

Protocolli SSL e TLS

Alfredo De Santis

Dipartimento di Informatica
Università degli Studi di Salerno

ads@unisa.it



Maggio 2020

Outline

- Caratteristiche ed Ambiti di Utilizzo
- Storia e Versioning
- Componenti
- OpenSSL s_client ed s_server
- SSL Server Test ed SSL Client Test

Outline

- **Caratteristiche ed Ambiti di Utilizzo**
- Storia e Versioning
- Componenti
- OpenSSL s_client ed s_server
- SSL Server Test ed SSL Client Test

Caratteristiche di SSL/TLS

Definizioni

- **SSL** = Secure Socket Layer
 - Socket = Concetto di UNIX per network API
- **TLS** = Transport Layer Security
- **DTLS** = Datagram Transport Layer Security



Caratteristiche di SSL/TLS

Definizioni

- Offrono meccanismi di sicurezza alle applicazioni che usano il protocollo TCP (o UDP)
- Sono in grado di rendere sicuri numerosi protocolli che operano a livello applicativo
 - HTTP
 - POP3
 - IMAP
 - SMTP
 - E molto altro ancora...



Caratteristiche di SSL/TLS

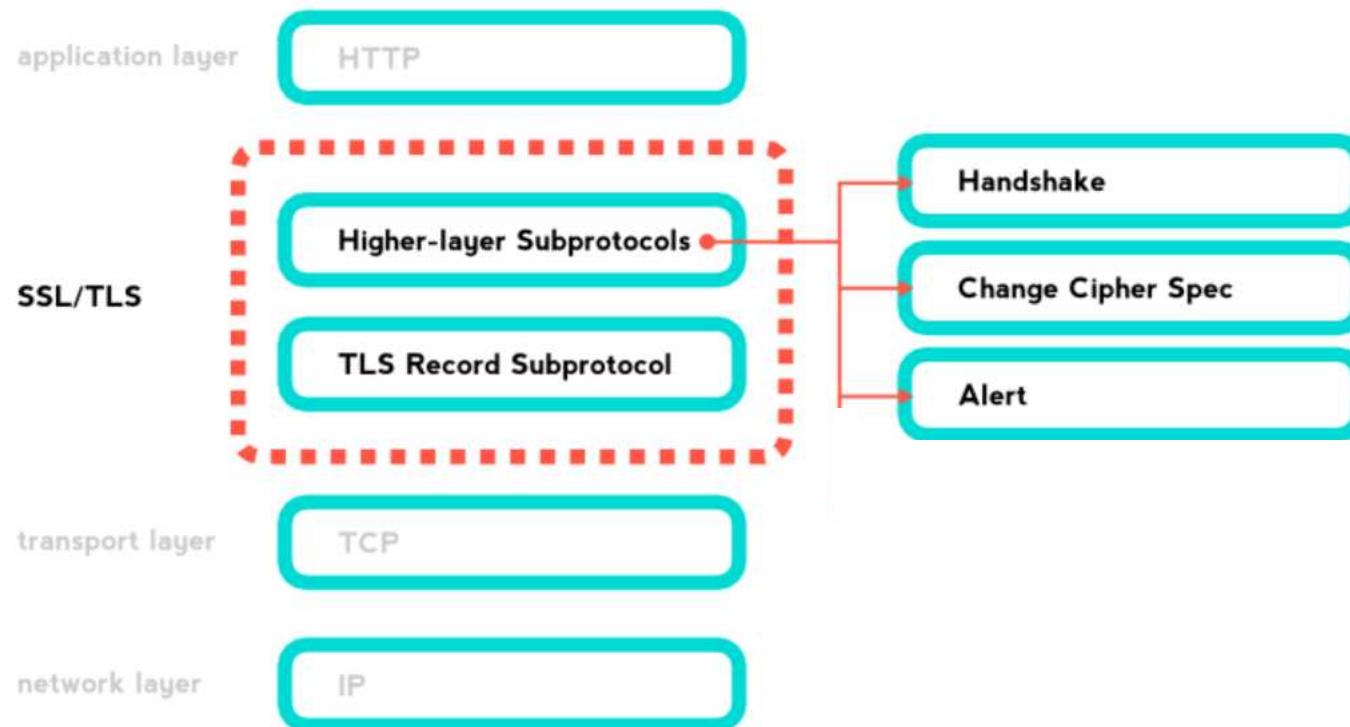
Protocolli per la Sicurezza in Internet

Communication layers	Security protocols
Application layer	SSH, S/MIME, Kerberos, PGP, WSS, etc
→ <u>Transport layer</u>	<u>SSL/TLS</u>
Network layer	IPSec
Data Link layer	IEEE 802.1X, IEEE 802.11i (WPA2), etc
Physical layer	Quantum Cryptography



Caratteristiche di SSL/TLS

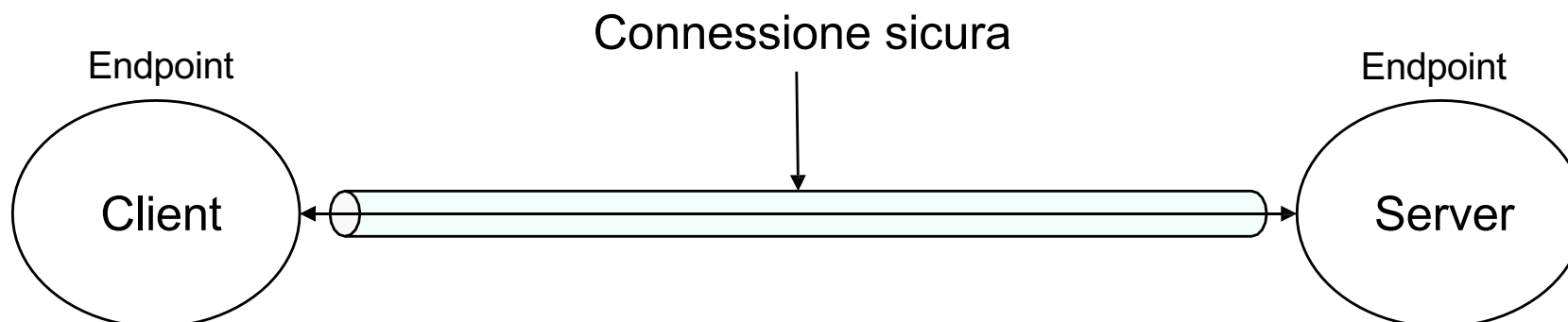
Dove Operano



Caratteristiche di SSL/TLS

Principali Funzionalità Offerte

- Fornisce autenticazione per applicazioni Server/Client
- Cifra i dati prima di inviarli su un canale pubblico (confidenzialità)
- Garantisce l'integrità dell'informazione
- Le primitive crittografiche (ma anche altre informazioni) vengono negoziare tra le parti (Client e Server)



Caratteristiche di SSL/TLS

Connessione vs. Sessione

➤ Connessione

- Definisce un canale di comunicazione tra due endpoint
 - Client e Server
- Ciascuna connessione è associata ad una sessione

➤ Sessione

- Definisce un insieme di parametri crittografici, negoziati tra le parti, che possono «sopravvivere» tra diverse connessioni
- Usata per evitare la (costosa) negoziazione di nuovi parametri di sicurezza per ciascuna connessione

Caratteristiche di SSL/TLS

Connessione vs. Sessione

- È possibile chiudere la connessione mantenendo attiva la sessione
 - I parametri di sessione potrebbero essere memorizzati e tale sessione potrebbe essere successivamente riutilizzata da un'altra connessione
 - La sessione potrebbe anche appartenere ad un processo completamente diverso (ad es., dopo il riavvio del sistema)
 - N.B. la sessione archiviata deve essere conservata sia dal Client che dal Server
- È anche possibile rinegoziare i parametri SSL/TLS e creare una sessione completamente nuova, senza interrompere la connessione



Ambiti di Utilizzo di SSL/TLS



➤ **Commercio elettronico**

- Ordinanze: le form con cui si ordina un prodotto vengono inviate usando SSL/TLS
 - HTTPS
- Pagamenti: quando viene inserito un numero di carta di credito, l'invio dei dati avviene usando SSL/TLS
 - HTTPS

➤ **Accesso sicuro ad informazioni**

- La consultazione di informazioni accessibili solo da utenti «qualificati»
- L'invio di password o altri dati riservati

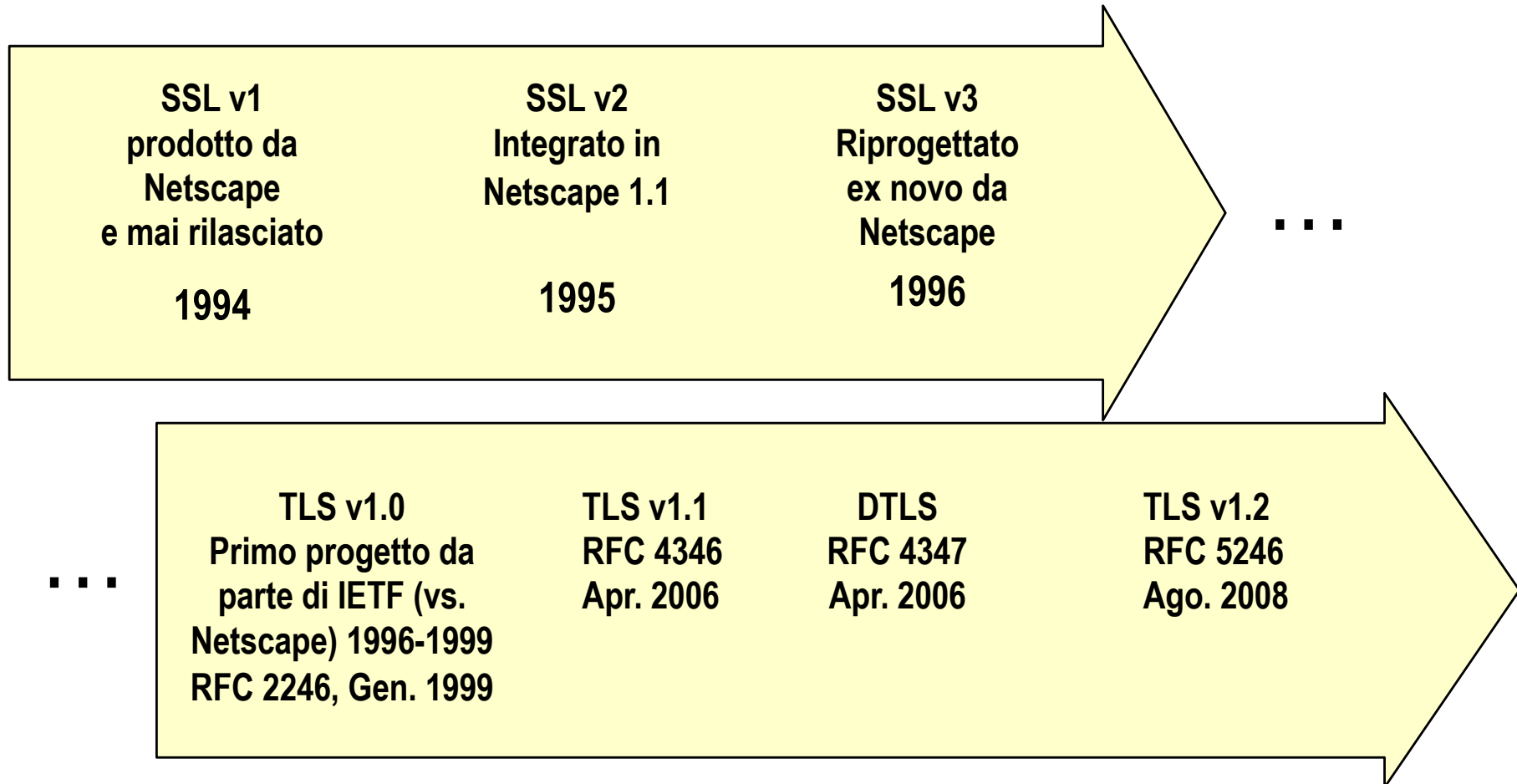


➤ **E molto altro ancora...**

Outline

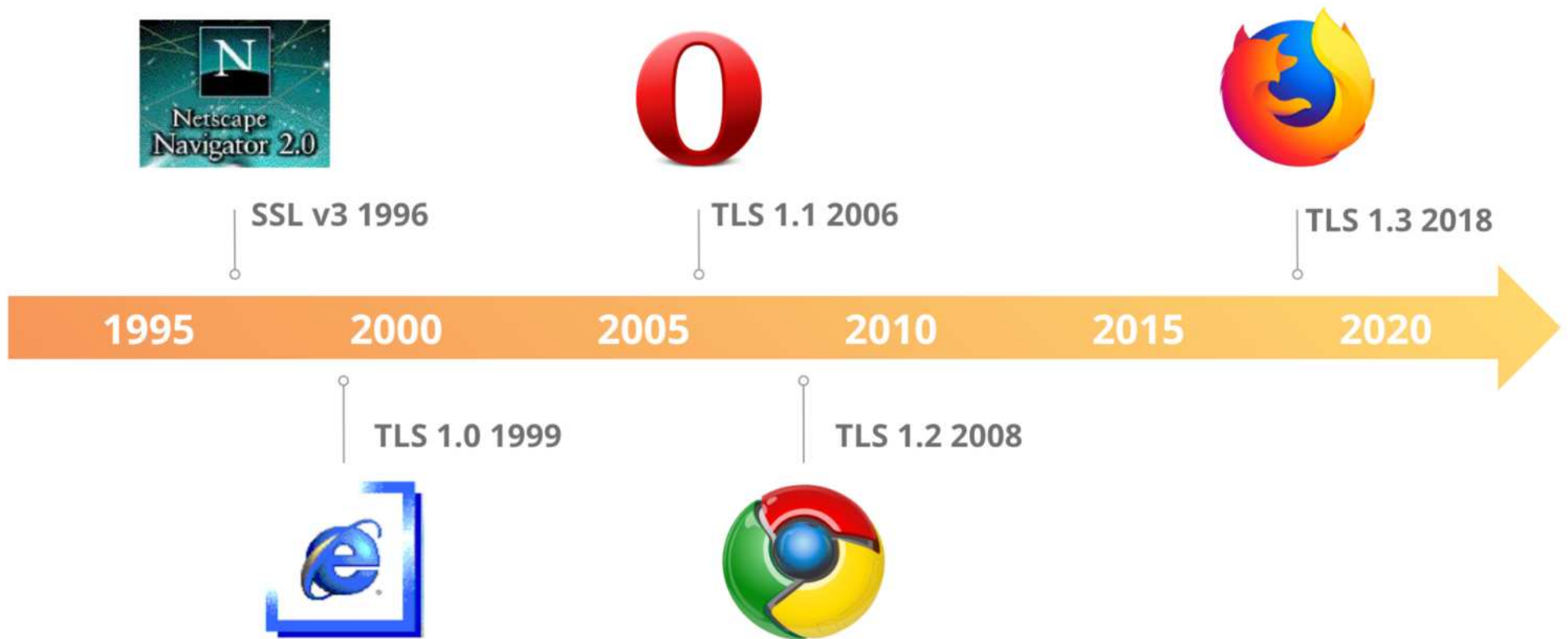
- Caratteristiche ed Ambiti di Utilizzo
- **Storia e Versioning**
- Componenti
- OpenSSL s_client ed s_server
- SSL Server Test ed SSL Client Test

Storia di SSL/TLS



TLS v1.3 approvato a fine marzo 2018 e definito nell'RFC 8446!

