SICUREZZA AL LIVELLO TRASPORTO: "SECURE SOCKET LAYER" – SSL

Corso di Laurea Magistrale in Ingegneria Informatica

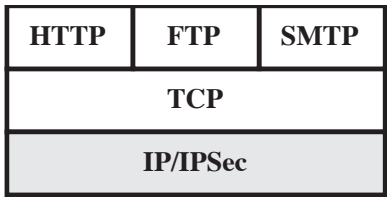
A.A. 2015/2016

Prof. Simon Pietro Romano

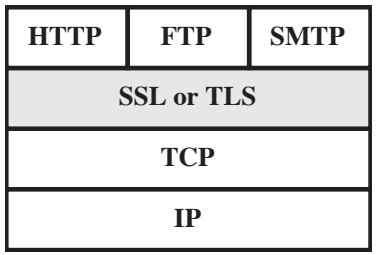
spromano@unina.it

STACK TCP/IP: MECCANISMI DI SICUREZZA

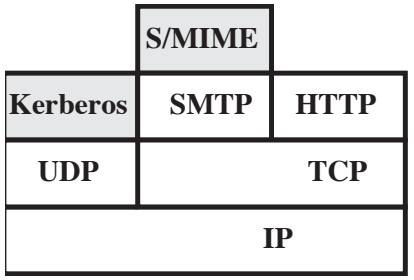
- A livello rete:
  - IPSec
    - Trasparente agli utenti finali e alle applicazioni
    - Funzionalità di filtraggio del traffico



- A livello trasporto:
  - Sicurezza “sopra” al protocollo TCP
    - SSL (Secure Socket Layer):
      - Trasparente alle applicazioni…
      - …o incorporato nelle applicazioni stesse



- A livello applicazione
  - Incorporare specifici meccanismi di sicurezza all’interno delle applicazioni:
    - es: posta elettronica sicura:
      - S/MIME

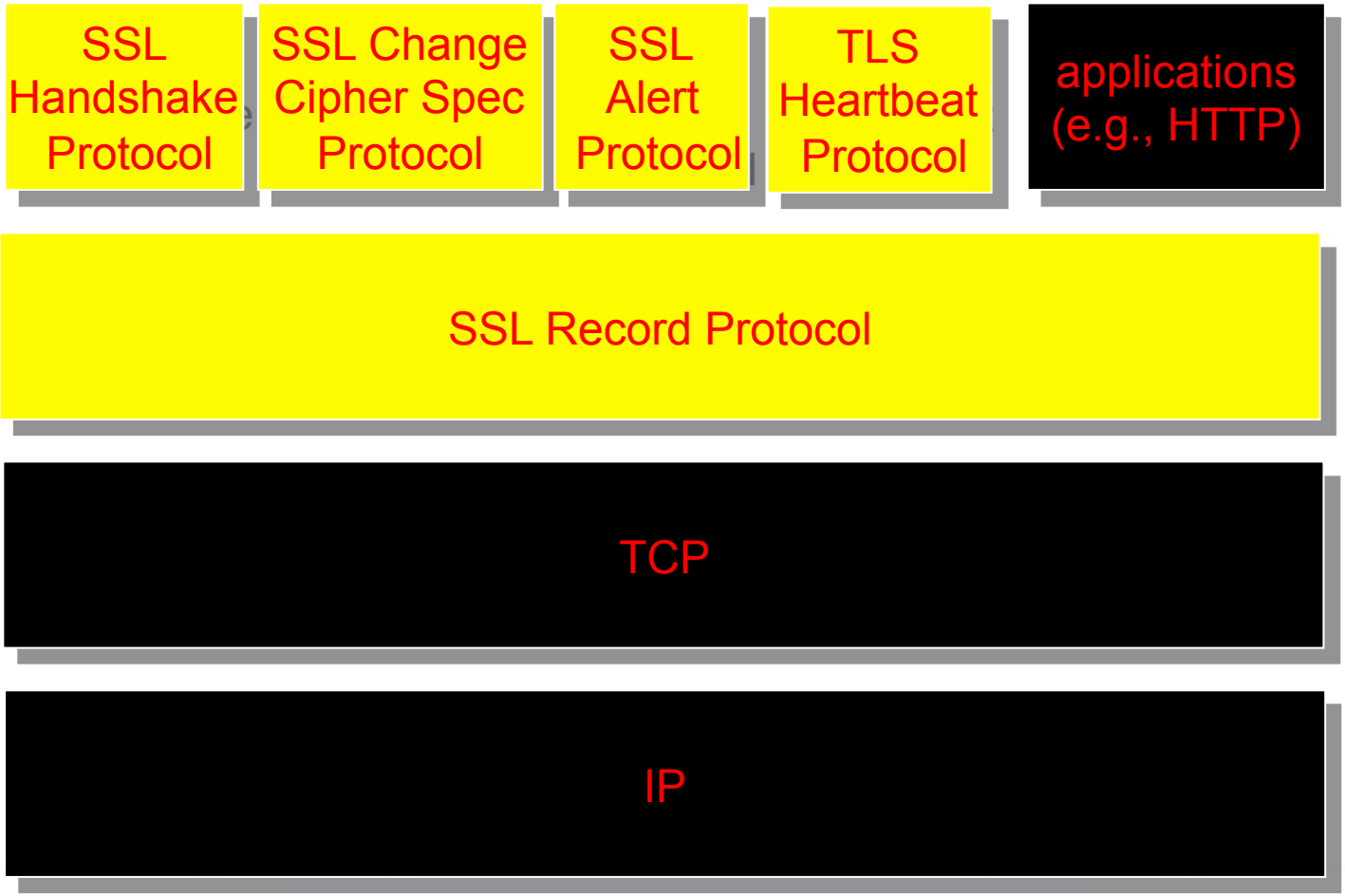


SSL – SECURE SOCKET LAYER

COSA SONO SSL E TLS?

- SSL: “Secure Socket Layer”
- TLS: “Transport Layer Security”
- Entrambi forniscono una connessione a livello trasporto sicura per le applicazioni
  - es: tra un web server e un browser
- SSL fu sviluppato da Netscape
- La versione 3.0 di SSL è stata implementata in molti browser e web-server ed è ampiamente utilizzata in Internet
- SSL v3.0 è stato proposto come standard in un Draft Internet del 1996
- TLS è un’evoluzione di SSL, ultima versione (TLS v1.2) in RFC 5246 del 2008
- TLS può essere visto come SSL v3.1 (TLS 1.2 -> protocol version 3.3)

L'ARCHITETTURA SSL



I LIVELLI PROTOCOLLARI IN SSL/TLS

- SSL non è un unico protocollo, ma una suite di due livelli protocollari:
  - Il protocollo SSL Record Protocol:
    - fornisce servizi di sicurezza di base ai protocolli di livello superiore
  - I protocolli “Handshake Protocol”, “Change Cipher Spec Protocol”, “Alert Protocol”, “Heartbeat Protocol”
    - utilizzati per la gestione degli scambi SSL

