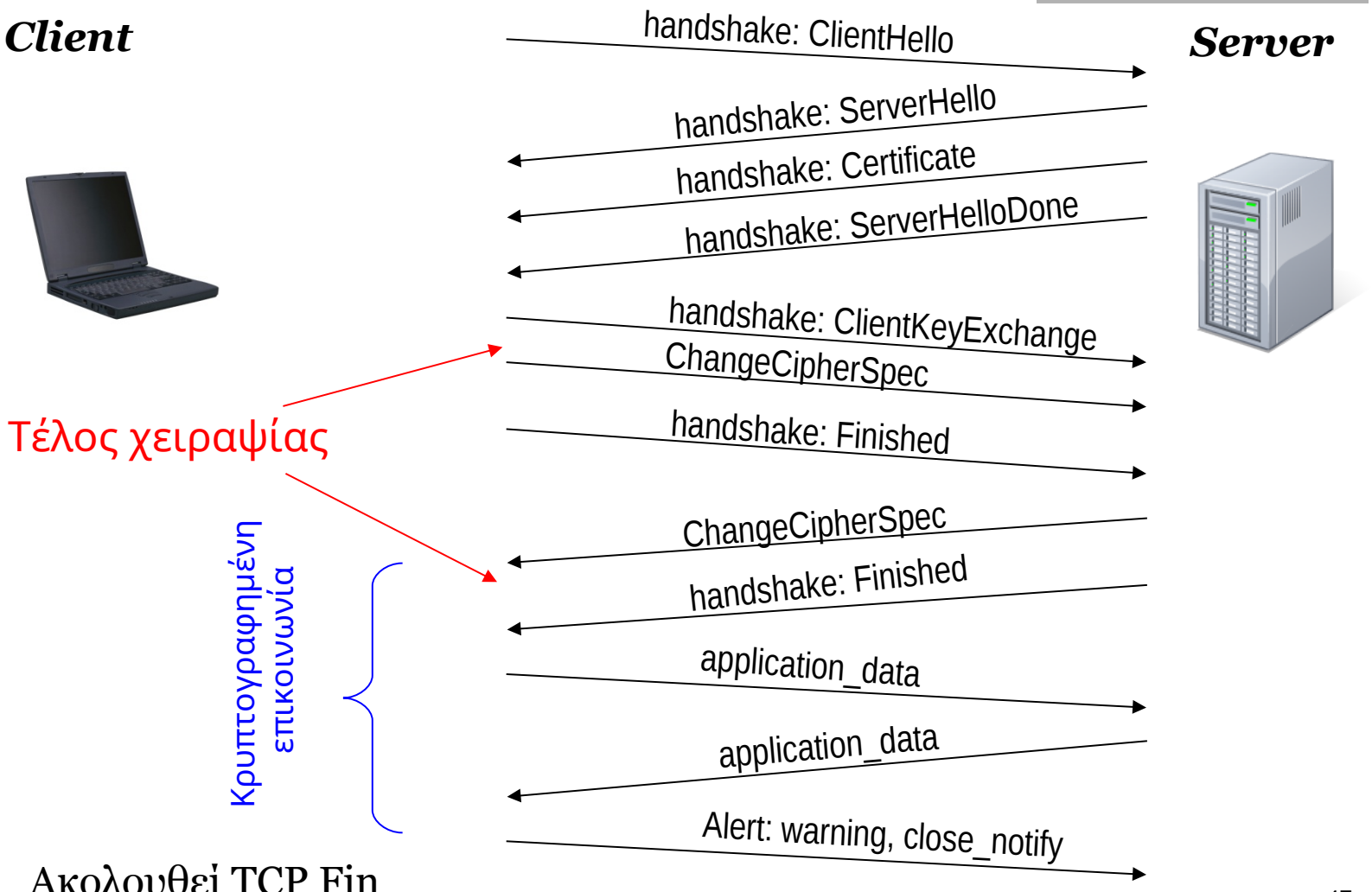


Τα δεδομένα και το MAC είναι κρυπτογραφημένα

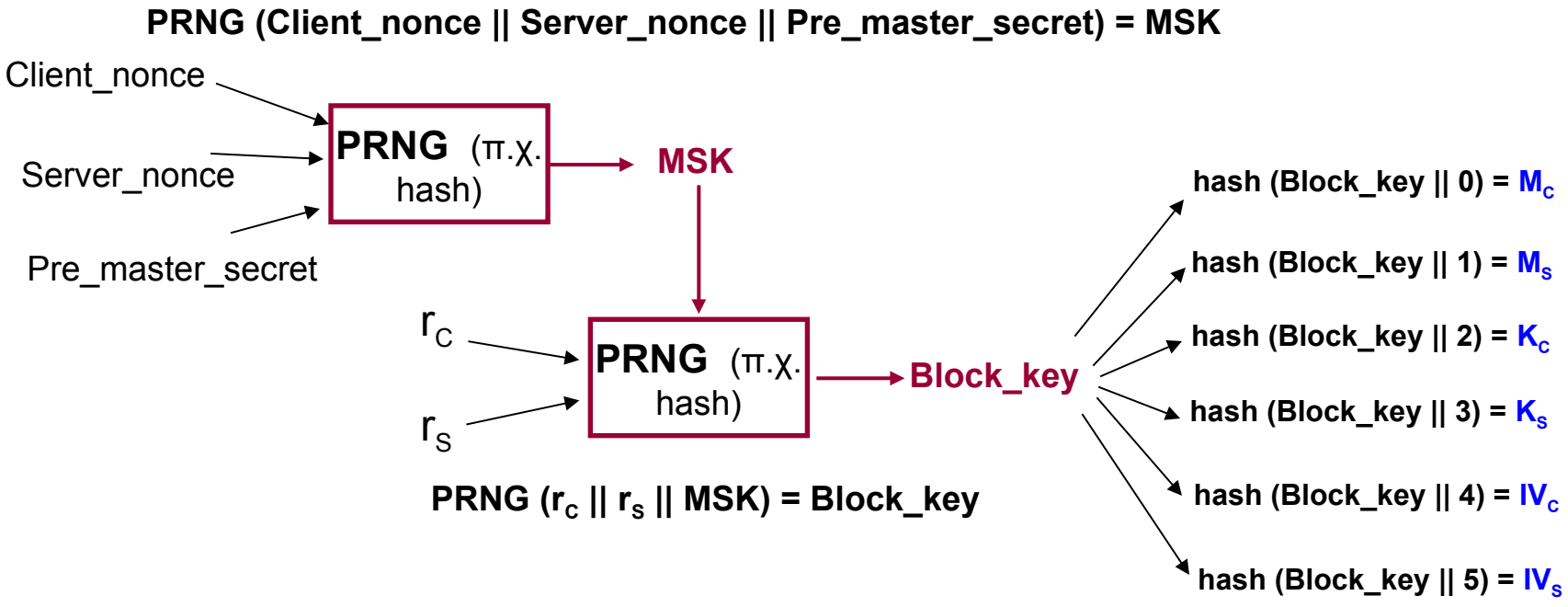
Τύποι περιεχομένου επικεφαλίδας TLS

- application_data (23)
  - Κανονική μετάδοση δεδομένων
- alert (21)
  - Σφάλματα σηματοδοσίας κατά τη χειραψία
- handshake (22)
  - Τα αρχικά μηνύματα χειραψίας μεταφέρονται σε εγγραφές TLS τύπου “handshake”
  - Τα μηνύματα hankshake έχουν επίσης τους δικούς τους υπο-τύπους
- change_cipher_spec (20)
  - Αλλαγή των αλγορίθμων κρυπτογράφησης-ελέγχου ακεραιότητας

Πραγματική σύνδεση TLS



Παραγωγή κλειδιών



Απόδοση του πρωτοκόλλου TLS

- Ο υπολογισμός κρυπτογραφικών πράξεων δημόσιου κλειδιού (για μεγάλους αριθμούς) έχει μεγάλο κόστος σε επεξεργαστική ισχύ
- Τυπικά, περισσότερο από το μισό επεξεργαστικό κόστος της χειραψίας SSL αφορά την RSA κρυπτογράφηση και αποκρυπτογράφηση του κοινού μυστικού κλειδιού
- Μετάδοση δεδομένων
  - Συμμετρική κρυπτογράφηση
  - Υπολογισμός MAC
  - Μικρότερο κόστος από τις κρυπτογραφικές πράξεις δημόσιου κλειδιού

Επαναχρησιμοποίηση συνόδου (Session resumption)