Storia di SSL/TLS



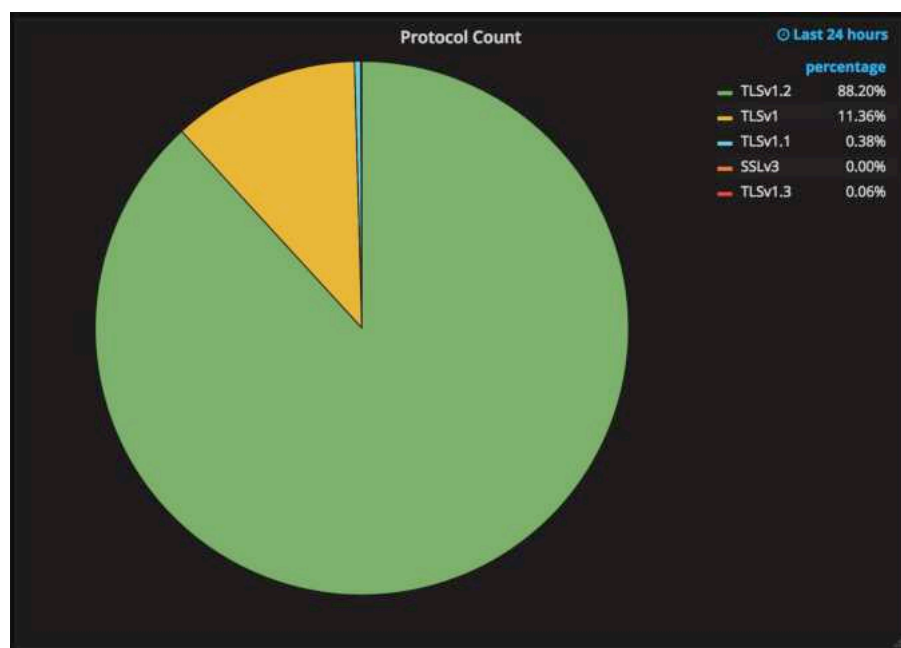
TLS Versione 1.3 (RFC 8446)

Caratteristiche Principali

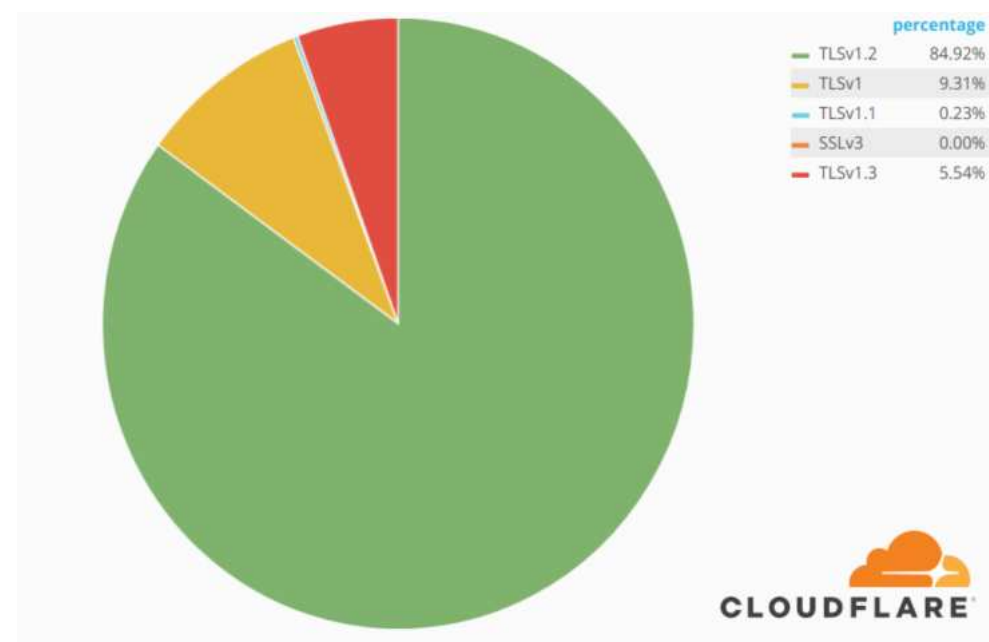
- Basato su TLS v1.2 (RFC 5246)
- Principali differenze rispetto a TLS v1.2
 - Rimozione del supporto per molte funzioni non sicure o obsolete
 - Curve ellittiche note come «deboli» o meno utilizzate
 - Funzioni hash crittografiche MD5 e SHA-224
 - Protocollo SSL e cifrario RC4 (che erano ancora supportati per retrocompatibilità)
 - Aggiunta di nuovi algoritmi e protocolli
 - Stream cipher ChaCha20 e algoritmo Poly1305 per il Message Authentication Code (MAC)
 - Algoritmi di firma digitale Ed25519 ed Ed448
 - Protocolli di scambio chiavi x25519 ed x448

TLS Versioning Statistiche

Dicembre 2017



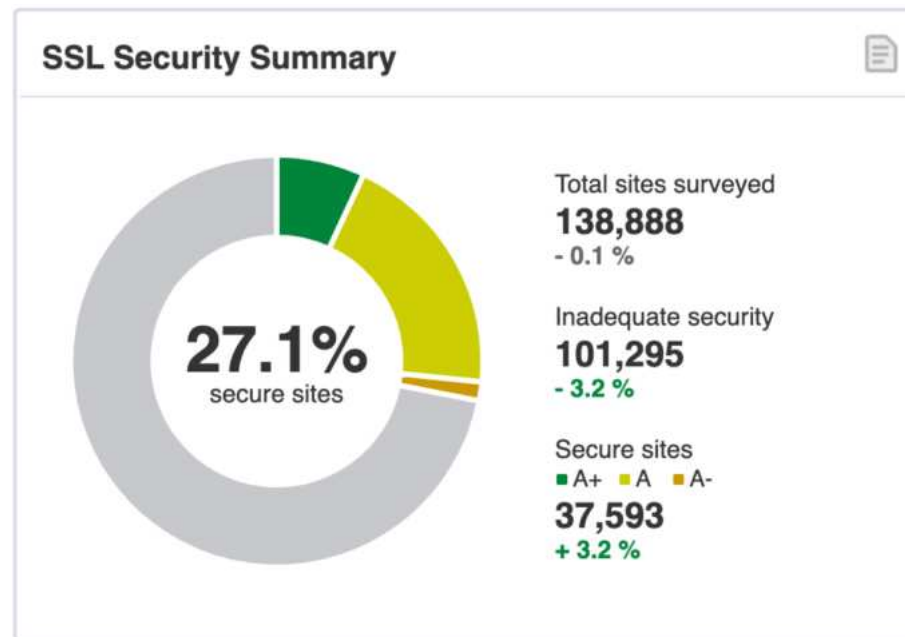
Maggio 2018



TLS Versioning

Statistiche - SSL Pulse

«SSL Pulse is a continuous and global dashboard for monitoring the quality of SSL/TLS support over time across 150,000 SSL- and TLS-enabled websites, based on Alexa's list of the most popular sites in the world»



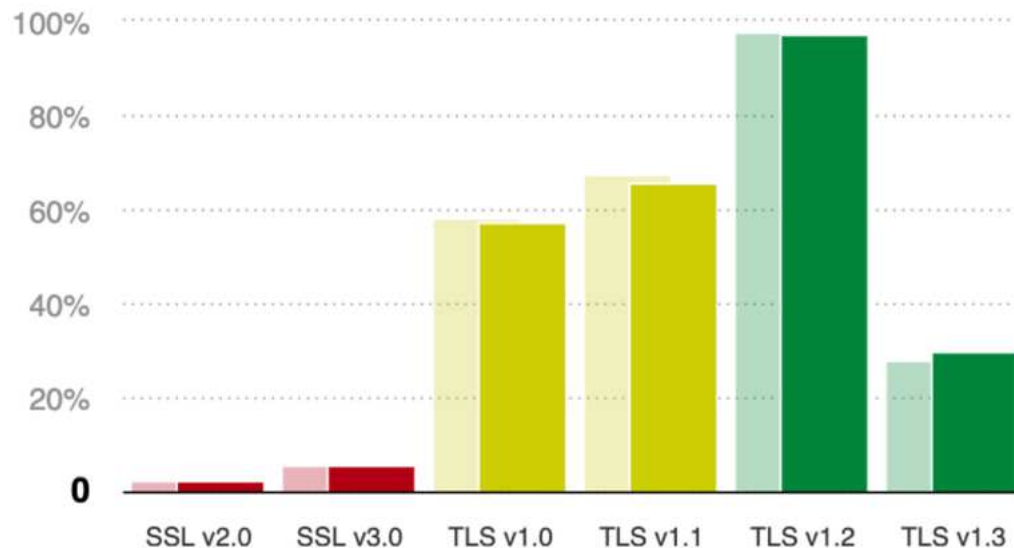
18 maggio 2020

TLS Versioning

Statistiche - SSL Pulse

«SSL Pulse is a continuous and global dashboard for monitoring the quality of SSL/TLS support over time across 150,000 SSL- and TLS-enabled websites, based on Alexa's list of the most popular sites in the world»

Protocol Support



18 maggio 2020

TLS Versione 1.3

Chrome

- Google Chrome ha abilitato di default il supporto a TLS v1.3 a partire dal 2017
- Tale supporto di default è stato poi rimosso a causa di problemi di retrocompatibilità con alcuni web proxy
- A partire dalla release 65, Google Chrome (marzo 2018) ha di nuovo abilitato di default il supporto a TLS v1.3



TLS Versione 1.3

Chrome

Chrome | chrome://flags

tls

Reset all to default

Experiments

81.0.4044.122

Available

Unavailable

TLS 1.3 hardening for local anchors

This option enables the TLS 1.3 downgrade hardening mechanism for connections authenticated by local trust anchors. This improves security for connections to TLS-1.3-capable servers while remaining compatible with older servers. Firewalls and proxies that do not function when this is enabled do not implement TLS 1.2 correctly or securely and must be updated. – Mac, Windows, Linux, Chrome OS, Android

[#tls13-hardening-for-local-anchors](#)

Default

TLS 1.3 Early Data

This option enables TLS 1.3 Early Data, allowing GET requests to be sent during the handshake when resuming a connection to a compatible TLS 1.3 server. – Mac, Windows, Linux, Chrome OS, Android

[#enable-tls13-early-data](#)

Default

Show security warnings for sites using legacy TLS versions

Show security warnings for sites that use legacy TLS versions (TLS 1.0 and TLS 1.1), which are deprecated and will be removed in the future. – Mac, Windows, Linux, Chrome OS

[#show-legacy-tls-warnings](#)

Default

Enforce deprecation of legacy TLS versions

Enable connection errors and interstitials for sites that use legacy TLS versions (TLS 1.0 and TLS 1.1), which are deprecated and will be removed in the future. – Mac, Windows, Linux, Chrome OS, Android

[#legacy-tls-enforced](#)

Default



TLS Versione 1.3

Mozilla

- Network Security Services (NSS)
 - Libreria crittografica sviluppata da Mozilla ed usata dal suo web browser Firefox
 - Fornisce il supporto a TLS v1.3 a partire da febbraio 2017



TLS Versione 1.3

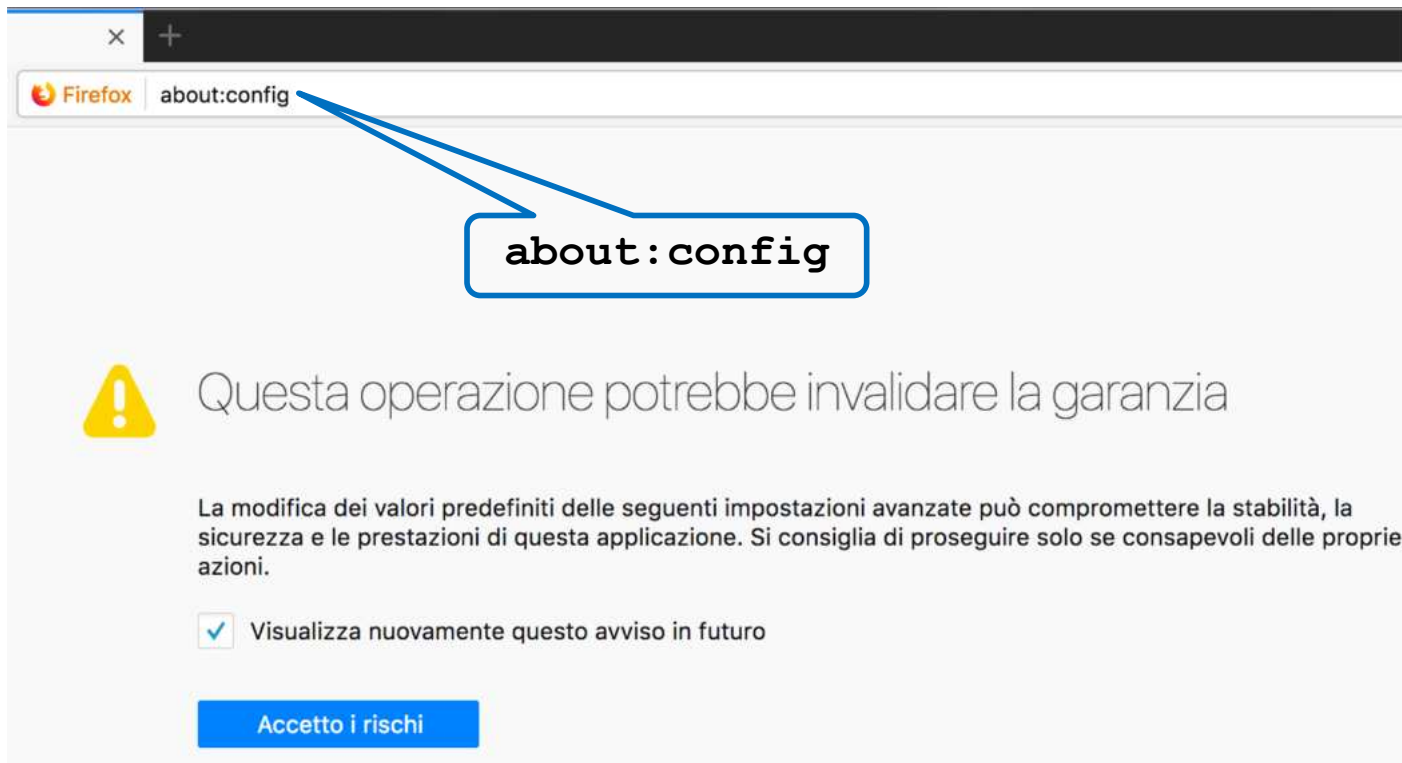
Mozilla

- TLS 1.3
 - Aggiunto a Firefox 52.0 (marzo 2017) e poco dopo disabilitato di default per problemi di retrocompatibilità con alcuni siti e plugin di Firefox
 - Nuovamente abilitato di default a partire da Firefox 60.0
 - Firefox 63 (ottobre 2018) supporta la versione finale di TLS v1.3