I CONCETTI DI CONNESSIONE E SESSIONE

- **Connessione**
  - Forma di trasporto (ISO/OSI) temporanea
  - Ogni connessione è associata ad una sola sessione
- **Sessione**
  - Associazione tra client e server
  - Creata dall'Handshake Protocol
  - Definisce una serie di parametri di sicurezza crittografica da utilizzare in più connessioni
  - Evita la ri-negoziazione per ogni singola connessione di una sessione tra client e server
- Due entità possono instaurare più connessioni sicure

GLI STATI DELLA SESSIONE

- Ad ogni sessione vengono associati più stati
- Per ogni sessione attivata ci sono uno stato operativo corrente per l'invio dei dati e uno per la ricezione
- Durante il protocollo di Handshake vengono creati due stati provvisori di invio e ricezione
- Alla fine del protocollo di Handshake gli stati diventano correnti

PARAMETRI DI STATO DI SESSIONE

- Session Identifier
  - Sequenza di byte per identificare una sessione
- Peer Certificate
  - Certificato X509 v3 del nodo (può essere nullo)
- Compression Method
  - Algoritmo per comprimere i dati prima della crittografia
- Cipher Spec
  - specifica l'algoritmo di crittografia dei dati e l'algoritmo di hash per il calcolo del codice MAC (Message Authentication Code)
- Master secret
  - Codice segreto di 48 bit condiviso tra client e server
- Is Resumable
  - Indica se la sessione puo' essere utilizzata per nuove connessioni

STATO DI CONNESSIONE: PARAMETRI (1/2)

- Server Random, Client Random
  - Sequenze di byte scelte per ciascuna connessione
- Server write MAC secret
  - Chiave segreta per operazioni MAC sui dati inviati dal server
- Client write MAC secret
  - Chiave segreta per operazioni MAC sui dati inviati dal client
- Server write key
  - Chiave di crittografia convenzionale per dati crittografati dal server e decrittografati dal client
- Client write key
  - Chiave di crittografia convenzionale per dati crittografati dal client e decrittografati dal server

STATO DI CONNESSIONE: PARAMETRI (2/2)

- Initialization Vectors
  - Vettore di inizializzazione associato a ciascuna chiave durante la cifratura a blocchi
  - Viene inizializzato dal protocollo di Handshake, ultimo ciphertext conservato per IV di nuovo record
- Sequence Numbers
  - Ciascuna parte gestisce numeri di sequenza distinti per i messaggi trasmessi e ricevuti per ogni connessione

STATO DI CONNESSIONE: PARAMETRI (2/2 !)

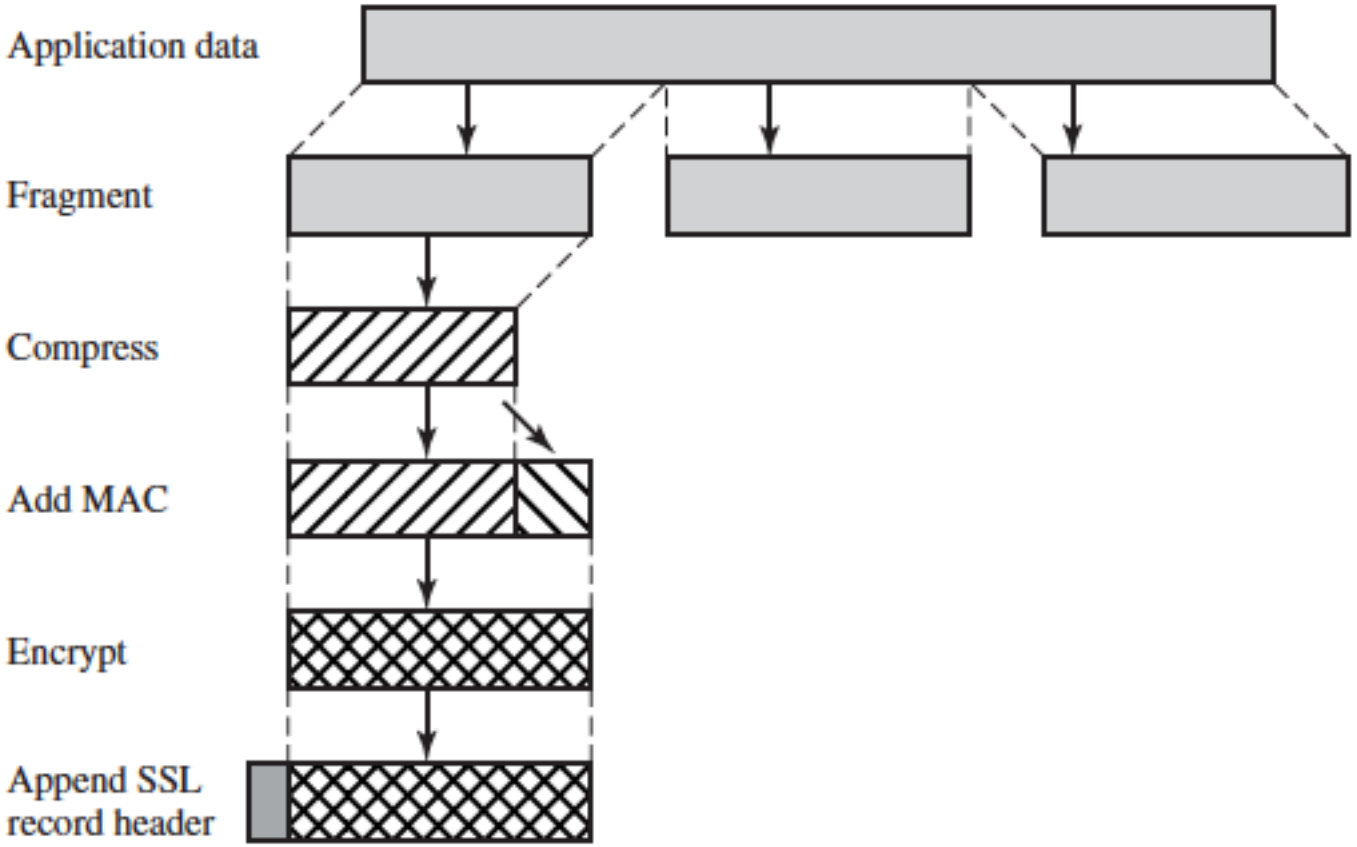
- Initialization Vectors
  - Vettore di inizializzazione associato a ciascuna chiave durante la cifratura a blocchi
  - Viene inizializzato dal protocollo di Handshake, ultimo ciphertext conservato per IV di nuovo record
- Sequence Numbers
  - Ciascuna parte gestisce [illegible] di sequenza distinti per i messaggi tra[illegible] connessione

Rizzo and Duong, Here Come The XOR Ninjas (2011)
Exploit– BEAST tool
Soluzione: IV espliciti in TLS 1.1