- Αν έχει προϋπάρξει επικοινωνία μεταξύ του client και του server, τότε το πρωτόκολλο χειραψίας μπορούν να προχωρήσουν απευθείας στη μετάδοση των δεδομένων
  - Για κάθε σύνοδο (session), ο client και ο server αποθηκεύουν τα: session_id, master_secret, negotiated ciphers
- Ο client στέλνει το session_id μέσα στο μήνυμα ClientHello
- Ο Server συμφωνεί στην επαναχρησιμοποίηση μέσα στο μήνυμα ServerHello
  - Το νέο block_key υπολογίζεται από το master_secret και τους τυχαίους αριθμούς του client και του server (r_C, r_S)
- Το TLS v1.3 καθιστά obsolete το session resumption.

Αυθεντικοποίηση client

- Μπορεί προαιρετικά να αυθεντικοποιείται και ο client στον server
- Ο server στέλνει ένα μήνυμα CertificateRequest στον client

Επιθέσεις στο πρωτόκολλο SSL/ TLS (1/3)

- Version rollback attack
  - Ο επιτιθέμενος προσπαθεί να πείσει τον server ότι ο client δεν υποστηρίζει ισχυρό cipher suite, ώστε να τους οδηγήσει να συμφωνήσουν σε weak cipher suite.
- BEAST attack (Browser Exploit Against SSL/TLS)
  - Εκμεταλλευόταν μία γνωστή αδυναμία του CBC mode στο TLS v.1.
- CRIME attack
  - Επιτρέπει την ανάκτηση των cookies εάν το TLS χρησιμοποιεί συμπίεση δεδομένων.

Επιθέσεις στο πρωτόκολλο SSL/ TLS (2/3)

- BREACH attack
  - Οδηγεί στην εξαγωγή login tokens, email διευθύνσεων και άλλων πληροφοριών από μία κρυπτογραφημένη TLS σύνδεση.
  - Για να επιτύχει η επίθεση, ο επιτιθέμενος οδηγεί αρχικά το θύμα να επισκεφτεί ένα κακόβουλο σύνδεσμο ή κάνει code injection σε έγκυρες σελίδες που επισκέπτεται το θύμα (π.χ. ελέγχει το ασύρματο δίκτυο που χρησιμοποιεί το θύμα)

Άλλες επιθέσεις:

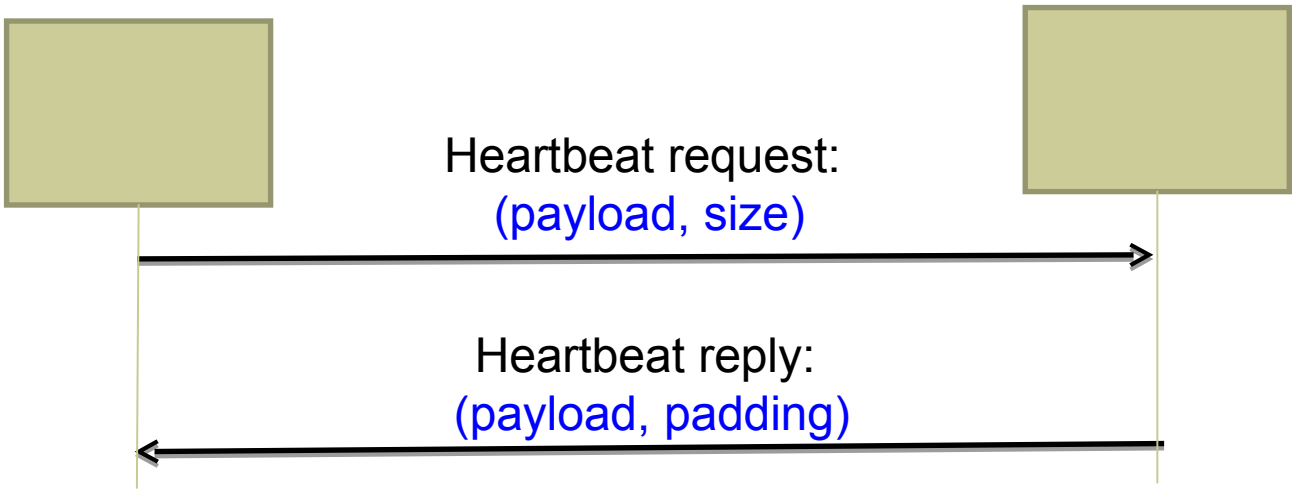
- Padding Oracle / Lucky Thirteen attack
- Renegotiation attack
- DES/ 3DES/ RC4 attacks
- Truncation attack