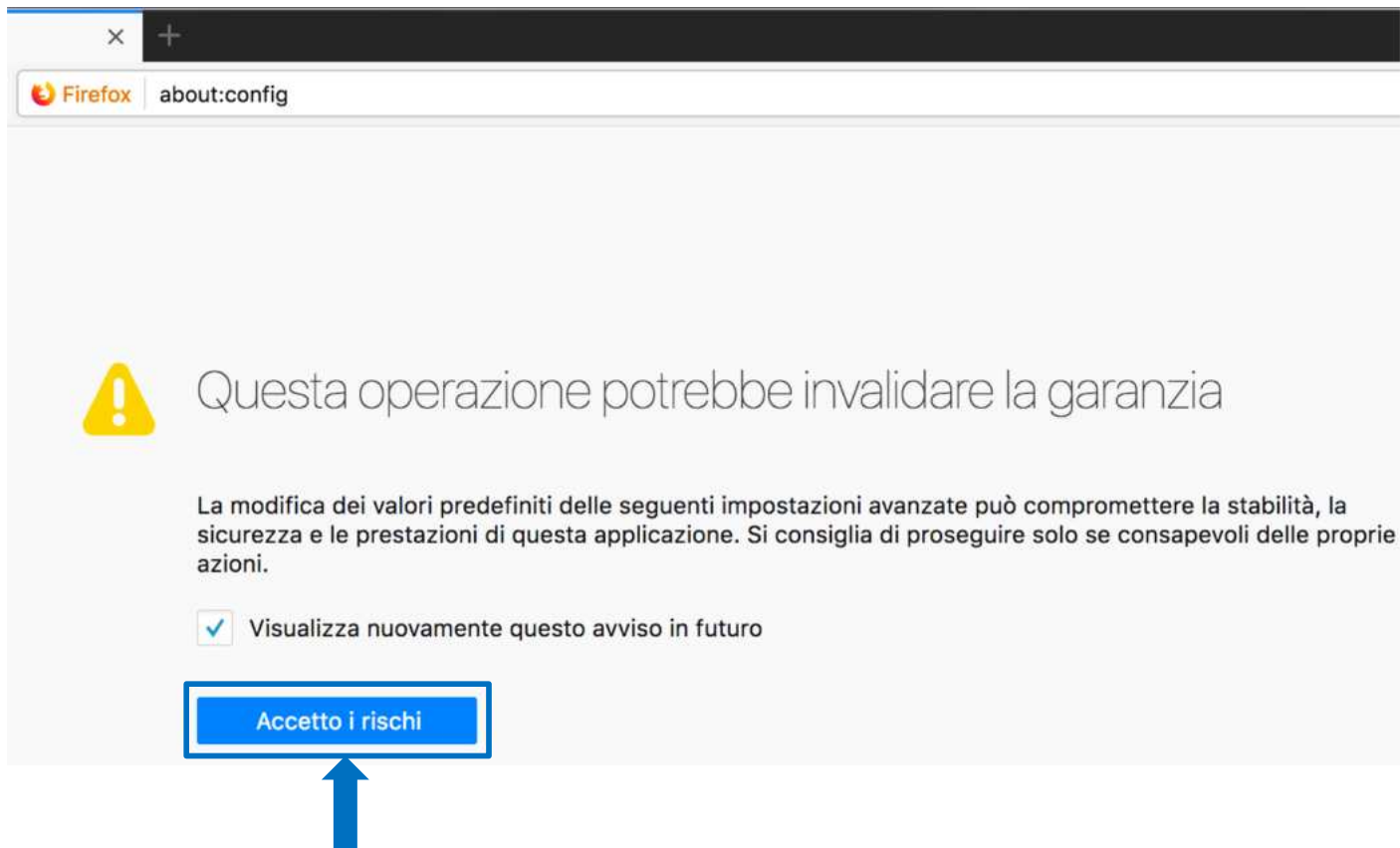
TLS Versione 1.3

Mozilla



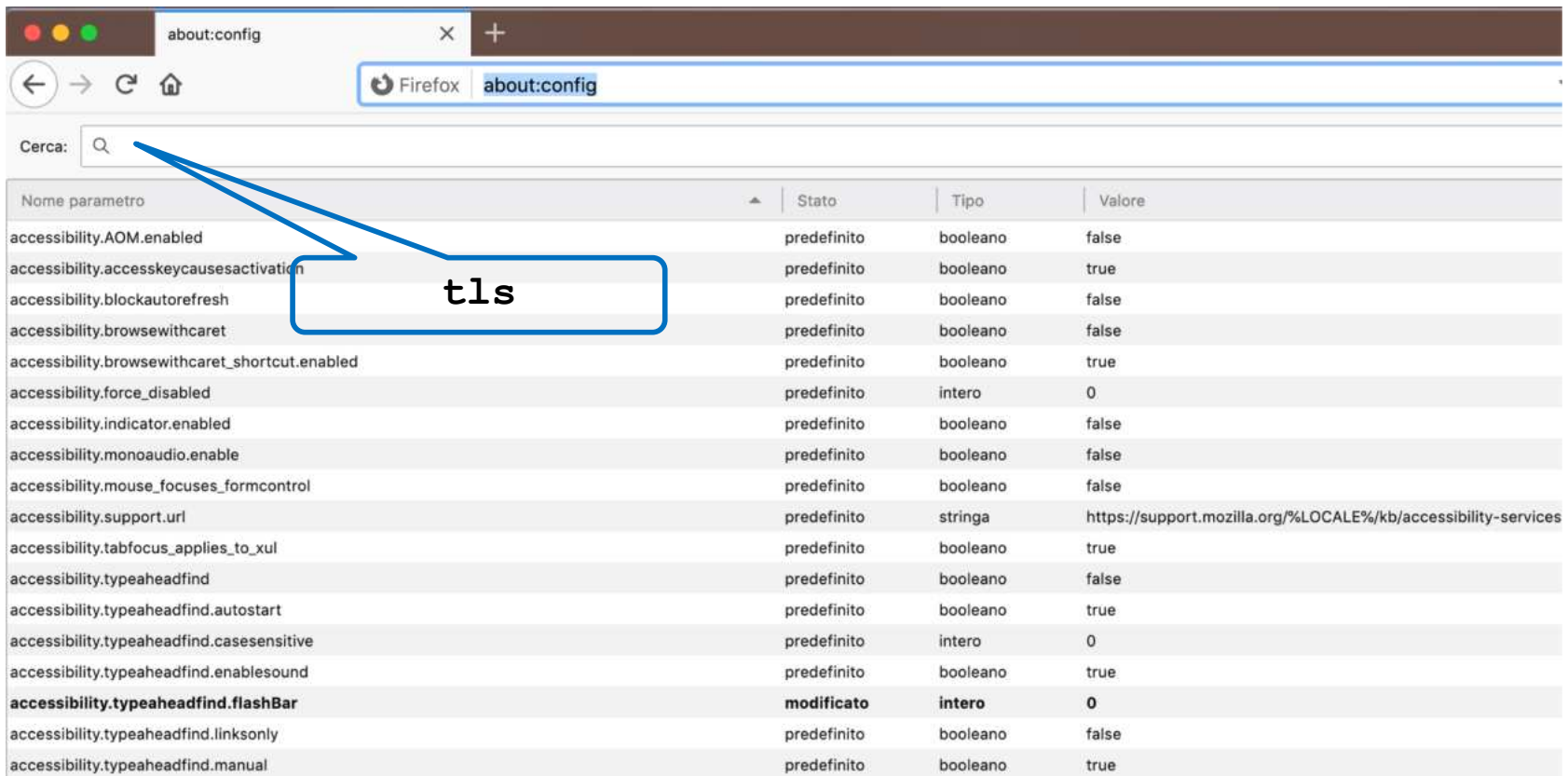
TLS Versione 1.3

Mozilla



TLS Versione 1.3

Mozilla



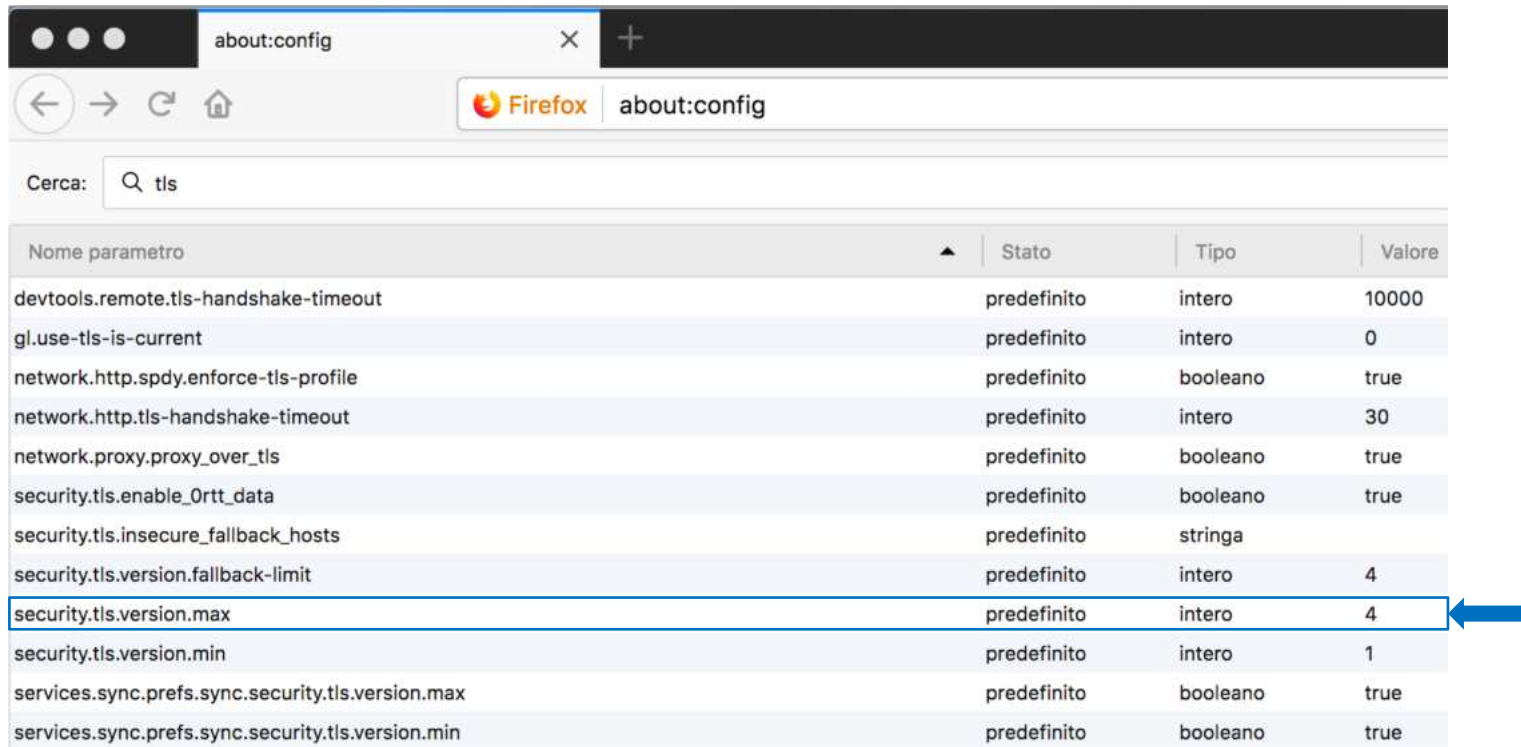
Cerca:

Nome parametro	Stato	Tipo	Valore
accessibility.AOM.enabled	predefinito	booleano	false
accessibility.accesskeycausesactivation	predefinito	booleano	true
accessibility.blockautorefresh	predefinito	booleano	false
accessibility.browsewithcaret	predefinito	booleano	false
accessibility.browsewithcaret_shortcut.enabled	predefinito	booleano	true
accessibility.force_disabled	predefinito	intero	0
accessibility.indicator.enabled	predefinito	booleano	false
accessibility.monoaudio.enable	predefinito	booleano	false
accessibility.mouse_focuses_formcontrol	predefinito	booleano	false
accessibility.support.url	predefinito	stringa	https://support.mozilla.org/%LOCALE%/kb/accessibility-services
accessibility.tabfocus_applies_to_xul	predefinito	booleano	true
accessibility.typeaheadfind	predefinito	booleano	false
accessibility.typeaheadfind.autostart	predefinito	booleano	true
accessibility.typeaheadfind.casesensitive	predefinito	intero	0
accessibility.typeaheadfind.enablestound	predefinito	booleano	true
accessibility.typeaheadfind.flashBar	modificato	intero	0
accessibility.typeaheadfind.linkonly	predefinito	booleano	false
accessibility.typeaheadfind.manual	predefinito	booleano	true



TLS Versione 1.3

Mozilla



The screenshot shows the Firefox 'about:config' page with the search bar set to 'tls'. A table lists various TLS-related configuration parameters. The row for 'security.tls.version.max' is highlighted with a blue border and a blue arrow pointing to its value '4'.

Nome parametro	Stato	Tipo	Valore
devtools.remote.tls-handshake-timeout	predefinito	intero	10000
gl.use-tls-is-current	predefinito	intero	0
network.http.spdy.enforce-tls-profile	predefinito	booleano	true
network.http.tls-handshake-timeout	predefinito	intero	30
network.proxy.proxy_over_tls	predefinito	booleano	true
security.tls.enable_Ortt_data	predefinito	booleano	true
security.tls.insecure_fallback_hosts	predefinito	stringa	
security.tls.version.fallback-limit	predefinito	intero	4
security.tls.version.max	predefinito	intero	4
security.tls.version.min	predefinito	intero	1
services.sync.prefs.sync.security.tls.version.max	predefinito	booleano	true
services.sync.prefs.sync.security.tls.version.min	predefinito	booleano	true



TLS Versione 1.3

Mozilla

Nome parametro	Stato	Tipo	Valore
security.tls.version.max		predefinito	intero 4

- **Valore 0:** SSL 3.0 è il massimo supportato
- **Valore 1:** TLS 1.0 è il massimo supportato
- **Valore 2:** TLS 1.1 è il massimo supportato
- **Valore 3:** TLS 1.2 è il massimo supportato
- **Valore 4:** TLS 1.3 è il massimo supportato



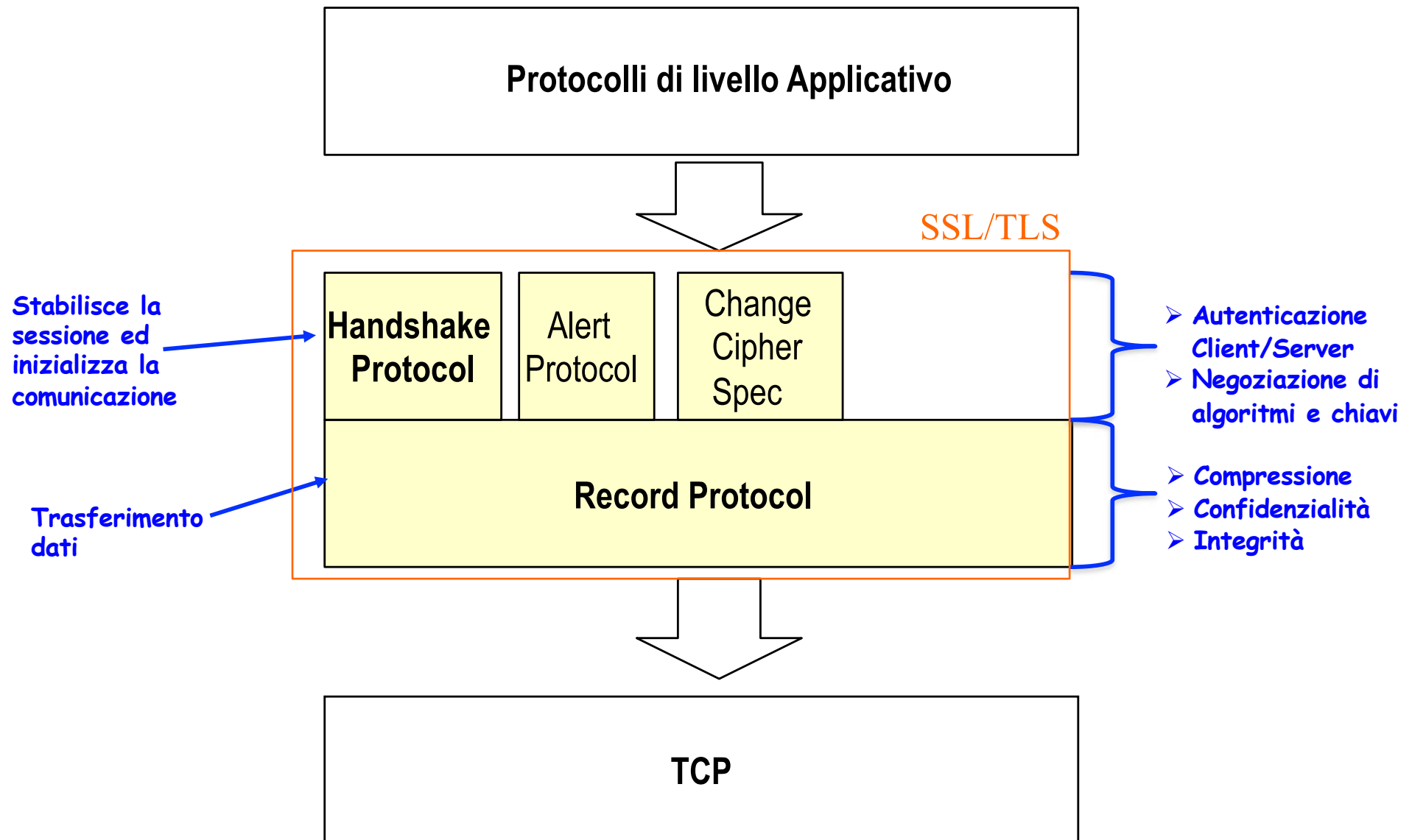
Outline

- Caratteristiche e Ambiti di Utilizzo
- Storia e Versioning
- **Componenti**
- OpenSSL s_client ed s_server
- SSL Server Test ed SSL Client Test