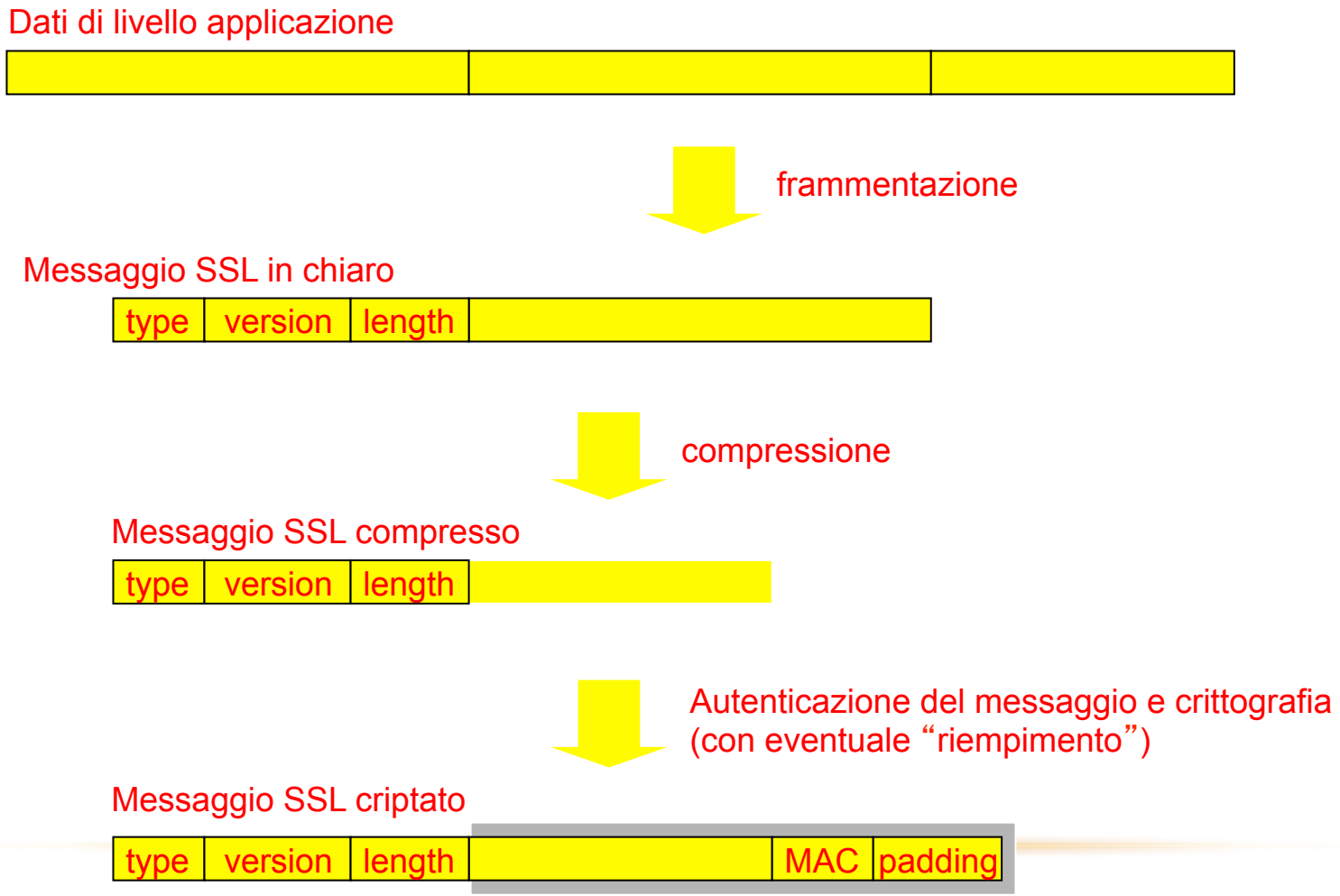
IL PROTOCOLLO SSL RECORD

- Il protocollo SSL Record fornisce due servizi per le connessioni SSL: Segretezza ed Integrità del messaggio
- Segretezza
  - Il protocollo Handshake definisce una chiave segreta condivisa per la crittografia del payload SSL
- Integrità del messaggio
  - Il protocollo Handshake definisce una chiave segreta condivisa per creare un codice di integrità MAC

PROTOCOLLO SSL RECORD: PRINCIPIO BASE



PROTOCOLLO SSL RECORD



FRAMMENTAZIONE E COMPRESSIONE

- Frammentazione
  - Ogni messaggio di livello superiore viene frammentato in blocchi di 2^{12} byte (16384 byte) o meno
- Compressione
  - Deve essere senza perdita di informazioni
  - È opzionale
  - In SSLv3 e in TLS non è specificato alcun algoritmo di compressione:
    - *L'algoritmo di compressione è "null"*

CALCOLO DEL CODICE MAC

- Viene utilizzata una chiave segreta condivisa:

MAC = hash (MAC_write_secret || pad_2
|| hash(MAC_write_secret || pad_1 || seq_num || type || length || fragment))

- dove:
  - || = concatenamento
  - MAC_write_secret = chiave segreta condivisa
  - hash = algoritmo crittografico (MD5 o SHA-1)
  - pad_1 = il byte 0x36 (0011 0110) ripetuto 48 volte (384 bit) per MD5 e 40 volte (320 bit) per SHA-1
  - pad_2 = il byte 0x5C (0101 1100) ripetuto 48 volte per MD5 e 40 volte per SHA-1
  - seq_num = numero di sequenza del messaggio
  - type = protocollo di livello superiore utilizzato per elaborare il frammento
  - length = lunghezza del frammento compresso
  - fragment = il frammento compresso (il frammento in chiaro se non si usa la compressione)

CRITTOGRAFIA (1/2)

Il messaggio compresso più il codice MAC vengono crittografati mediante crittografia simmetrica

Algoritmi di crittografia supportati (dimensione della chiave in bit)

- Crittografia a blocchi
  - AES (128 o 256)
  - IDEA (128)
  - RC2-40 (40)
  - DES-40 (40)
  - DES (56)
  - 3DES (168)
  - Fortezza (80)
- Crittografia di flussi
  - RC4-40 (40)
  - RC4-128 (128)

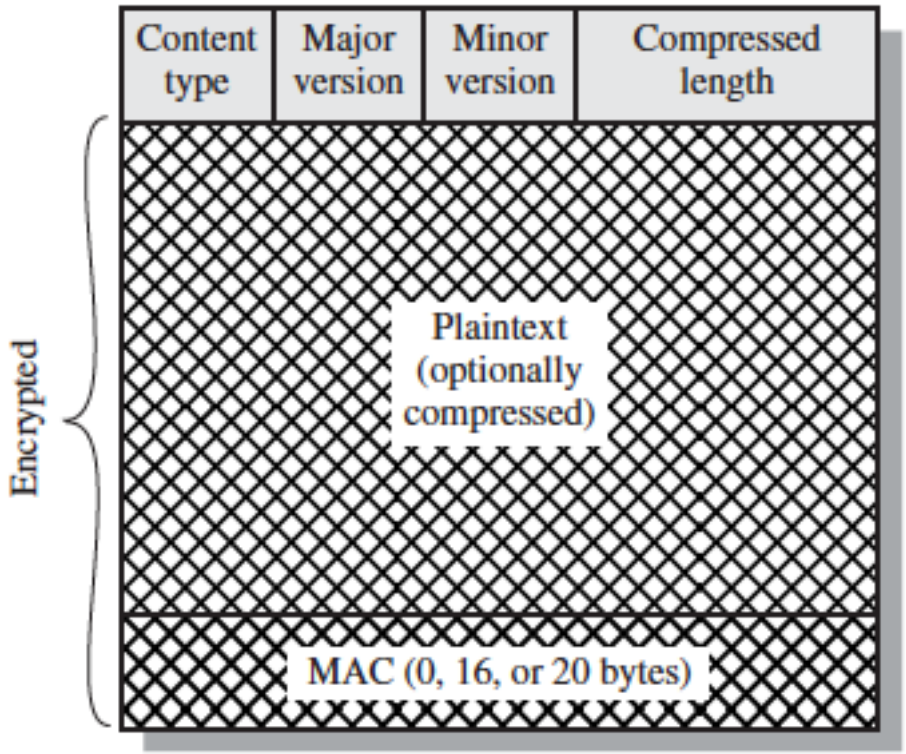
CRITTOGRAFIA (2/2)