Επιθέσεις στο πρωτόκολλο SSL/ TLS (3/3)

- Heartbleed attack (CVE-2014-0160)
  - Οδηγεί στην αντιγραφή της μνήμης που χρησιμοποιεί το openssl.
  - Εκμεταλλεύεται μία αδυναμία του heartbeat extension του openssl
  - Όλες οι εκδόσεις 1.0.1 και ορισμένες beta εκδόσεις 1.0.2 είναι ευάλωτες στην επίθεση αυτή.

- Heartbeat extension
  - Διατηρεί «ζωντανό» το tls session σε περίπτωση που ένα από τα δύο μέρη είναι μη ενεργό για αρκετή ώρα, στέλνοντας hearbeat requests και λαμβάνοντας heartbeat replies.

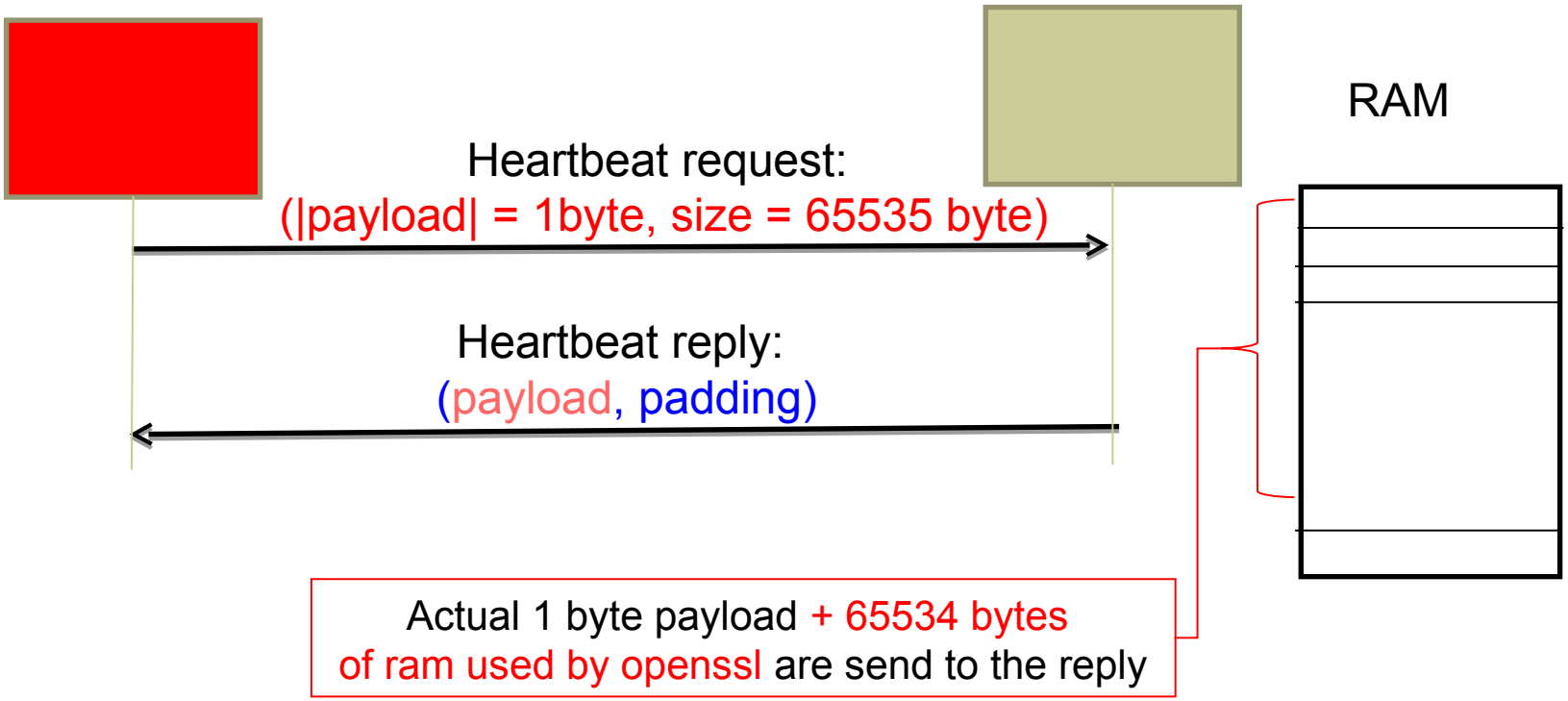
Η επίθεση Heartbleed (1/2)