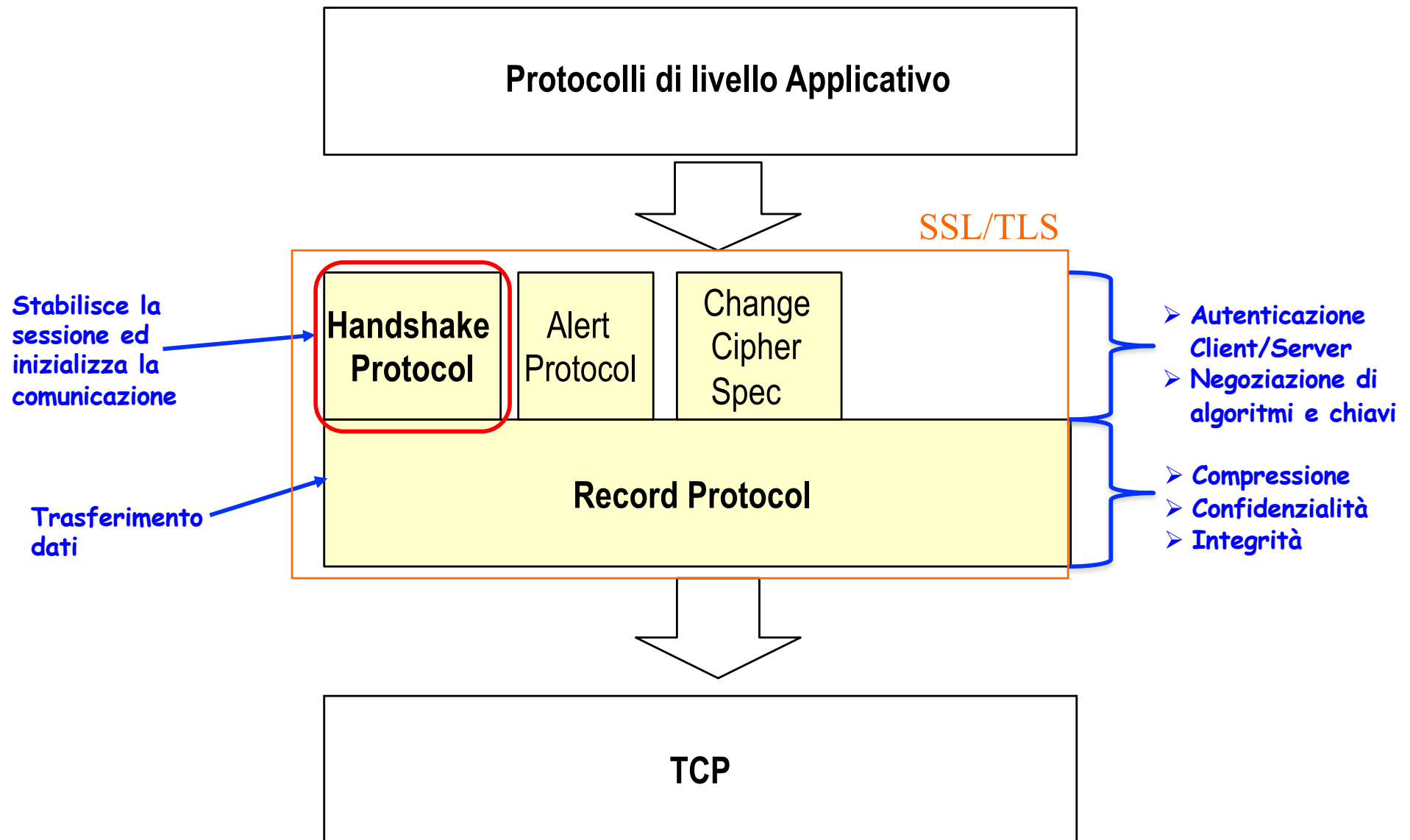
Componenti di SSL/TLS

- **SSL/TLS è composto da quattro protocolli**
 - **Handshake Protocol**
 - Permette alle parti di negoziare i diversi algoritmi (ad es., primitive crittografiche) e parametri necessari per la sicurezza della comunicazione
 - Consente l'eventuale autenticazione tra le parti
 - **Record Protocol**
 - Utilizza algoritmi e parametri negoziati durante l'Handshake Protocol
 - Si occupa della compressione, del MAC e della cifratura
 - **Change Cipher Spec Protocol**
 - Impone l'esecuzione di un nuovo handshake per rinegoziare i parametri di sicurezza e ripetere l'autenticazione
 - **Alert Protocol**
 - Notifica situazioni anomale o segnala eventuali problemi

SSL/TLS Protocol Stack



SSL/TLS Protocol Stack



Handshake Protocol

- Utilizzato dalle due parti (endpoint)
 - Client e Server
- Permette alle parti di
 - Negoziare la versione del protocollo SSL/TLS e l'insieme delle primitive crittografiche da utilizzare (Ciphersuite)
 - Garantendo interoperabilità tra le diverse implementazioni del protocollo
 - Stabilire le chiavi crittografiche da utilizzare
 - Autenticare il Server (ed opzionalmente il Client)



Handshake Protocol

Ciphersuite

- Una Ciphersuite definisce
 - **Schema per l'accordo/scambio di chiavi:** stabilisce il modo in cui verranno scambiate le chiavi utilizzate dagli algoritmi a chiave simmetrica
 - **Schema per l'autenticazione:** stabilisce in che modo verrà eseguita l'autenticazione del Server e (se necessario) quella del Client
 - **Schema per la cifratura simmetrica:** stabilisce quale algoritmo a chiave simmetrica verrà utilizzato per cifrare i dati
 - **Schema per l'autenticazione del messaggio (MAC):** stabilisce il metodo che la sessione utilizzerà per il controllo di integrità dei dati



Handshake Protocol

Ciphersuite - Esempio

- Algoritmi per lo scambio di chiavi: RSA, DH, ECDH, etc
- Algoritmi per l'autenticazione: RSA, DSA, ECDSA, etc
- Algoritmi per la cifratura: AES, 3DES, CAMELLIA, etc
- Algoritmi per il calcolo del MAC: SHA, MD5, etc

Esempio di una tipica Ciphersuite

- **TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384**
 - **TLS** indica il protocollo
 - **ECDH** indica l'algoritmo per lo scambio di chiavi
 - **ECDSA** indica l'algoritmo per l'autenticazione
 - **AES_256_CBC** indica l'algoritmo per la cifratura
 - **SHA384** indica l'algoritmo per il MAC



Handshake Protocol

Client

Server

ClientHello

ServerHello
Certificate*
ServerKeyExchange*
CertificateRequest*
ServerHelloDone

Certificate*
ClientKeyExchange
CertificateVerify*
ChangeCipherSpec
Finished

ChangeCipherSpec
Finished

Application Data

Application Data

*messaggio opzionale



Handshake Protocol

Client

Server

ClientHello

ServerHello
Certificate*
ServerKeyExchange*
CertificateRequest*
ServerHelloDone

Certificate*
ClientKeyExchange
CertificateVerify*
ChangeCipherSpec
Finished

ChangeCipherSpec
Finished

Application Data

Application Data

*messaggio opzionale



Handshake Protocol

ClientHello

- **ClientHello:** Messaggio iniziale inviato dal Client al Server, composto dai seguenti campi
 - **client_version:** versione di TLS più recente supportata dal Client
 - **random:** numero di 4 byte che specifica data e ora del Client + numero casuale di 28 byte generato dal Client
 - **session_id:** permette al Client di proporre al Server di riutilizzare una sessione precedentemente stabilita
 - Valore diverso da zero per una nuova connessione sulla sessione corrente
 - Valore uguale a zero per una nuova connessione su una nuova sessione
 - **cipher_suites:** lista di Ciphersuite supportate dal Client
 - **compression_methods:** lista degli algoritmi di compressione supportati dal Client



Handshake Protocol

Client

Server

ClientHello

ServerHello

Certificate*

ServerKeyExchange*

CertificateRequest*

ServerHelloDone

Certificate*

ClientKeyExchange

CertificateVerify*

ChangeCipherSpec

Finished

ChangeCipherSpec

Finished

Application Data

Application Data

*messaggio opzionale



Handshake Protocol

ServerHello

- **ServerHello:** Messaggio iniziale inviato dal Server al Client, composto dai seguenti campi
 - **server_version:** versione di TLS più recente supportata da ambedue le parti
 - **random:** numero di 4 byte che specifica data e ora del Server + numero casuale di 28 byte generato dal Server
 - **session_id** (se presente)
 - *new session_id*
 - *resumed session_id*, lo stesso ID indicato dal **ClientHello**; indica che il Server è d'accordo nel riutilizzare una vecchia sessione
 - **cipher_suite:** migliore Ciphersuite supportata dalle due parti
 - Se le parti non hanno una ciphersuite in comune, la sessione termina con il messaggio di alert «**handshake failure**»
 - **compression_method:** miglior algoritmo di compressione supportato dalle due parti



Handshake Protocol

Client

Server

ClientHello

ServerHello

Certificate*

ServerKeyExchange*

CertificateRequest*

ServerHelloDone

Certificate*

ClientKeyExchange

CertificateVerify*

ChangeCipherSpec

Finished

ChangeCipherSpec

Finished

Application Data

Application Data

*messaggio opzionale



Handshake Protocol

Messaggi Opzionali dopo ServerHello

➤ **Certificate**

- Contiene il Certificato del Server
 - La chiave pubblica inclusa nel certificato è utilizzata dal Client per autenticare il Server e per cifrare il Pre-master secret

➤ **ServerKeyExchange**

- Messaggio usato dal Server per spedire una chiave temporanea al Client
 - Inviato solo se il certificato del Server non contiene informazioni tali da permettere al Client di spedire il Pre-master secret in modo sicuro

➤ **CertificateRequest**

- Messaggio tramite cui il Server richiede di autenticare il Client
 - Utilizzato, ad es., dai siti web delle banche, in cui il Server deve verificare l'identità del Client prima di rilasciare informazioni sensibili

Handshake Protocol

Messaggi Opzionali dopo ServerHello

➤ Certificate

➤ Contiene il Certificato del Server

- La chiave pubblica inclusa nel certificato è utilizzata dal Client per autenticare il Server e per cifrare il **Pre-master secret**

Numero casuale generato dal Client che verrà utilizzato dalle due parti per la generazione della chiave di sessione

per spedire una chiave temporanea al

- Inviato solo se il certificato del Server non contiene informazioni tali da permettere al Client di spedire il Pre-master secret in modo sicuro

➤ CertificateRequest

➤ Messaggio tramite cui il Server richiede di autenticare il Client

- Utilizzato, ad es., dai siti web delle banche, in cui il Server deve verificare l'identità del Client prima di rilasciare informazioni sensibili

Handshake Protocol

Client

Server

ClientHello

ServerHello
Certificate*
ServerKeyExchange*
CertificateRequest*
ServerHelloDone

Certificate*
ClientKeyExchange
CertificateVerify*
ChangeCipherSpec
Finished

ChangeCipherSpec
Finished

Application Data

Application Data

*messaggio opzionale



Handshake Protocol

ServerHelloDone

- **ServerHelloDone**
 - Indica la fine della fase di «presentazione» del Server



A questo punto, la fase di «saluto» tra le due parti termina!

Handshake Protocol

Client

Server

ClientHello

ServerHello
Certificate*
ServerKeyExchange*
CertificateRequest*
ServerHelloDone

Certificate*
ClientKeyExchange
CertificateVerify*
ChangeCipherSpec
Finished

ChangeCipherSpec
Finished

Application Data

Application Data

*messaggio opzionale



Handshake Protocol

- **Certificate** (opzionale)
 - Viene spedito solo se il Server lo abbia richiesto
- **ClientKeyExchange**
 - Il Pre-master secret viene cifrato con la chiave pubblica del Server (o con la chiave temporanea spedita tramite il messaggio **ServerKeyExchange**) e viene trasmesso a quest'ultimo
- **CertificateVerify** (opzionale)
 - Spedito solo nel caso in cui il Server abbia richiesto l'autenticazione del Client
 - Il Client invia un messaggio contenente la firma di tutti i messaggi precedentemente scambiati
 - In questo modo il Server, verificando la firma con la chiave pubblica presente nel certificato del Client, si accerta dell'identità del Client

Handshake Protocol

- **ChangeCipherSpec**
 - Viene spedito per confermare al Server che tutti i messaggi che seguono saranno protetti usando le chiavi e gli algoritmi (*ciphersuite*) appena negoziati
- **Finished**
 - Primo messaggio «sicuro» (cifrato ed autenticato)
 - Questo messaggio contiene l'hash di quanto scambiato fino a quel momento ed è cifrato usando la chiave di sessione