- Da notare che il MAC viene calcolato prima della crittografia e aggiunto al testo in chiaro
- Per la crittografia a blocchi possono essere aggiunti dei byte di padding dopo il codice MAC
  - Byte di riempimento seguiti da un byte che indica la lunghezza del riempimento stesso
- Il numero di byte di riempimento è il più piccolo valore tale che la dimensione totale dei dati da crittografare sia un multiplo della lunghezza del blocco di testo cifrato

HEADER

- **Content type** (8bit): Il protocollo di livello superiore utilizzato per elaborare il frammento
  - Change_cipher_spec
  - Alert
  - Handshake
  - Dati applicazione
  - Heartbeat
- **Major Version** (8 bit)
  - La versione major di SSL in uso (per SSL v3 è 3)
- **Minor Version** (8 bit)
  - La versione minor di SSL in uso (per SSL v3 è 0)
- **Compressed Length** (16 bit)
  - La lunghezza in byte del frammento compresso

FORMATO DI UN RECORD SSL



IL PROTOCOLLO CHANGE CIPHER SPEC

- È uno dei protocolli specifici per SSL che utilizzano il formato SSL Record
- È costituito da un solo byte contenente il valore 1
- Scopo del messaggio è quello di far in modo che:
  - lo stato provvisorio venga copiato nello stato corrente (e provvisiorio settato a NULL)
  - la tecnica di cifratura utilizzata nella connessione sia aggiornata ed attivata

IL PROTOCOLLO ALERT

- Viene utilizzato per trasmettere allarmi SSL tra gli end-system
- Come per le atre applicazioni che utlizzano SSL, i messaggi di alert sono compressi e crittografati
- Un messaggio alert è costituito da due byte
  - Level
  - Alert
- Il byte Level può assumere i valori
  - 1 (warning)
  - 2 (fatal)
- Se il livello è “fatal” SSL chiude immediatamente la connessione
- Altre connessioni attive nella stessa sessione possono continuare ma non è possibile attivarne di nuove

I CODICI DI ALERT: IRREVERSIBILI

- unexpected_message
  - è stato ricevuto un messaggio inappropriato
- bad_record_mac
  - è stato ricevuto un codice MAC errato
- decompression_failure
  - la funzione di compressione ha ricevuto un input errato
- handshake_failure
  - il mittente non è stato in grado di negoziare un insieme di parametri di sicurezza accettabili date le opzioni disponibili
- illegal_parameter
  - un campo in un messaggio di handshake è oltre i limiti consentiti o è incoerente rispetto agli altri campi

I CODICI DI ALERT: ALTRI

- close_notify
  - notifica al destinatario che il mittente non invierà altri messaggi nella connessione corrente
  - ciascuna parte deve inviare un “close_notify” quando si vuole chiudere la connessione sul lato “scrittura”
- no_certificate
  - è inviato in risposta ad una richiesta di certificato qualora non si disponga di un certificato di risposta adeguato
- bad_certificate
  - un certificato ricevuto risulta alterato
- unsupported_certificate
  - il tipo di certificato ricevuto non è supportato
- certificate_revoked
  - il certificato è stato revocato dal suo firmatario
- certificate_expired
  - il certificato è scaduto
- certificate_unknown
  - errore non specificato nell’elaboazione del certificato che lo rende inutilizzabile

IL PROTOCOLLO DI HANDSHAKE

- Il protocollo di handshake consente al client ed al server:
  - di autenticarsi reciprocamente
  - di negoziare un algoritmo di crittografia e MAC
  - di negoziare le chiavi crittografiche da utilizzare per proteggere i dati
- Viene utilizzato prima della trasmissione di qualsiasi dato dell'applicazione
- È costituito da una serie di messaggi scambiati tra il client e il server

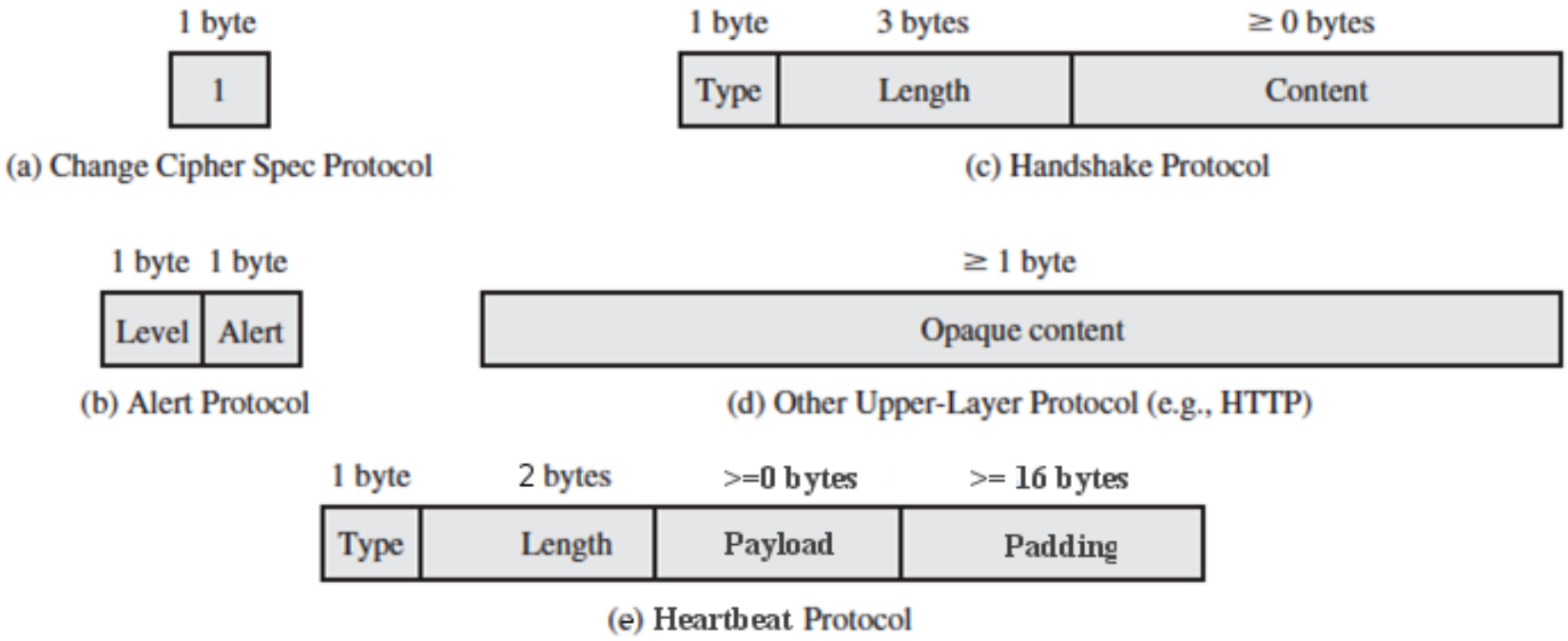
HANDSHAKE: FORMATO DEI MESSAGGI (1/2)