Αδυναμία:

- Η υλοποίηση δεν ελέγχει εάν |payload| =? size
- Ένα κακόβουλο μέρος μπορεί να στείλει ένα μικρό payload και να δηλώσει ένα μεγάλο size.
- Τι θα συμβεί;

Η επίθεση Heartbleed (2/2)



Μέτρα αντιμετώπισης

- Certificate pinning ή SSL pinning.
  - Ο client προσθέτει ένα ακόμα βήμα ελέγχου:
    - Μόλις λάβει το certificate του server, ελέγχει εάν είναι «έμπιστο» με βάση έγκυρα δεδομένα επαλήθευσης (π.χ. ένα έγκυρο αντίγραφο του certificate από κάποια άλλη πηγή).
    - Π.χ. ο Google Chrome περιλαμβάνει επικύρωση δεδομένων για ένα πιστοποιητικό στο domain *.google.com
    - Οδήγησε στην αποκάλυψη πλαστών πιστοποιητικών το 2011.
    - Σε άλλα συστήματα η ασφάλεια στηρίζεται στο ότι την πρώτη φορά που ο client λαμβάνει και αποθηκεύει ένα πιστοποιητικό, αυτό είναι έγκυρο. Σε επόμενες συνόδους με τον ίδιο server, ο client ελέγχει το πιστοποιητικό που λαμβάνει από τον server με το ήδη αποθηκευμένο, για προστασία από επιθέσεις MITM.

6. Εφαρμογή SSL/TLS σε apache web server με τη χρήση του openssl

OpenSSL

- Ευρέως διαδεδομένη εφαρμογή ελεύθερου λογισμικού για την υλοποίηση των πρωτοκόλλων SSL/TLS.
- Λειτουργεί σε κάθε μεγάλη πλατφόρμα
  - Unix / Linux (συνήθως pre-compiled packets)
  - Windows (http://www.openssl.org/related/binaries.html)
- Πληροφορίες
  - http://www.openssl.org/
  - http://www.openssl.org/docs/ (documentation)
  - http://www.openssl.org/docs/apps/openssl.html (περιγραφή εντολών OpenSSL)

Έλεγχος και ανανέωση έκδοσης openssl

Το kali χρησιμοποιεί έκδοση του openssl η οποία είναι ευάλωτη (μεταξύ άλλων) στο heartbleed attack

Περιγραφή	**Ενέργειες**
Έλεγχος έκδοσης	(Στο terminal): **openssl version**
Download τελευταίας έκδοσης σε κάποιο φάκελο	(Στον web browser): **opensssl.org/sources** Κατέβασμα τελευταίας έκδοσης σε φάκελο (π.χ. /home)
	(Στο terminal): **cd /home** **tar –xvf openssl**<tab> **cd /home/openssl**<tab> **./config --prefix=/usr --openssldir=/etc/ssl** **make** **make install**
Έλεγχος νέας έκδοσης	**openssl version**

Επεκτάσεις αρχείων openssl

- KEY: περιέχει το ιδιωτικό κλειδί (απαιτείται προστασία του αρχείου)
- CSR (Certificate Signing Request): αίτημα προς την Α.Π. για την υπογραφή ενός πιστοποιητικού (χρήστη, server κτλ)
- CRT: Περιέχει ένα πιστοποιητικό και διανέμεται ελεύθερα σε όλους
- PEM: Αρχείο που περιέχει και το ιδιωτικο κλειδί και το Πιστοποιητικό (είναι απαραίτητο σε ορισμένους server). Απαιτείται προστασία του αρχείου
- CRL (Certificate Revokation List): Λίστα Ανάκλησης Πιστοποιητικών η οποία περιλαμβάνει όσα πιστοποιητικά είχε εκδώσει στο παρελθόν η Α.Π. αλλά δεν τα θεωρεί πλέον αξιόπιστα.