Handshake Protocol

Client

Server

ClientHello

ServerHello
Certificate*
ServerKeyExchange*
CertificateRequest*
ServerHelloDone

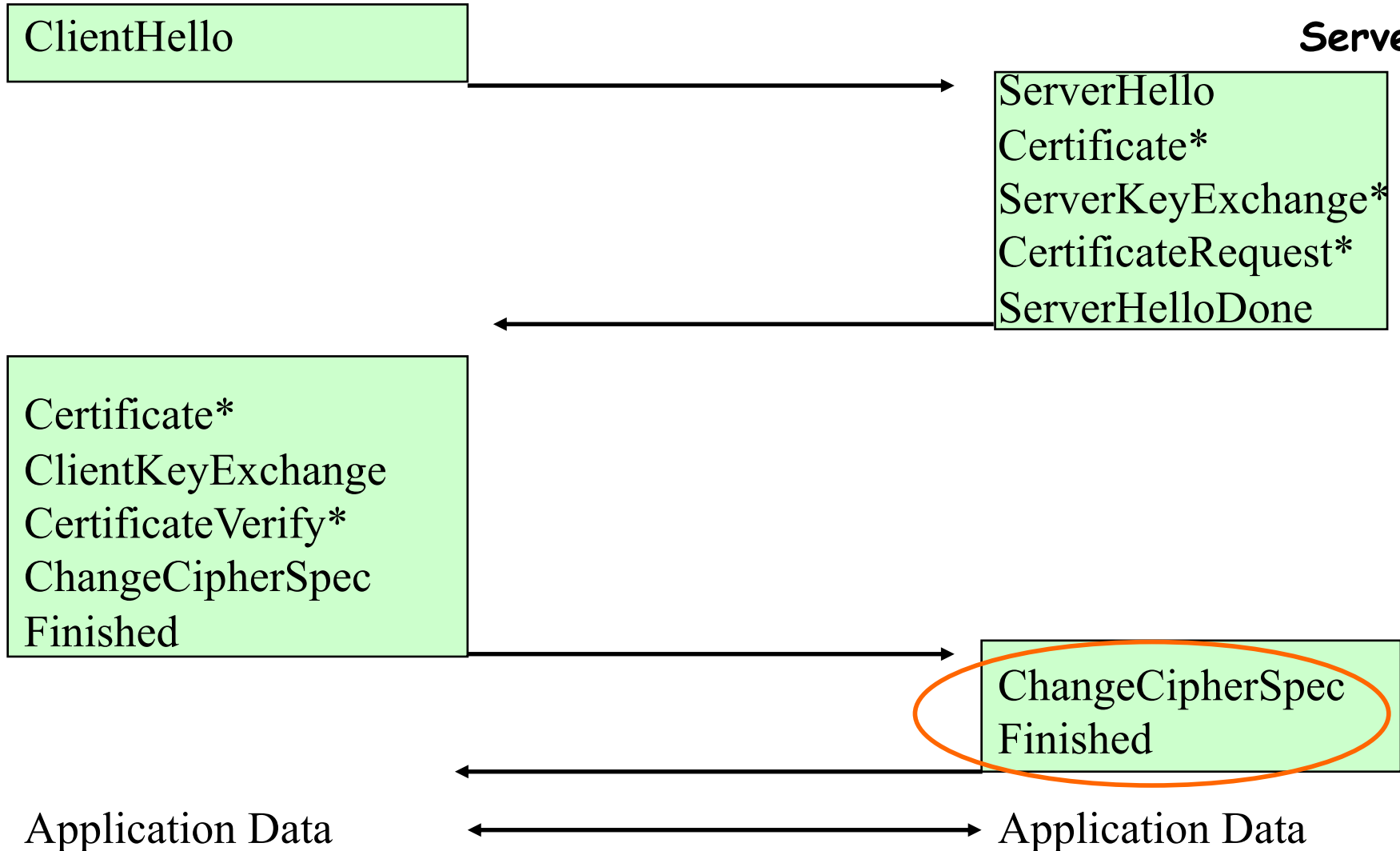
Certificate*
ClientKeyExchange
CertificateVerify*
ChangeCipherSpec
Finished

ChangeCipherSpec
Finished

Application Data

Application Data

*messaggio opzionale



Handshake Protocol

➤ ChangeCipherSpec

- Questo messaggio conferma al Client che tutti i messaggi che seguono verranno protetti usando le chiavi e gli algoritmi (*ciphersuite*) appena negoziati

➤ Finished

- Questo messaggio contiene l'hash di tutti i messaggi scambiati fino a quel momento ed è cifrato usando la chiave di sessione
- Osservazione: Se il Client è in grado di decifrare tale messaggio e di validare gli hash in esso contenuti, allora
 - La fase di handshake SSL/TLS è terminata con successo
 - La chiave di sessione da lui calcolata coincide con quella calcolata dal Server

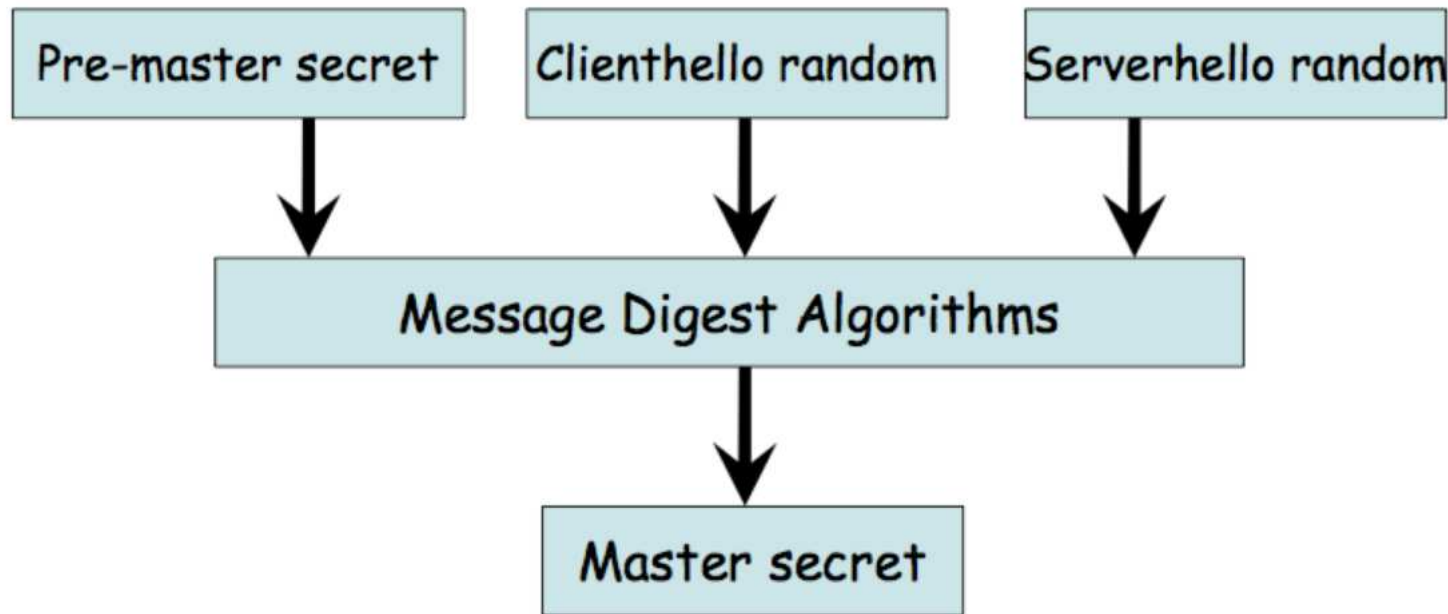
Handshake Protocol

Calcolo della Chiave di Sessione

- L'Handshake Protocol permette alle due parti di calcolare in maniera indipendente la stessa chiave di sessione
- Tale calcolo avviene in due fasi
 - Nella prima fase viene calcolato un Master Secret
 - Nella seconda fase viene calcolata una Chiave di Sessione

Handshake Protocol

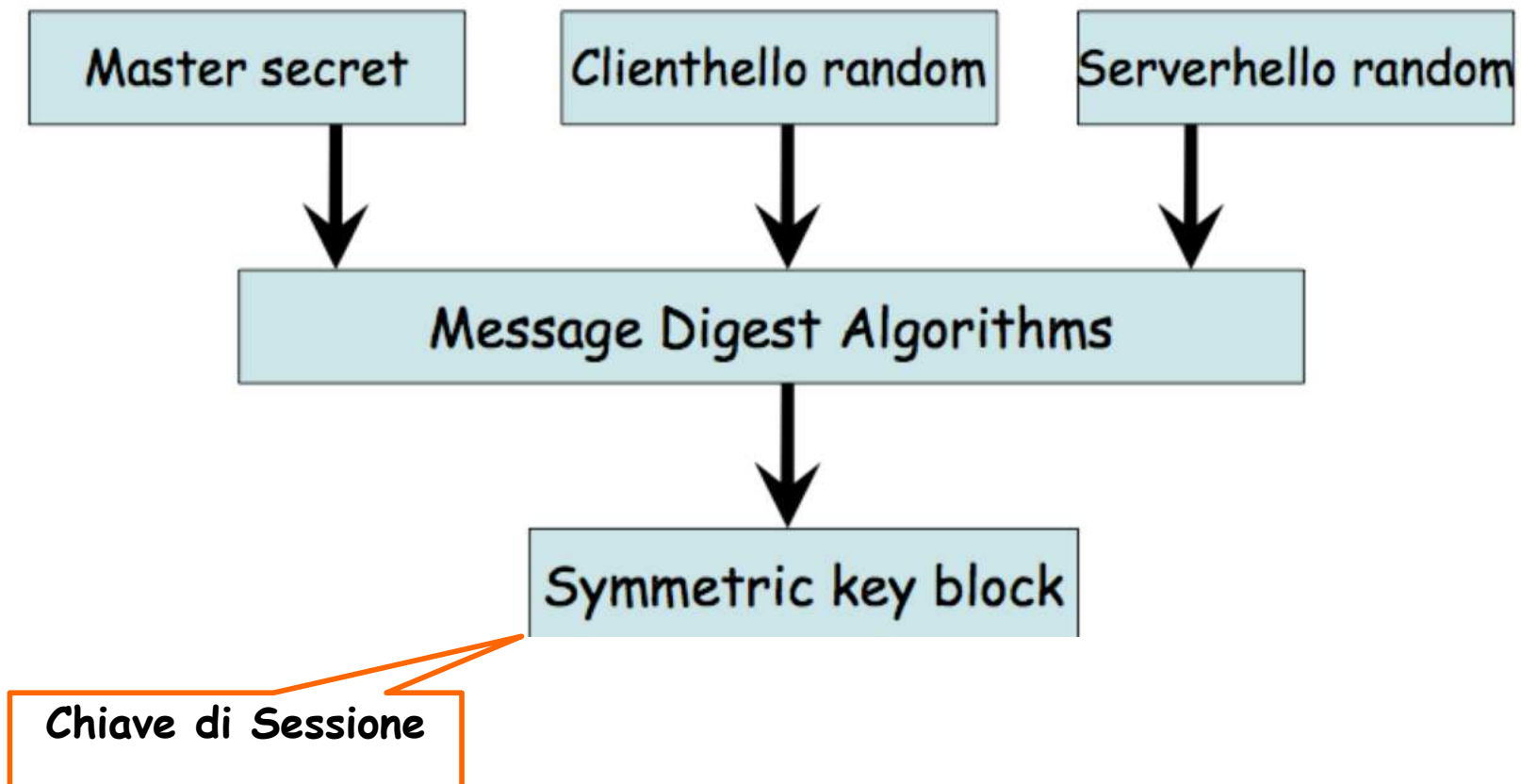
Calcolo della Chiave di Sessione - Fase 1



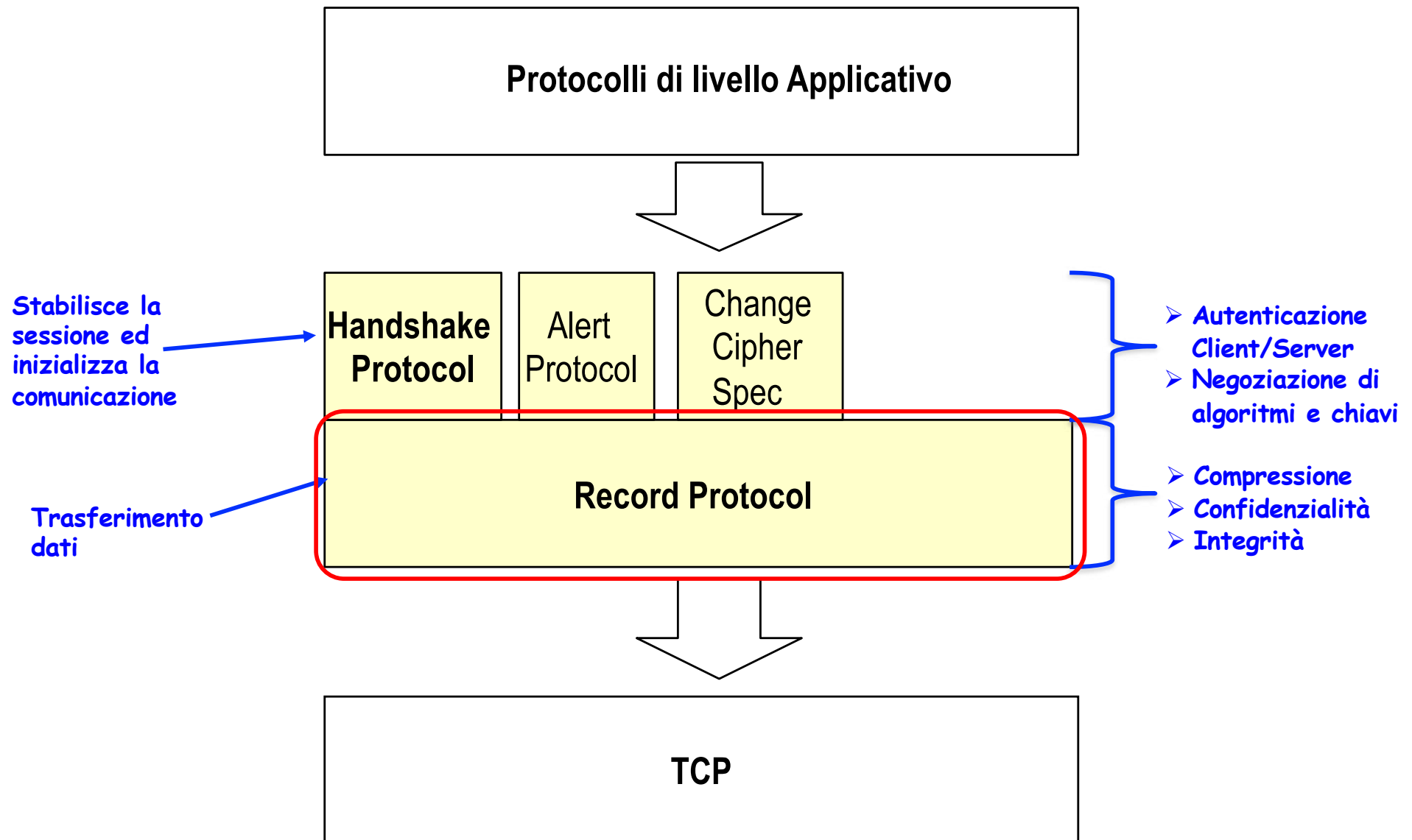
- **N.B.**: Il Master secret è sempre di 48 bit, mentre la lunghezza del Pre-master secret dipende dagli algoritmi negoziati