- Type (1 byte)
  - Indica il particolare tipo di messaggio di handshake
- Length (3 byte)
  - La lunghezza del messaggio in byte
- Content (>= 0 byte)
  - I parametri associati a questo messaggio

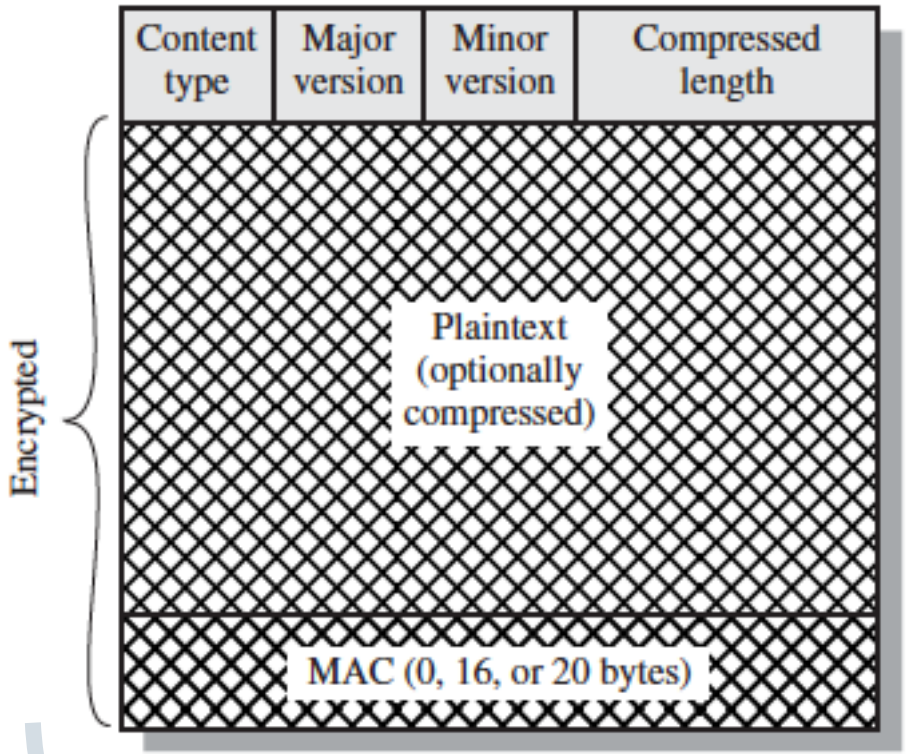
HANDSHAKE: FORMATO DEI MESSAGGI (2/2)

Tipo di messaggio	Parametri
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	parameter, signature
finished	hash value

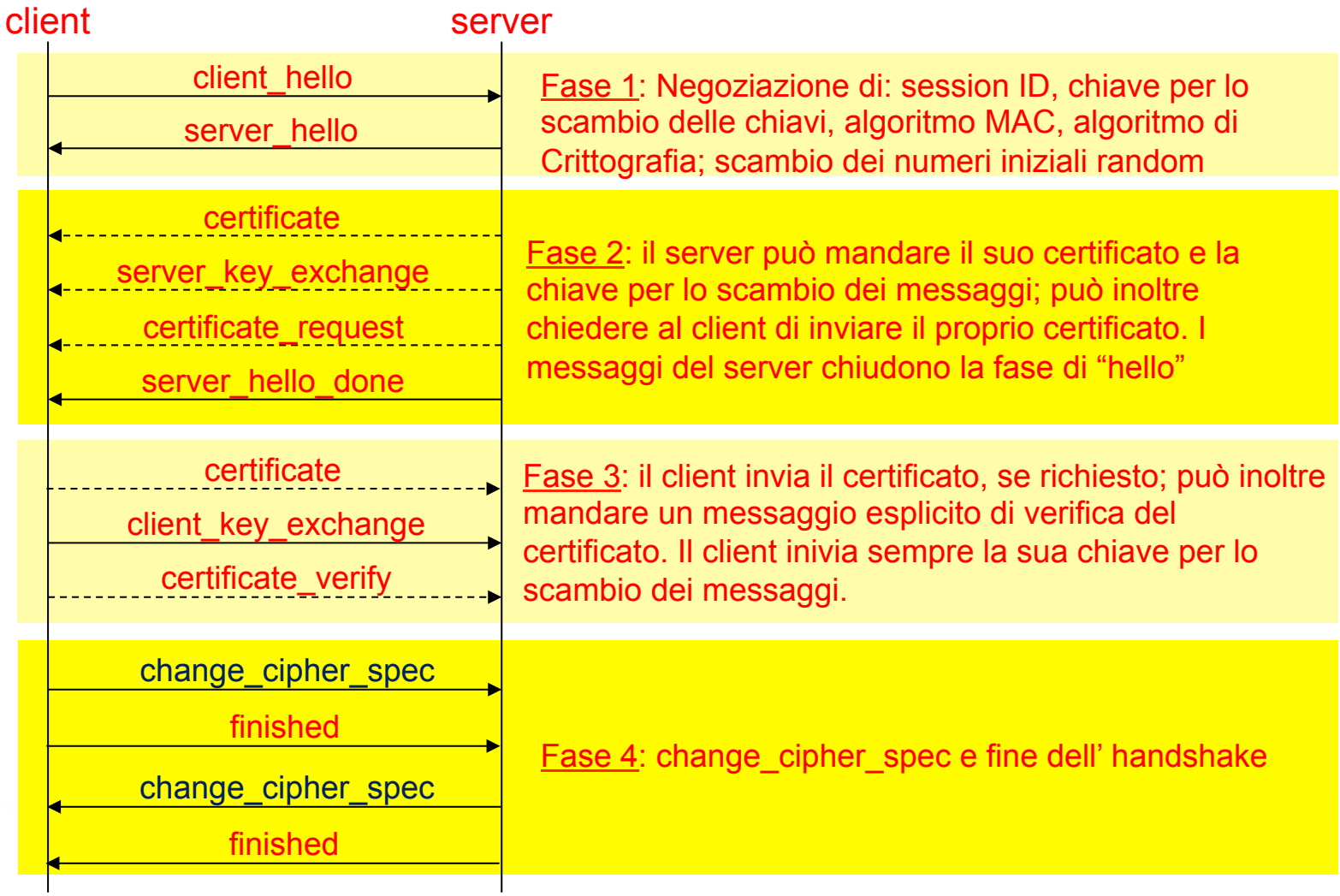
SSL E PROTOCOLLI: SINTESI



FORMATO DI UN RECORD SSL (DEJA-VU)