Χρήση OpenSSL

- Δημιουργία πιστοποιητικών X.509
- Δημιουργία αιτημάτων πιστοποίησης
- Πιστοποίηση κλειδιών χρηστών
- Δημιουργία παραμέτρων κλειδιών για RSA, DSA
- Δημιουργία λιστών ανάκλησης πιστοποιητικών (CRLs)
- Υπολογισμός αποτελεσμάτων συναρτήσεων κατακερματισμού
- Κρυπτογράφηση - αποκρυπτογράφηση

Χρήση OpenSSL

- Μορφή εντολών OpenSSL

openssl command [command_opts] [command_args]

Βήματα:
(όλα εκτελούνται σε περιβάλλον Linux)

1. Εκκίνηση server, δοκιμή https πρόσβασης
2. Ενεργοποίηση ssl mod σε apache web server και προβολή πιστοποιητικού
3. Δημιουργία νέου (self-signed) πιστοποιητικού με openssl
4. Εγκατάσταση και δοκιμή πιστοποιητικού

1) Εκκίνηση server, δοκιμή https πρόσβασης

Περιγραφή	**Ενέργειες**
Δοκιμή http στο localhost	`(Στον browser)` **`http://localhost`**
Εκκίνηση apache2	`(Terminal)` **`service apache2 start`**
Δοκιμή http στο localhost	**`http://localhost`**
Δοκιμή https	**`https://localhost`**

2) Ενεργοποίηση ssl mod και προβολή πιστοποιητικού

Περιγραφή	**Ενέργειες**
Επισκόπηση `/etc/apache2`	`(mods available, mods enabled sites available, sites enabled)`

Ενεργοποίηση ssl	`a2enmod ssl`
	`service apache2 restart`
	`a2ensite default-ssl`
	`service apache2 reload`

Δοκιμή https	`https://localhost`
Προβολή του πιστοποιητικού	`(δημιουργία προσωρινής εξαίρεσης και προβολή πιστοποιητικού)`
Επισκόπηση `/etc/ssl`	`(ssl/certs, ssl/private)`

3) Δημιουργία self-signed πιστοποιητικού με openssl

Περιγραφή	**Ενέργειες**
Δημιουργία καταλόγου για τα νέα κλειδιά	`mkdir /etc/apache2/mySSLKeys`
Δημιουργία νέου πιστοποιητικού	`openssl req -new -x509 -days 365 -nodes -out /etc/apache2/mySSLKeys/serverCert.pem -keyout /etc/apache2/mySSLKeys/serverKey.key`
Επισκόπηση κλειδιού και πιστοποιητικού	`cd /etc/apache2/mySSLKeys` `ls`

4) Εγκατάσταση και δοκιμή πιστοποιητικού

Περιγραφή	**Ενέργειες**
Επεξεργασία αρχείου διαμόρφωση site	`Επεξεργασία αρχείου /etc/apache2/sites-available/default-ssl`

Προσθήκη κλειδιού και πιστοποιητικού	`SSLCertificateFile /etc/apache2/mySSLKeys/serverCert.pem`
	
	`SSLCertificateKeyFile /etc/apache2/mySSLKeys/serverKey.key`

Επανεκκίνηση apache	`service apache2 restart`
Δοκιμή και επισκόπηση πιστοποιητικού	

Βιβλιογραφικές πηγές

1. The Transport Layer Security (TLS) Protocol Version 1.3 (draft) draft-ietf-tls-rfc5246-bis-00 https://tools.ietf.org/html/draft-ietf-tls-rfc5246-bis-00 και https://tlswg.github.io/tls13-spec/
2. RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2". http://tools.ietf.org/html/rfc5246 *(The current standard replaces these former versions, which are now considered obsolete)*
3. RFC 4346: "The Transport Layer Security (TLS) Protocol Version 1.1". http://tools.ietf.org/html/rfc4346
4. RFC 2246: "The TLS Protocol Version 1.0". http://tools.ietf.org/html/rfc2246
5. Openssl project, http://www.openssl.org
6. Shawn Fitzgerald and Pratik Guha Sarkar: “Attacks on SSL – A comprehensive study of BEAST, CRIME, TIME, BREACH, LUCKY13 and RC4 BIASES”, https://www.isecpartners.com/media/106031/ssl_attacks_survey.pdf