Handshake Protocol

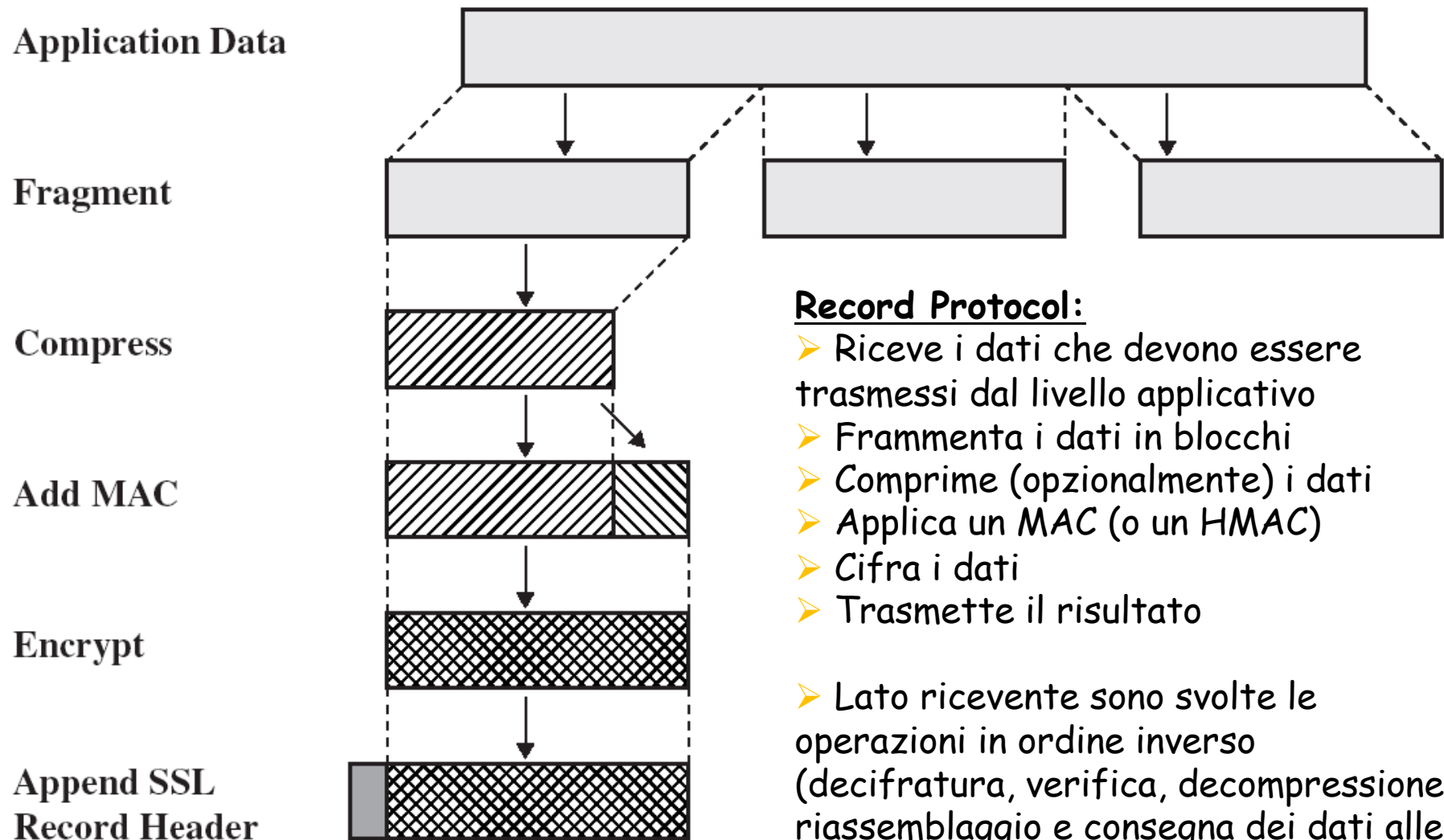
Calcolo della Chiave di Sessione - Fase 2



SSL/TLS Protocol Stack



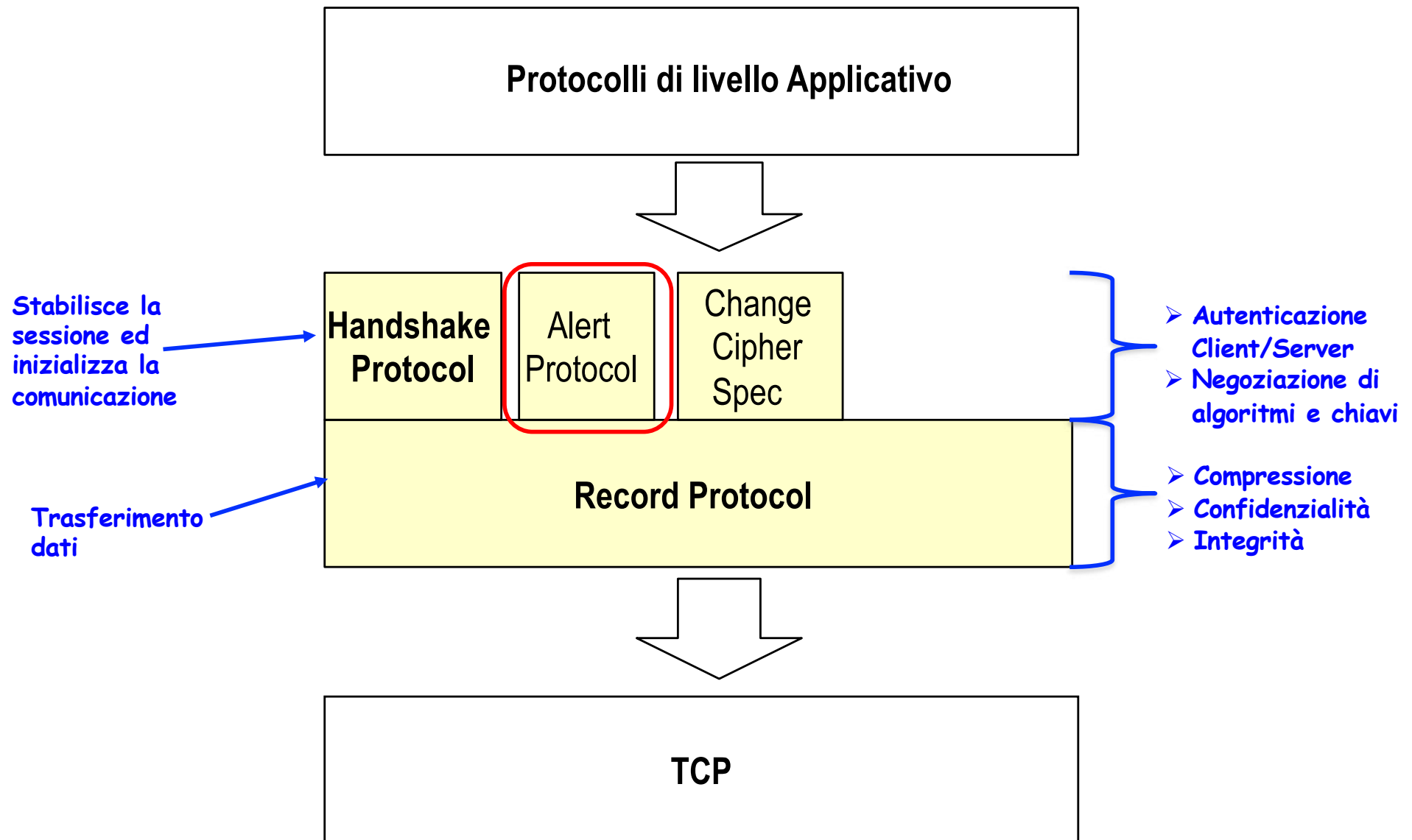
Record Protocol



Record Protocol:

- Riceve i dati che devono essere trasmessi dal livello applicativo
 - Frammenta i dati in blocchi
 - Comprime (opzionalmente) i dati
 - Applica un MAC (o un HMAC)
 - Cifra i dati
 - Trasmette il risultato
-
- Lato ricevente sono svolte le operazioni in ordine inverso (decifratura, verifica, decompressione, riassettaggio e consegna dei dati alle applicazioni)

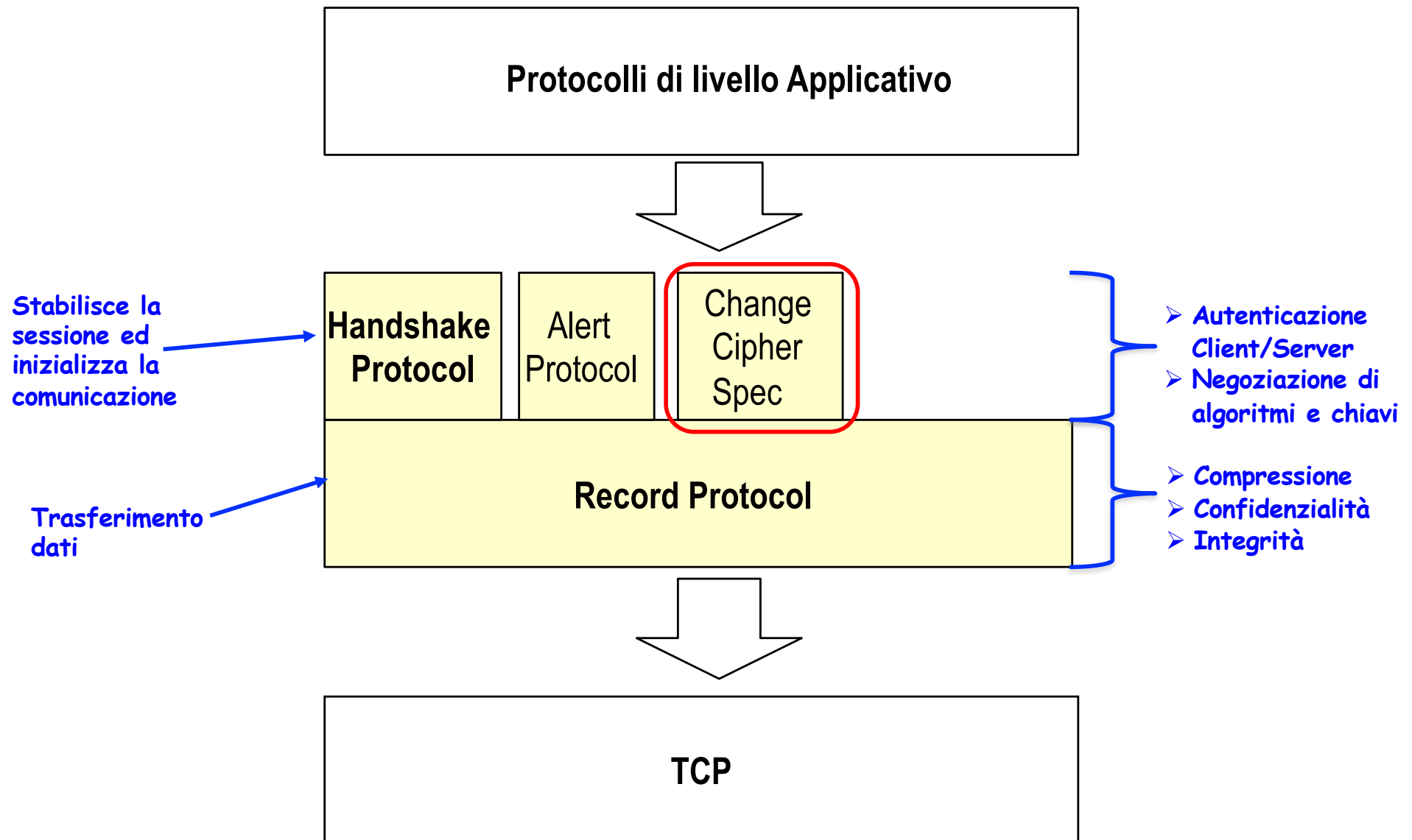
SSL/TLS Protocol Stack



Alert Protocol

- Usato da una parte per trasmettere messaggi di alert all'altra
- Ciascun messaggio è caratterizzato da un
 - **Livello di severità**
 - warning o fatal
 - **Tipo di alert**
 - unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter, close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown, etc

SSL/TLS Protocol Stack



Change Cipher Spec Protocol

- Utilizzato per aggiornare la ciphersuite in uso tra Client e Server
- Il protocollo consiste in un unico messaggio
 - Inviato da una delle due parti all'altra



Outline

- Caratteristiche ed Ambiti di Utilizzo
- Storia e Versioning
- Componenti
- **OpenSSL s_client ed s_server**
- SSL Server Test ed SSL Client Test

OpenSSL `s_client` ed `s_server`

- I comandi `s_client` ed `s_server` sono i principali strumenti forniti da OpenSSL per il debug di applicazioni Client/Server che utilizzano SSL/TLS
- Tali comandi
 - Possono essere eseguiti «indipendentemente» l'uno dall'altro
 - Sono configurabili mediante opportuni parametri
 - Che permettono di scegliere il tipo di connessione SSL/TLS che si intende stabilire

OpenSSL s_client

- Il comando `s_client`
 - Permette di realizzare tutte le funzioni di un semplice Client SSL/TLS
 - Può essere utilizzato per connettersi ad un Server che supporta tali protocolli
 - Utile soprattutto come strumento diagnostico per la creazione e la configurazione di Server SSL/TLS
 - Fornisce funzionalità molto simili a quelle offerte da Telnet

OpenSSL s_client

Struttura generale del comando s_client

```
openssl s_client args
```

➤ args

- -connect host:port Server e porta a cui connettersi (default localhost:4433)
- -CApath arg Directory con i certificati delle CA
- -CAfile arg File con i certificati delle CA
- -debug Visualizza ulteriori informazioni per il debug
- -cipher Specifica le Ciphersuite
- -verify arg Imposta la verifica del certificato del Server
- -cert arg Certificato da usare, in formato PEM
- -key arg Chiave privata relativa al certificato usato
- -msg Mostra i messaggi del protocollo
- -showcerts Mostra tutti i certificati presenti nella certificate chain
- -ssl2, -ssl3, -tls1, -no_ssl2, -no_ssl3, -no_tls1,... Richiedono o disabilitano l'uso delle versioni dei protocolli SSL o TLS specificati
- -starttls prot Permette di specificare quale protocollo si intende utilizzare over SSL/TLS. Attualmente, i protocolli supportati sono: smtp, pop3, imap, ftp, xmpp, xmpp-server, irc, postgres, mysql, lmtp, nntp, sieve, ldap

OpenSSL s_client

Struttura generale del comando s_client

`openssl s_client args`

➤ args

- `-connect host:port` Specifica l'host e la porta a cui connettersi (default localhost:4433)
- `-CApath arg` Directory con i file di certificato
- `-CAfile arg` File con i certificati
- `-debug` Visualizza ulteriori informazioni
- `-cipher` Specifica le cifre da usare
- `-verify arg` Imposta la verifica del certificato del Server
- `-cert arg` Certificato da usare, in formato PEM
- `-key arg` Chiave privata relativa al certificato usato
- `-msg` Mostra i messaggi del protocollo
- `-showcerts` Mostra tutti i certificati presenti nella certificate chain
- `-ssl2`, `-ssl3`, `-tls1`, `-no_ssl2`, `-no_ssl3`, `-no_tls1`, ... Richiedono o disabilitano l'uso delle versioni dei protocolli SSL o TLS specificati
- `-starttls prot` Permette di specificare quale protocollo si intende utilizzare over SSL/TLS. Attualmente, i protocolli supportati sono: `smtp`, `pop3`, `imap`, `ftp`, `xmpp`, `xmpp-server`, `irc`, `postgres`, `mysql`, `lmtp`, `nntp`, `sieve`, `ldap`

Per ottenere la lista completa delle opzioni del comando `s_client` è possibile utilizzare `man s_client`

OpenSSL s_client

Esempio

- Mediante il comando `s_client` è possibile interrogare un Server che offre connessioni SSL/TLS per uno specifico protocollo.
 - Esempio: interrogheremo un Server SMTP sulla porta 587

```
openssl s_client -msg -connect smtp.unipi.it:587 -starttls smtp
```



Handshake

```
>>> TLS 1.2, Handshake [length 0136], ClientHello
<<< TLS 1.2, Handshake [length 003d], ServerHello
<<< TLS 1.2, Handshake [length 0c36], Certificate
<<< TLS 1.2, Handshake [length 014d], ServerKeyExchange
<<< TLS 1.2, Handshake [length 0004], ServerHelloDone
>>> TLS 1.2, Handshake [length 0046], ClientKeyExchange
>>> TLS 1.2, ChangeCipherSpec [length 0001]
>>> TLS 1.2, Handshake [length 0010], Finished
<<< TLS 1.2, Handshake [length 0010], Finished
---
```

Output parziale

OpenSSL s_client

Esempio

- Mediante il comando `s_client` è possibile interrogare un Server che offre connessioni SSL/TLS per uno specifico protocollo.
 - **Esempio:** interrogheremo un Server SMTP sulla porta 587

```
openssl s_client -msg -connect smtp.unipi.it:587 -starttls smtp
```



Certificate Chain

```
---
Certificate chain
 0 s:C = IT, ST = Pisa, L = Pisa, O = Universit\C3\A0 di Pisa, OU = Area SERRA, CN = smtp.unipi.it
   i:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
 1 s:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
   i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Assured ID Root CA
---
```