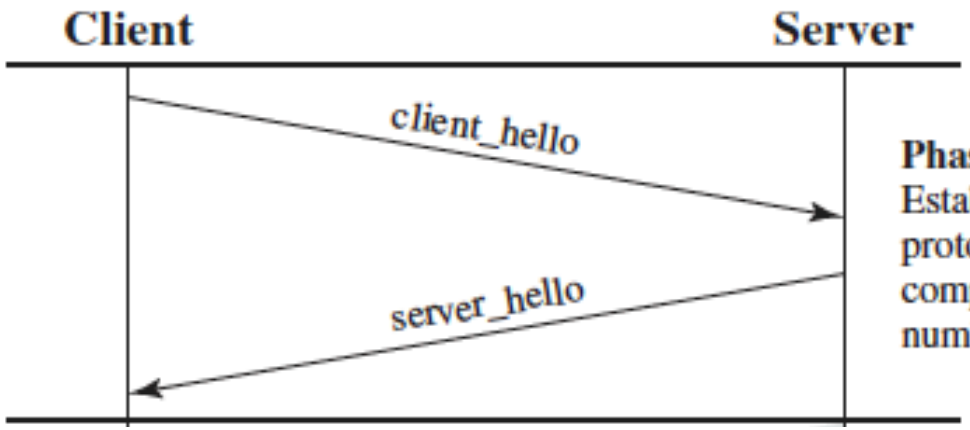
IL PROTOCOLLO DI HANDSHAKE IN AZIONE



INIZIALIZZAZIONE FUNZIONALITÀ DI SICUREZZA

- In questa fase viene avviata una connessione logica per stabilire le funzionalità di sicurezza che possono essere impiegate
- Lo scambio viene iniziato dal client che invia un messaggio client_message

IL PROTOCOLLO DI HANDSHAKE: FASE 1



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

IL MESSAGGIO CLIENT_HELLO

- **Version --** La versione di SSL più elevata fra quelle utilizzabili dal client
- Random
  - Struttura casuale generata dal client costituita da un timestamp di 32 bit e 28 byte generati da un generatore di numeri casuali “sicuro”
  - Funge da “nonce”, contro gli attacchi di tipo “replay”
- Session ID -- Identificatore di sessione di lunghezza variabile
  - vuoto se il client vuole creare una nuova connessione nell’ambito di una nuova sessione
  - corrispondente all’ID di una sessione esistente, se il client vuole:
    - aggiornare una connessione attiva nell’ambito della sessione
    - creare una nuova connessione nell’ambito della sessione
- CipherSuite -- Elenco degli algoritmi crittografici supportati dal client in ordine di preferenza
- Compression Method -- Elenco dei metodi di compressione supportati dal client

IL MESSAGGIO SERVER_HELLO

- **Version --** La versione di SSL più bassa tra quella suggerita dal client e quella più alta supportata dal server
- **Random --** Viene generato dal server ed è indipendente dal campo random del client
- **Session ID**
  - Stesso valore del session ID del client se questo è diverso da zero
  - Se il client vuole creare una nuova sessione il server genera un nuovo session ID
- **CipherSuite** (cfr. slide successiva…)
  - Metodo crittografico scelto dal server tra quelli proposti e supportati dal client
  - Per ogni elemento della lista c'è l'algoritmo di scambio delle chiavi ed un "CipherSpec"
- **Compression Method** -- Metodo di compressione scelto dal server tra tutti quelli proposti e supportati dal client

IL MESSAGGIO SERVER_HELLO (!)

- **Version --** La versione di SSL più bassa tra quella suggerita dal client e quella più alta supportata dal server
- **Random --** Viene generato dal server ed è indipendente dal campo random del client
- **Session ID**
  - Stesso valore del session ID del client se questo è diverso da zero
  - Se il client vuole creare
    session ID
- **CipherSuite** (cfr. slide succe
  - Metodo crittografico sce
    client
  - Per ogni elemento della lista c'è
    “CipherSpec”
- **Compression Method** -- Metodo di compressione scelto dal server tra tutti quelli proposti e supportati dal client

Kelsey, J.: Compression and information leakage of plaintext. (2002).
Exploit pubblicato in 2012 – CRIME attack
Soluzione: compressione disabilitata in **draft** TLS 1.3

IL MESSAGGIO SERVER_HELLO (!!)

- **Version --** La versione di SSL più bassa tra quella suggerita dal client e quella più alta supportata dal server
- **Random --** Viene generato dal server ed è indipendente dal campo random del client
- **Session ID**
  - Stesso valore del session ID del client se questo è diverso da zero
  - Se il client vuole creare u
    session ID
- **CipherSuite** (cfr. slide succe
  - Metodo crittografico sce
    client
  - Per ogni elemento della
    "CipherSpec"
- **Compression Method** --Met
  quelli proposti e supportati dal client

Kelsey, J.: Compression and information leakage of plaintext. (2002).
Exploit pubblicato in 2012 – CRIME attack
Soluzione: compressione disabilitata in **draft** TLS 1.3
** Non basta: **
Exploit pubblicato in 2013 -- BREACH attack
(sfrutta compressione di HTTP, ed effetto su crittografia)

METODI DI SCAMBIO DELLE CHIAVI

- **RSA** -- La chiave segreta viene crittografata con la chiave pubblica RSA del destinatario
- **Fixed Diffie-Hellman**
  - I parametri pubblici di D-H del server sono contenuti in un certificato firmato da una CA
  - Il client fornisce i propri parametri della chiave pubblica D-H in un certificato o attraverso un messaggio per lo scambio delle chiavi
  - Questo metodo produce come risultato una chiave segreta fissa fra i due nodi basata sul calcolo D-H, utilizzando chiavi pubbliche fisse
- **Ephemeral Diffie-Hellman**
  - Utilizzato per creare chiavi temporanee
  - Vengono scambiate le chiavi pubbliche D-H temporanee firmate utilizzando la chiave privata RSA
  - Il destinatario può verificare l'autenticità con la corrispondente chiave pubblica
  - I certificati vengono utilizzati per autenticare la chiave pubblica
  - Il **più sicuro** tra gli approcci DH, perché produce una chiave temporanea autenticata!
- **Anonymous Diffie_Hellman** -- Ciascun peer invia all'altro i propri parametri D-H temporanei senza autenticarli
- **Fortezza** -- Schema proprietario di scambio delle chiavi alternativo a D-H

METODI DI SCAMBIO DELLE CHIAVI (!)

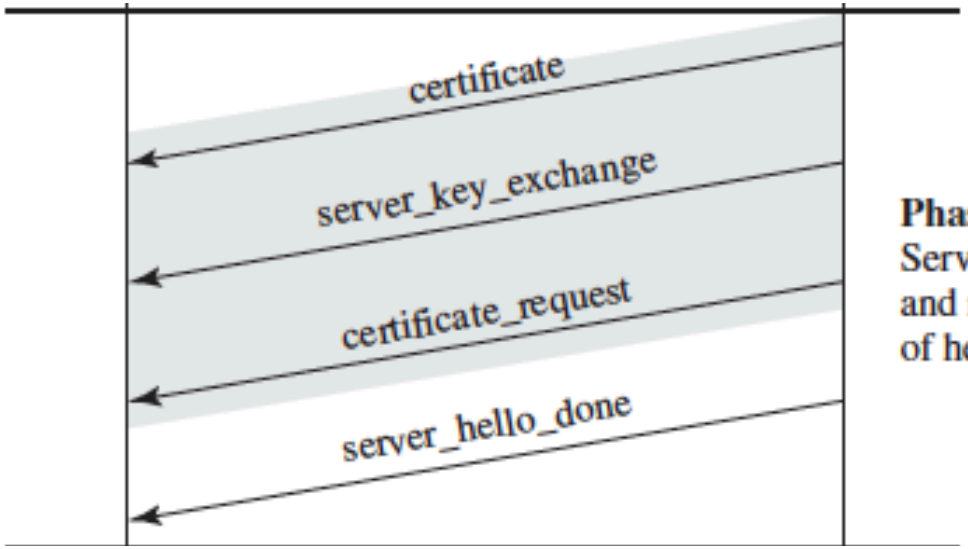
- **RSA** -- La chiave segreta viene crittografata con la chiave pubblica RSA del destinatario
- **Fixed Diffie-Hellman**
  - I parametri pubblici di D-H del server sono contenuti in un certificato firmato da una CA
  - Il client fornisce i propri parametri della chiave pubblica D-H in un certificato o attraverso un messaggio per lo scambio delle chiavi
  - Questo metodo produce come risultato una chiave segreta fissa fra i c… li basata sul calcolo D-H, utilizzando chiavi pubbliche fisse
- **Ephemeral Diffie-Hellman**
  - Utilizzato per creare chiavi te…
  - Vengono scambiate le chiav… RSA
  - Il destinatario può verificare…
  - I certificati vengono utilizzat…
  - Il **più sicuro** tra gli approcci…
- **Anonymous Diffie_Hellman** -- C… autenticarli
- **Fortezza** -- Schema proprietario di…

Problemi con PKI, CA, implementazioni di constraints…
2005 Lenstra *et al.* (teoria) -> 2008 Sotirov *et al.* (pratico)
2008 Moxie Marlinspike pubblica report, crea ***sslsniff***
Compromissione di CA, etc almeno fino al 2013

CIPHERSPEC

- **CipherAlgorithm** -- Uno degli algoritmi di crittografia supportati da SSL
- **MACAlgorithm** -- MD5 o SHA-1
- **CipherType** -- A blocchi o a flussi
- **IsExportable** -- True o False
- **HashSize** -- 0, 16 (per MD5), 20 (per SHA-1) byte
- **Key Material** -- Una sequenza di byte che contiene i dati utilizzati per la generazione delle chiavi di scrittura
- **IV Size** -- Le dimensioni per il vettore di inizializzazione

IL PROTOCOLLO DI HANDSHAKE: FASE 2



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

AUTENTICAZIONE SERVER E SCAMBIO CHIAVI

- Il server inizia questa fase inviando il proprio certificato X.509 o una catena di certificati nel messaggio “certificate”
- Il messaggio “certificate”
  - È obbligatorio per ogni metodo di scambio delle chiavi, escluso Anonymous Diffie-Hellman
  - Con il metodo Fixed Diffie-Hellman, funge da messaggio per lo scambio delle chiavi:
    - contiene i parametri pubblici D-H del server
- Il messaggio “server_key_exchange” non è necessario se:
  - si utilizza Fixed Diffie-Hellman
  - si utilizza lo scambio delle chiavi RSA

SERVER_KEY_EXCHANGE

- Messaggio obbligatorio nei seguenti casi:
  - Anonymous Diffie-Hellman
    - contiene i due valori globali di D-H e la chiave pubblica D-H del server
  - Ephemeral Diffie-Hellman
    - contiene i due valori globali di D-H, la chiave pubblica D-H del server e una firma di questi parametri
  - Scambio RSA in cui il server ha una chiave ***RSA di sola firma***: il server
    - crea una coppia di chiavi pubblica/privata temporanea
    - utilizza il messaggio server_key_exchange per inviare la chiave pubblica
      - il contenuto del messaggio include:
        - i due parametri (esponente e modulo) della chiave pubblica temporanea RSA
        - una firma di tali parametri
  - Fortezza

FIRMA DIGITALE

- La firma viene creata a partire dal valore hash di un messaggio e crittografandolo con la chiave privata del mittente:

$$hash(client\_random \,||\, server\_random \,||\, server\_params)$$

- Digital Signature Standard (DSS):
  - si utlizza il calcolo hash SHA-1
- Rivest-Shamir-Adleman (RSA):
  - si utlizzano entrambi i valori di hash MD5 e SHA-1
  - il loro concatenamento viene crittografato mediante chiave privata del server

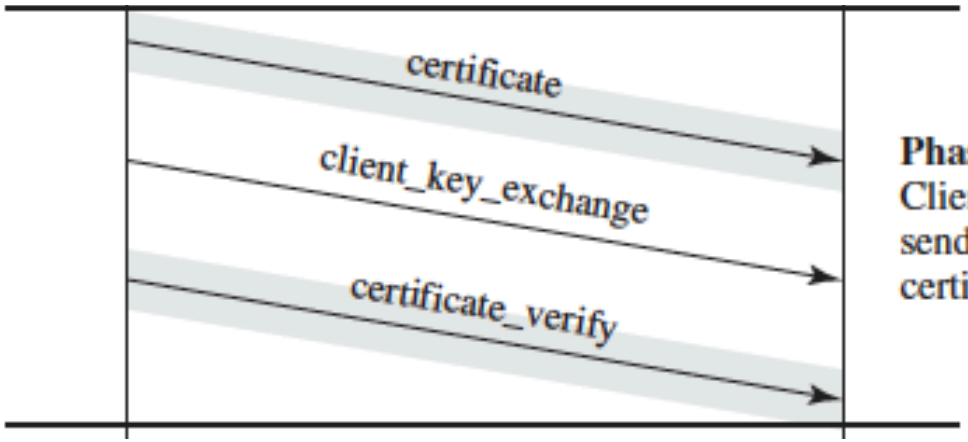
IL MESSAGGIO CERTIFICATE_REQUEST

- È inviato qualora sia richiesta l'autenticazione del client
- Specifica due parametri
  - certificate_type
  - certificate_authorities
- Il certificate_type indica quale algoritmo a chiave pubblica deve essere utilizzato
  - RSA
  - DSS
  - RSA per fixed Diffie-Hellman
  - DSS per fixed Diffie-Hellman
  - RSA per ephemeral Diffie-Hellman
  - DSS per ephemeral Diffie-Hellman
  - Fortezza

IL MESSAGGIO SERVER_HELLO_DONE

- È inviato per indicare la fine dei messaggi di hello del server
- Dopo aver inviato questo messaggio il server si mette in attesa della risposta del client
- Il messaggio non contiene alcun parametro

IL PROTOCOLLO DI HANDSHAKE: FASE 3



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

AUTENTICAZIONE CLIENT E SCAMBIO CHIAVI (1/2)

- Ricevuto un “server_done” il client deve:
  - verificare la validità del certificato del server
  - verificare i parametri del messaggio “server_hello”
- Messaggio “certificate”
  - Inviato solo se richiesto
  - Se non si dispone di certificato viene inviato un alert del tipo “no_certificate”
- Messaggio “client_key_exchange”
  - Contiene, a seconda dell’algoritmo di crittografia impiegato:
    - RSA: segreto pre-master di 48 byte criptato con chiave pubblica o temporanea del server
    - Emphemeral o Anonymous D-H: parametri pubblici D-H del client
    - Fixed D-H: i parametri sono già stati inviati e quindi il contenuto del messaggio è nullo
    - Fortezza: parametri Fortezza del client

AUTENTICAZIONE CLIENT E SCAMBIO CHIAVI (1/2 !)