OpenSSL s_client

Esempio

- Mediante il comando `openssl s_client` si può verificare la *Certificate Chain* di un server connesso.
- **Osservazioni**
 - Per ciascun certificato, la prima riga mostra il *Subject* e la seconda l'*Issuer*
 - Il primo certificato della *Certificate Chain* è il certificato foglia (certificato 0)
 - Si prosegue scorrendo la *Certificate Chain* verso il basso, verificando che l'*Issuer* del certificato corrente coincida col *Subject* del prossimo certificato
 - L'ultimo *Issuer* presente nella *Certificate Chain* può puntare a qualche *Certificato Root* che non è presente nella chain
 - Oppure, se si tratta di un *Certificato Self-signed*, può puntare a se stesso

Certificate Chain

```
---
Certificate chain
 0 s:C = IT, ST = Pisa, L = Pisa, O = Universit   di Pisa, OU = Area SERRA, CN = smtp.unipi.it
   i:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
 1 s:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
   i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Assured ID Root CA
---
```

OpenSSL s_client

Esempio

- Mediante il comando `s_client` è possibile interrogare un Server che offre connessioni SSL/TLS per uno specifico protocollo.
 - **Esempio:** interrogheremo un Server SMTP sulla porta 587

```
openssl s_client -msg -connect smtp.unipi.it:587 -starttls smtp
```



Certificate Chain

Certificate chain

Certificato 1

```
0 s:C = IT, ST = Pisa, L = Pisa, O = Universit\C3\A0 di Pisa, OU = Area SERRA, CN = smtp.unipi.it
  i:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
1 s:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
  i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Assured ID Root CA
```

OpenSSL s_client

Esempio

- Mediante il comando `s_client` è possibile interrogare un Server che offre connessioni SSL/TLS per uno specifico protocollo.
 - **Esempio:** interrogheremo un Server SMTP sulla porta 587

```
openssl s_client -msg -connect smtp.unipi.it:587 -starttls smtp
```



Certificate Chain

```
---
Certificate chain
 0 s:C = IT, ST = Pisa, L = Pisa, O = Universit\C3\A0 di Pisa, OU = Area SERRA, CN = smtp.unipi.it
   i:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
 1 s:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
   i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Assured ID Root CA
---
```

Certificato 2

OpenSSL s_client

Esempio

- Mediante il comando `s_client` è possibile interrogare un Server che offre connessioni SSL/TLS per uno specifico protocollo.
 - **Esempio:** interrogheremo un Server SMTP sulla porta 587

```
openssl s_client -msg -connect smtp.unipi.it:587 -starttls smtp
```



Certificate Chain

```
---
Certificate chain
 0 s:C = IT, ST = Pisa, L = Pisa, O = Universit\C3\A0 di Pisa, OU = Area SERRA, CN = smtp.unipi.it
  i:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
 1 s:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
  i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Assured ID Root CA
---
```

Subject

OpenSSL s_client

Esempio

- Mediante il comando `s_client` è possibile interrogare un Server che offre connessioni SSL/TLS per uno specifico protocollo.
 - **Esempio:** interrogheremo un Server SMTP sulla porta 587

```
openssl s_client -msg -connect smtp.unipi.it:587 -starttls smtp
```



Certificate Chain

```
---
Certificate chain
 0 s:C = IT, ST = Pisa, L = Pisa, O = Universit\C3\A0 di Pisa, OU = Area SERRA, CN = smtp.unipi.it
  i:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
 1 s:C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
  i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Assured ID Root CA
---
```



Issuer

OpenSSL s_client

Esempio

Server certificate

-----BEGIN CERTIFICATE-----

```
MIIHJjCCBg6gAwIBAgIQB2RwSDloiciqWFEHT6cUqTANBgkqhkiG9w0BAQsFADBk
MQswCQYDVQQGEwJOTDEWMBQGA1UECBMNTm9vcmtSG9sbGFuZDESMBAGA1UEBxMJ
QWlzdGVyZGFtMQ8wDQYDVQQKEwZURVJFTkExGDAWBgNVBAMTD1RFUkVOQSBTU0wg
Q0EgMzAeFw0xOTAxMDkwMDAwMDBaFw0yMTA0MTMwMDAwMDBaMHYxCzAJBgNVBAYT
AklUMQ0wCwYDVQQIEwRQaXNhMQ0wCwYDVQQHEwRQaXNhMRwwGgYDVQQKDBNvbm12
ZXJzaXTDoCBkaSBQaXNhMRMwEQYDVQQLEwpBcmVhIFNFULJBMRYwFAYDVQQDEw1z
bXRwLnVuaXBpLml0MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwsJI
4/e+GiR42eZWgzhlFpuZ+RKS2EpIgsNZw/fSx1hxxXuK52M1cAReeTCwymkA9Yob
Rm6A4SBiNB9otMfcZvAvhiEOR9zAkWljFPEDrDVV4afv8p4LngYW0sM8UVAUUrL1M
g88Lo/ZufvsTbml01WU7SRo/Nbr5vcDTJ2WXUpvFIdY0cCJ3fzOjHU+5BtUH6abL
fC3QcxfkEPAq43RqzGYFhDg8smoV3MGgYTYiPS1qLtZUuBGT5GYi3cjsROpFAQNT
Vr0qjv3FxiUxT3uVxbVct8KE8Wr+XESplHA6+dHLq1gfwlbrYdV7Z5vfBYMs1RH
o0mxiXdlpJemNFFA2wIDAQABo4IDwDCCA7wwHwYDVR0jBBGwFoAUZ/2IIBQnmMcJ
0iUZu+1REWN1UG1wHQYDVRO0BBYEFGlJmstweFNlclZWFp9Afrf15t8jMBGGA1Ud
EQQRMa+CDXNtdHAudW5pcGkuaXQwDgYDVROPAQH/BAQDAgWgMB0GA1UdJQQWMBQG
CCsGAQUFBwMBBggrBgEFBQcDAjBrBgNVHR8EZDBiMC+gLaArhilodHRwOi8vY3Js
My5kaWdpY2VydC5jb20vVEVSU5BU1NMQ0EzLmNybDavoC2gK4YpaHR0cDovL2Ny
bdQuZGlnaWN1cnQuY29tL1RFUkVOQVNTTENBMy5jcmwWTAYDVROgBEUwQzA3Bg1g
hkgBhv1sAQEWKjAoBggrBgEFBQcCARYcaHR0cHM6Ly93d3cuZGlnaWN1cnQuY29t
L0NQUzAIBGZngQwBAGIwbgYIKwYBBQUHAQEYjBgMCQGCSGAQUFBzABhhhodHRw
Oi8vb2Nzc5kaWdpY2VydC5jb20wOAYIKwYBBQUHMAKGLGh0dHA6Ly9jYWN1cnRz
LmRpZ21jZXJ0LmNvbS9URVJFTkFTU0xDQTMuY3J0MAWGA1UdEWEB/wQCMAAwggH2
BgorBgEEAdZ5AgQCBIIB5gSCAeIB4AB2AO5Lvbd1zmC64UjPH6vhnmajd35fsHLY
gwDEe4l6qP3LAAABaDIUJAoAAAQDAEcwRQIhAIqvrU4KQC7SPPJthqyRYrDKARR0
BUFYz2uxfp1IGUjmaIBrTAc4cXag47bBXWhVlPmPwpvRtetJJAmwzAdLZCsv9QB2
AIdlv+dZfPiMQ51fvfNu/1aNR1Y2/0q1YMG06v9eoIMPAAABADIUJOEAAAQDAEcw
RQIhAI5z1ZphjuPQDHXQrtvSuSBCiRl3WAKdTYDjrdkk8nhFAiAjvOSsImy9eRgU
/A8lQwNRFUpEiP/UJwJc4F2IXMuQWwB2APZclC/RdzAiFFQYCDUCUvo7jTRMzM7/f
DC8gC8x08WTjAAABaDIUJB0AAAQDAEcwRQIgFhOsvgofVTE/sBWmEoicWY01y7rX
```

*Subject ed Issuer del
certificato Server*

```
N9guFfEKPSBmdMvB+...6+1jDvP5PaSwFMM79IeYpvc9beIlwFvge3uY
nRVO9mtj4brJB1oCh/dga...zrAT+OimLp5MmqSWrARsUqQEKyXZYGCxtom
1BfRGEEJspNimRuGA6UINaKB...Fvt/YiD7xlQvKk9QjQbLTUQDHCrrZYhbUH
NDO23k/59k01f2n1GyjjwLDpGKlp...dytWe6mJS6CaOSavvAlN/x6HpwPPKPkf
NNQeCzLNMP19+g==
```

-----END CERTIFICATE-----


```
subject=C = IT, ST = Pisa, L = Pisa, O = Universit  C3   di Pisa, OU = Area SERRA, CN = smtp.unipi.it
issuer=C = NL, ST = Noord-Holland, L = Amsterdam, O = TERENA, CN = TERENA SSL CA 3
```

Certificato del Server,
codificato in Base64

OpenSSL s_client

Esempio

```
---
SSL handshake has read 4001 bytes and written 474 bytes
Verification: OK
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: 2158E0041F711D2D55066E6280C6B73CEAB4F5A1CE7150963C2231328AF453B7
    Session-ID-ctx:
    Master-Key: 8E499CE6573B33C1CFE74D46DF81609ADE26780BDCACE366340B3F62415F762FB41C790FBCCE3FDE7A56926F0E064D32
...
---
```



Versione del protocollo TLS
e Ciphersuite utilizzata

OpenSSL s_client

Esempio

```
---
SSL handshake has read 4001 bytes and written 474 bytes
Verification: OK
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: 2158E0041F711D2D55066E6280C6B73CEAB
    Session-ID-ctx:
    Master-Key: 8E499CE6573B33C1CFE74D46DF81609ADE2
...
---
```

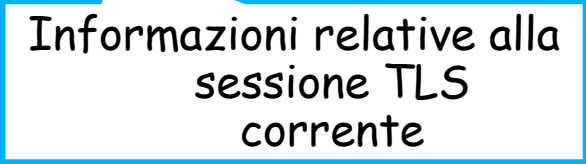
Informazioni relative alla chiave pubblica
del Server, alla rinegoziazione dei
parametri ed alla compressione

64D32

OpenSSL s_client

Esempio

```
---
SSL handshake has read 4001 bytes and written 474 bytes
Verification: OK
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: 2158E0041F711D2D55066E6280C6B73CEAB4F5A1CE7150963C2231328AF453B7
    Session-ID-ctx:
    Master-Key: 8E499CE6573B33C1CFE74D46DF81609ADE26780BDCACE366340B3F62415F762FB41C790FBCCE3FDE7A56926F0E064D32
...
---
```



Informazioni relative alla
sessione TLS
corrente

OpenSSL s_server

- Il comando `s_server`
 - Permette di realizzare tutte le funzioni di un Server SSL/TLS
 - Utile per il debug di applicazioni Client che supportano SSL/TLS
 - È possibile configurare l'esecuzione di questo comando attraverso l'impostazione di opportuni parametri
 - Ad es., eventuale uso di certificati, autenticazione Client, selezione della ciphersuite, versione del protocollo, etc.

OpenSSL s_server

Struttura generale del comando s_server

```
openssl s_server args
```

➤ args

- -accept arg Porta TCP/IP del Server (default 4433)
- -verify arg Richiede l'autenticazione Client
- -Verify arg Arresta la connessione se non c'è autenticazione del Client
- -cert arg File col certificato Server
- -key arg File con la chiave privata
- -debug Visualizza ulteriori informazioni per il debug
- -CApath arg Directory con i certificati delle CA
- -CAfile arg File con i certificati delle CA
- -ssl2, -ssl3, -tls1, -no_ssl2, -no_ssl3, -no_tls1,... Richiedono o disabilitano l'uso delle versioni dei protocolli SSL o TLS specificati
- -cipher arg Specifica le Ciphersuite
- -www Risposta a GET / con una pagina di prova

OpenSSL s_server

Struttura generale del comando s_server

`openssl s_server args`

➤ args

- `-accept arg` Porta TCP su cui il Server (default 4433)
- `-verify arg` Richiede la verifica del Client
- `-Verify arg` Arresta la comunicazione del Client
- `-cert arg` File col certificato
- `-key arg` File con la chiave privata
- `-debug` Visualizza ulteriori informazioni per il debug
- `-CApath arg` Directory con i certificati delle CA
- `-CAfile arg` File con i certificati delle CA
- `-ssl2, -ssl3, -tls1, -no_ssl2, -no_ssl3, -no_tls1, ...` Richiedono o disabilitano l'uso delle versioni dei protocolli SSL o TLS specificati
- `-cipher arg` Specifica le Ciphersuite
- `-www` Risposta a GET / con una pagina di prova

Per ottenere la lista completa delle opzioni del comando `s_server` è possibile utilizzare `man s_server`

OpenSSL s_server

Esempio 1

- Mediante il comando `s_server` è possibile creare un Server che offre connessioni SSL/TLS
 - Esempio: Server in ascolto sulla porta 12345

```
openssl s_server -accept 12345 -cert server-cert.pem -key  
private/server-key.pem
```



```
Enter pass phrase for private/server-key.pem:  
Using default temp DH parameters  
Using default temp ECDH parameters  
ACCEPT
```

Il Server è stato avviato ed
è pronto ad accettare
richieste di connessione

OpenSSL s_server

Esempio 1

- Mediante il comando `s_client` è possibile connettersi al Server sulla porta 12345

```
openssl s_client -msg -connect localhost:12345
```

- Alla ricezione di una richiesta di connessione effettuata mediante l'`s_client`, l'`s_server` mostrerà un output simile a quello seguente

```
-----BEGIN SSL SESSION PARAMETERS-----
MHUCAQECAgMBBAIAOQQgU/gS+ul8meXlMVvGTnfdrc/ddc0gXnqFsk+yE3c
sT/ME
MLYSu2pX8cv9ZJH29UfuxDkNzUYU7oopAvPybF/iEBzZfj68SzLkh7LL/Fw
FpXog
JKEGAgRYux70ogQCAgEspAYEBAEAAAA=
-----END SSL SESSION PARAMETERS-----
Shared ciphers:DHE-RSA-AES256-SHA:DHE-DSS-AES256-
SHA:AES256-SHA:EDH-RSA-DES-CBC3-SHA:EDH-DSS-DES-CBC3-
SHA:DES-CBC3-SHA:DHE-RSA-AES128-SHA:DHE-DSS-AES128-
SHA:AES128-SHA:DHE-RSA-SEED-SHA:DHE-DSS-SEED-SHA:SEED-
SHA:RC4-SHA:RC4-MD5:EDH-RSA-DES-CBC-SHA:EDH-DSS-DES-CBC-
SHA:DES-CBC-SHA:EXP-EDH-RSA-DES-CBC-SHA:EXP-EDH-DSS-DES-
CBC-SHA:EXP-DES-CBC-SHA:EXP-RC2-CBC-MD5:EXP-RC4-MD5
CIPHER is DHE-RSA-AES256-SHA
Secure Renegotiation IS supported
```