- Ricevuto un “server_done” il client deve:
  - verificare la val
  - verificare i para
- Messaggio “certificat
  - Inviato solo se
  - Se non si dispone
- Messaggio “client_key_exchange”
  - Contiene, a seconda dell’algo mo di crittografia impiegato:
    - RSA: segreto pre-master di 48 byte criptato con chiave pubblica o temporanea del server
    - Emphemeral o Anonymous D-H: parametri pubblici D-H del client
    - Fixed D-H: i parametri sono già stati inviati e quindi il contenuto del messaggio è nullo
    - Fortezza: parametri Fortezza del client

pre_master_secret decifrabile!
Attacchi chosen-ciphertext basati su oracolo
(messaggi di errore, tempo, …)
Primo: Bleichenbacher (1998), recente: Bardou, et al.:
Efficient Padding Oracle Attacks on Cryptographic Hardware. (2012)

AUTENTICAZIONE CLIENT E SCAMBIO CHIAVI (2/2)

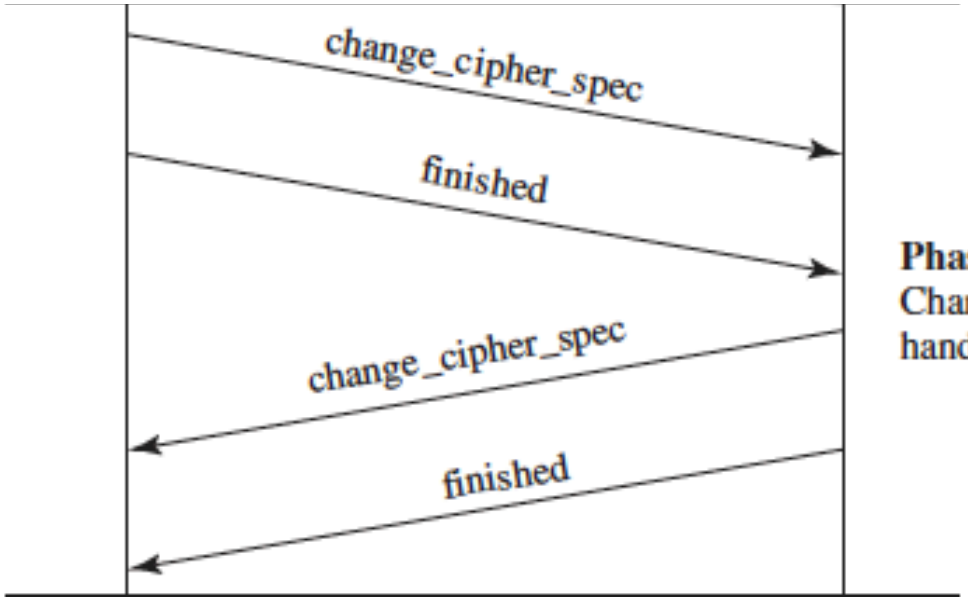
- Messaggio “certificate_verify”
  - Inviato per fornire una *verifica esplicita* di un certificato del client
  - Firma un codice hash sulla base dei messaggi precedenti

MD5(master_secret || pad_2 || MD5(handshake_messages || master_secret || pad_1)) ;

SHA(master_secret || pad_2 || SHA(handshake_messages || master_secret || pad_1)) ;

- Chiave privata utente DSS:
  - si utilizza il codice hash SHA-1
- Chiave privata utente RSA:
  - si utilizza concatenamento dei codici hash MD5 e SHA-1

IL PROTOCOLLO DI HANDSHAKE: FASE 4



Phase 4
Change cipher suite and finish handshake protocol.

FINE

- Il client:
  - invia un messaggio di “change_cipher_spec”, copiando il CipherSpec temporaneo nel CipherSpec corrente
  - invia il messaggio “finished” con i nuovi algoritmi, le nuove chiavi ed i nuovi segreti negoziati
  - verifica che i processi di scambio delle chiavi e autenticazione siano andati a buon fine
- Il contenuto del messaggio “finished” è il concatenamento di due valori hash:
  1. *MD5(master_secret || pad_2 || MD5(handshake_messages || sender || master_secret || pad_1))*
  2. *SHA(master_secret || pad_2 || SHA(handshake_messages || sender || master_secret || pad_1))*
     - “sender”:
       - un codice per identificare se il mittente è il client o il server:
         client: 0x434C4E54; server: 0x53525652
     - “handshake_messages”: dati associati ai messaggi di handshake scambiati tra client e server (messaggio corrente escluso)
- Il server:
  - invia il proprio messaggio “change_cipher_spec”
  - trasferisce il CipherSpec provvisorio in quello corrente
  - invia il proprio messaggio “finished”

FINE (!)

- Il client:
  - invia un messaggio di "change_cipher_spec", copiando il CipherSpec temporaneo nel CipherSpec corrente
  - invia il messaggio "finished" con i nuovi algori
  - verifica che i processi di scambio delle chiavi
- Il contenuto del messaggio "finished" è il concaten
  1. *MD5(master_secret || pad_2 || MD5(handshake_messages || sender || master_secret || pad_1))*
  2. *SHA(master_secret || pad_2 || SHA(handshake_messages || sender || master_secret || pad_1))*
     - "sender":
       - un codice per identificare se il mittente è il client o il server:
         client: 0x434C4E54; server: 0x53525652
     - "handshake_messages": dati associati ai messaggi di handshake scambiati tra client e server (messaggio corrente escluso)
- Il server:
  - invia il proprio messaggio "change_cipher_spec"
  - trasferisce il CipherSpec provvisorio in quello corrente
  - invia il proprio messaggio "finished"

Change_chipher_spec drop attack, Wagner and Schneier (1996)
Risolto in TLS1.0

FINE (!!)

- Il client:
  - invia un messaggio di “change_chiper_spec”, copiando il CipherSpec temporaneo nel CipherSpec corrente
  - invia il messaggio “finished” con i nuovi algoritmi, le nuove chiavi ed i nuovi segreti negoziati
  - verifica che i processi di scambio delle chiavi e autenticazione siano andati a buon fine
- Il contenuto del messaggio “finished” è il concatenamento di due valori hash:
  1. *MD5(master_secret || pad_2 || MD5(handshake_messages || sender || master_secret || pad_1))*
  2. *SHA(master_secret || pad_2 || SHA(handshake_messages || sender || master_secret || pad_1))*
     - “sender”:
       - un codice per identificare se il mittente è il client o il server:
         client: 0x434C4E54; server: 0x53525652
     - “handshake_messages”: dati associati ai messaggi di handshake scambiati tra client e server (messaggio corrente escluso)
- Il server:
  - invia il proprio messaggio “change_cipher_spec”
  - trasferisce il CipherSpec provvisorio in quello c
  - invia il proprio messaggio “finished”

cipher-suite rollback attack,
Wagner and Schneier (1996)
Risolto in SSL3.0