OpenSSL s_server

Esempio 1

Client

Server

```
-----BEGIN SSL SESSION PARAMETERS-----
MHUCAQECAgMBBAIAOQQgU/gS+ul8meXlMVvGTnfdr
c/ddc0gXnqFsk+yE3csT/ME
MLYSu2pX8cv9ZJH29UfuxDkNzUYU7oopAvPybF/iE
BzZfj68SzLkh7LL/FwFpXog
JKEGAgRYux70ogQCAGEspAYEBAEAAAA=
-----END SSL SESSION PARAMETERS-----
Shared ciphers:DHE-RSA-AES256-SHA:DHE-
DSS-AES256-SHA:AES256-SHA:EDH-RSA-DES-
CBC3-SHA:EDH-DSS-DES-CBC3-SHA:DES-CBC3-
SHA:DHE-RSA-AES128-SHA:DHE-DSS-AES128-
SHA:AES128-SHA:DHE-RSA-SEED-SHA:DHE-DSS-
SEED-SHA:SEED-SHA:RC4-SHA:RC4-MD5:EDH-
RSA-DES-CBC-SHA:EDH-DSS-DES-CBC-SHA:DES-
CBC-SHA:EXP-EDH-RSA-DES-CBC-SHA:EXP-EDH-
DSS-DES-CBC-SHA:EXP-DES-CBC-SHA:EXP-RC2-
CBC-MD5:EXP-RC4-MD5
CIPHER is DHE-RSA-AES256-SHA
Secure Renegotiation IS supported
```

ciao
prova

```
New, TLSv1/SSLv3, Cipher is DHE-RSA-
AES256-SHA
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol    : TLSv1
    Cipher      : DHE-RSA-AES256-SHA
    Session-ID: 53F812FAE97C99E5F5315BC64E77DDADCFDD75CD2
05E7A85B24FB213772C4FF3
    Session-ID-ctx:
    Master-Key: B612BB6A57F1CBFD6491F6F547EEC4390DCD4614E
E8A2902F3F26C5FE2101CD97E3EBC4B32E487B2CB
FC5C05A57A2024
    Key-Arg     : None
    Start Time: 1488658164
    Timeout     : 300 (sec)
    Verify return code: 21 (unable to
verify the first certificate)
---
```

ciao
prova

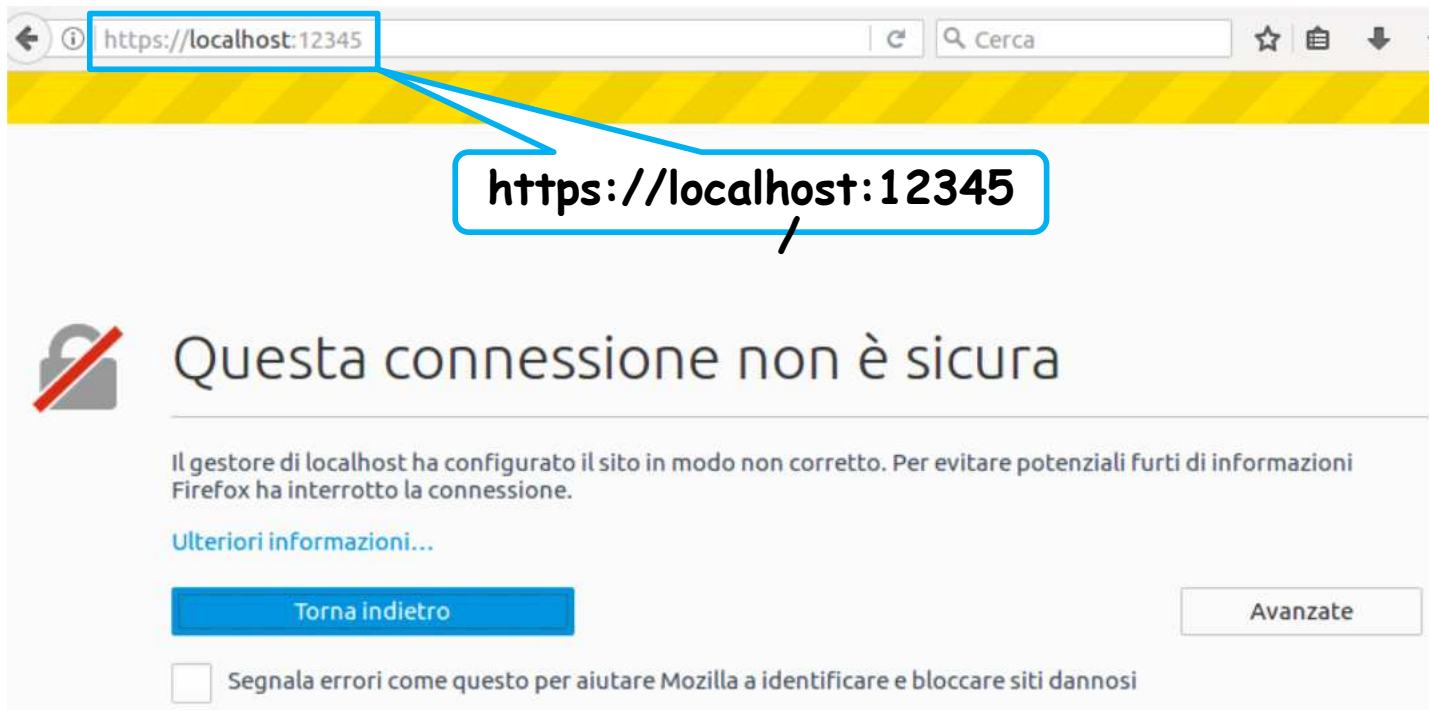
Dopo che la sessione sicura è stata stabilita, è possibile inviare messaggi, dal Client al Server e viceversa

OpenSSL s_server

Esempio 2

- Mediante il comando `s_server` si può creare un semplice Web Server a cui connettersi tramite browser

```
openssl s_server -accept 12345 -cert server-cert.pem -key  
private/server-key.pem -www
```



Connessione tramite browser, al Web Server SSL/TLS simulato dal comando `s_server`

Outline

- Caratteristiche ed Ambiti di Utilizzo
- Storia e Versioning
- Componenti
- OpenSSL s_client ed s_server
- **SSL Server Test ed SSL Client Test**

SSL Server Test ed SSL Client Test

- Esistono vari strumenti per valutare la sicurezza di un Server o di un Client che supporta SSL/TLS
- Strumenti Web-based (gratuiti), resi disponibili dalla società Qualys SSL Labs
 - SSL Server Test
 - SSL Client Test

SSL Server Test

➤ <https://www.ssllabs.com/ssltest/index.html>

Q QUALYS[®] SSL LABS

[Home](#) [Projects](#) [Qualys.com](#) [Contact](#)

You are here: [Home](#) > [Projects](#) > SSL Server Test

SSL Server Test

This free online service performs a deep analysis of the configuration of any SSL web server on the public Internet. **Please note that the information you submit here is used only to provide you the service. We don't use the domain names or the test results, and we never will.**

Hostname:

☐ Do not show the results on the boards

www.unisa.i
†

SSL Server Test

This server supports weak Diffie-Hellman (DH) key exchange parameters. Grade capped to B. [MORE INFO »](#)

The server supports only older protocols, but not the current best TLS 1.2 or TLS 1.3. Grade capped to C. [MORE INFO »](#)

This server does not support Forward Secrecy with the reference browsers. Grade capped to B. [MORE INFO »](#)

This server does not support Authenticated encryption (AEAD) cipher suites. Grade capped to B. [MORE INFO »](#)

This server supports TLS 1.0. Grade capped to B. [MORE INFO »](#)

Livello generale di sicurezza del server

SSL Server Test

Certificate #1: RSA 2048 bits (SHA256withRSA)



Server Key and Certificate #1



Subject	www.unisa.it Fingerprint SHA256: 1f8a8b0251785e1c4a8685d0a7c648dd31bb7973e59e0b6ad00d50ecf59acadb Pin SHA256: jCvNAFF0li6OMvP7hMQEVW4DF8TKjHkFyIMDGCK+Wc=
Common names	www.unisa.it
Alternative names	www.unisa.it unisa.it web.unisa.it www.personaldesk.unisa.it rubrica.unisa.it docenti.unisa.it corsi.unisa.it dipartimenti.unisa.it www.diem.unisa.it www.disa.unisa.it www.dises.unisa.it www.dispac.unisa.it www.dcb.unisa.it www.dipmed.unisa.it www.dsg.unisa.it www.difarma.unisa.it www.di.unisa.it www.df.unisa.it www.dipmat.unisa.it www.dipsun.unisa.it www.dispsc.unisa.it www.dijn.unisa.it www.diciv.unisa.it www.disuff.unisa.it trasparenza.unisa.it www.pqa.unisa.it www.cqa.unisa.it www.alternanza.unisa.it www.bilanciosociale.unisa.it www.disabilidsa.unisa.it www.placement.unisa.it wifi.unisa.it www.cug.unisa.it www.pharmanomics.unisa.it www.beta.unisa.it cd.unisa.it www.numismatica.unisa.it www.multical.unisa.it www.blockchain.unisa.it alternanza.unisa.it scelgodilessereinformatica.unisa.it neuronelab.unisa.it www.rnrc.unisa.it www.gcf.unisa.it www.biblioteche.unisa.it www.disps.unisa.it www.dispc.unisa.it ambiente.unisa.it www.ambiente.unisa.it
Serial Number	0423f3b7ff9303eb03bf8f44948b96ac
Valid from	Thu, 10 Oct 2019 00:00:00 UTC
Valid until	Wed, 12 Jan 2022 00:00:00 UTC (expires in 1 year and 8 months)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No
Issuer	TERENA SSL CA 3 AIA: http://cacerts.digicert.com/TERENASSLCA3.crl
Signature algorithm	SHA256withRSA
Extended Validation	No
Certificate Transparency	Yes (certificate)
OCSP Must Staple	No
Revocation information	CRL, OCSP CRL: http://crl3.digicert.com/TERENASSLCA3.crl OCSP: http://ocsp.digicert.com
Revocation status	Good (not revoked)

SSL Server Test



Additional Certificates (if supplied)

Certificates provided 3 (5049 bytes)

Chain issues Contains anchor


#2

Subject TERENA SSL CA 3
Fingerprint SHA256: beb8efe9b1a73c841b375a90e5fff8048848e3a2af66f6c4dd7b938d6fe8c5d8
Pin SHA256: 8651wEkMkH5ftiaLp57oqmx3KHTFzDgp7ZeJXR0ToBs=
Valid until Mon, 18 Nov 2024 12:00:00 UTC (expires in 4 years and 6 months)
Key RSA 2048 bits (e 65537)
Issuer DigiCert Assured ID Root CA
Signature algorithm SHA256withRSA

#3

Subject DigiCert Assured ID Root CA In trust store
Fingerprint SHA256: 3e9099b5015e8f486c00bcea9d111ee721faba355a89bcf1df69561e3dc6325c
Pin SHA256: I/Lt/z7ekCWanjD0Cvj5EqXis2lOaThEA0H2Bg4BT/o=
Valid until Mon, 10 Nov 2031 00:00:00 UTC (expires in 11 years and 6 months)
Key RSA 2048 bits (e 65537)
Issuer DigiCert Assured ID Root CA Self-signed
Signature algorithm SHA1withRSA Weak, but no impact on root certificate

SSL Server Test



Certification Paths		
Mozilla Apple Android Java Windows		
Path #1: Trusted		
1	Sent by server	<p>www.unisa.it</p> <p>Fingerprint SHA256: 1f8a8b0251785e1c4a8685d0a7c648dd31bb7973e59e0b6ad00d50ecf59acadb</p> <p>Pin SHA256: jCvNAFF0Ii6OMvP7hMQEVW4DF8TKkjHkFylMDGCK+Wc=</p> <p>RSA 2048 bits (e 65537) / SHA256withRSA</p>
2	Sent by server	<p>TERENA SSL CA 3</p> <p>Fingerprint SHA256: beb8efe9b1a73c841b375a90e5fff8048848e3a2af66f6c4dd7b938d6fe8c5d8</p> <p>Pin SHA256: 8651wEkMkH5ftiaLp57oqmx3KHTFzDgp7ZeJXR0ToBs=</p> <p>RSA 2048 bits (e 65537) / SHA256withRSA</p>
3	Sent by server In trust store	<p>DigiCert Assured ID Root CA Self-signed</p> <p>Fingerprint SHA256: 3e9099b5015e8f486c00bcea9d111ee721faba355a89bcf1df69561e3dc6325c</p> <p>Pin SHA256: l/Lt/z7ekCWanjD0Cvj5EqXls2lOaThEA0H2Bg4BT/o=</p> <p>RSA 2048 bits (e 65537) / SHA1withRSA</p> <p>Weak or insecure signature, but no impact on root certificate</p>

SSL Server Test



Protocols

TLS 1.3

TLS 1.2

TLS 1.1

TLS 1.0

SSL 3

SSL 2



Cipher Suites

TLS 1.0 (suites in server-preferred order)

TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33) DH 1024 bits FS **WEAK**

TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) DH 1024 bits FS **WEAK**

TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x16) DH 1024 bits FS **WEAK**

TLS_RSA_WITH_AES_128_CBC_SHA (0x2f) **WEAK**

TLS_RSA_WITH_AES_256_CBC_SHA (0x35) **WEAK**

TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa) **WEAK**

SSL Client Test

➤ <https://www.ssllabs.com/ssltest/viewMyClient.html>

Protocol Support
Your user agent has good protocol support. Your user agent supports TLS 1.2 and TLS 1.3, which are recommended protocol version at the moment.
CVE-2020-0601 (CurveBall) Vulnerability
Your user agent is not vulnerable. For more information about the CVE-2020-0601 (CurveBall) Vulnerability, please go to CVE-2020-0601 . To test manually, click here . Your user agent is not vulnerable if it fails to connect to the site.
Logjam Vulnerability
Your user agent is not vulnerable. For more information about the Logjam attack, please go to weakdh.org . To test manually, click here . Your user agent is not vulnerable if it fails to connect to the site.
FREAK Vulnerability
Your user agent is not vulnerable. For more information about the FREAK attack, please go to www.freakattack.com . To test manually, click here . Your user agent is not vulnerable if it fails to connect to the site.
POODLE Vulnerability
Your user agent is not vulnerable. For more information about the POODLE attack, please read this blog post .



SSL Client Test



Protocols

TLS 1.3

TLS 1.2

TLS 1.1

TLS 1.0

SSL 3

SSL 2



Cipher Suites (in order of preference)

TLS_GREASE_3A (0x3a3a)

TLS_AES_128_GCM_SHA256 (0x1301) Forward Secrecy

TLS_AES_256_GCM_SHA384 (0x1302) Forward Secrecy

TLS_CHACHA20_POLY1305_SHA256 (0x1303) Forward Secrecy

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b) Forward Secrecy

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) Forward Secrecy

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c) Forward Secrecy

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) Forward Secrecy

TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9) Forward Secrecy

TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8) Forward Secrecy

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) **WEAK**

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) **WEAK**

TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c) **WEAK**

TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d) **WEAK**

TLS_RSA_WITH_AES_128_CBC_SHA (0x2f) **WEAK**

TLS_RSA_WITH_AES_256_CBC_SHA (0x35) **WEAK**

TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa) **WEAK**

SSL Client Test

Protocol Details

Server Name Indication (SNI)	Yes
Secure Renegotiation	Yes
TLS compression	No
Session tickets	Yes
OCSP stapling	Yes
Signature algorithms	SHA256/ECDSA, RSA_PSS_SHA256, SHA256/RSA, SHA384/ECDSA, RSA_PSS_SHA384, SHA384/RSA, RSA_PSS_SHA512, SHA512/RSA, SHA1/RSA
Named Groups	tls_grease_2a2a, x25519, secp256r1, secp384r1
Next Protocol Negotiation	No
Application Layer Protocol Negotiation	Yes h2 http/1.1
SSL 2 handshake compatibility	No

Bibliografia

- **Cryptography and Network Security (Principles and Practice) - Settima Edizione**

by W. Stallings, 2017

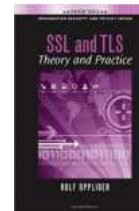
- cap. 17



- **SSL and TLS (Theory and Practice)**

by Rolf Oppliger, 2009

- cap. 3, 4 e 5



- **Network Security with OpenSSL**

Pravir Chandra, Matt Messier and John Viega (2002), O'Reilly

- Cap. 1
- Appendix A. Command-Line Reference



- **Documentazione su OpenSSL**

- https://wiki.openssl.org/index.php/SSL_and_TLS_Protocols