Secure Socket Layer (SSL) - Transport Layer Security (TLS)

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Email: gianluca.dini@unipi.it
Version: 28/04/2025

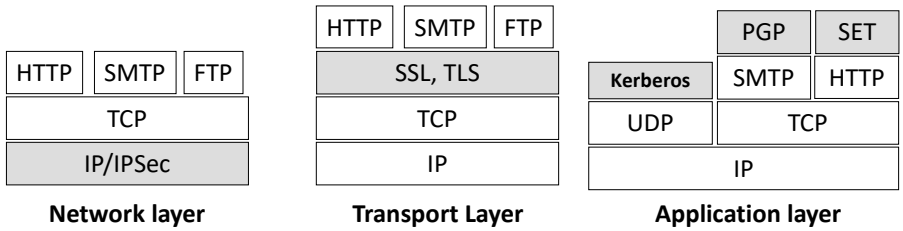
1

SSL

INTRODUCTION

2

Security in the TCP/IP stack

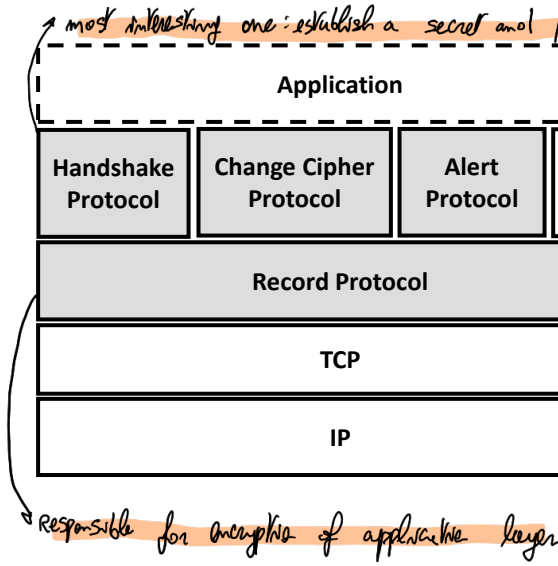


They work at T.L.
and they are a collection of protocols

3

The SSL protocol suite

most interesting one: establish a secret and protect communication between the two parties



Responsible for encryption of application layer

4

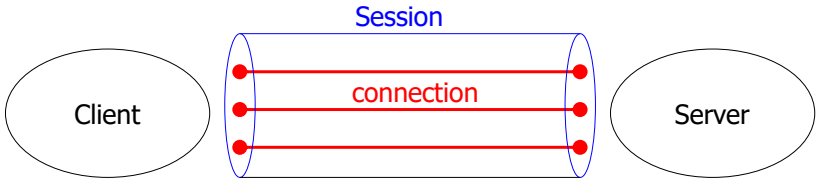
References

- Secure Socket Layer (SSL)
  - Netscape
  - http://wp.netscape.com/eng/ssl3/
- Transport Layer Security (TLS)
  - Based on SSL v3.0
  - RFC 2246
  - ftp://ftp.rfc-editor.org/in-notes/rfc2246.txt
- Same design as SSL but different algorithms

History of the protocol

- SSL
  - Developed by Netscape in mid 1990s
  - SSLv1 broken at birth (never publicly released)
  - SSLv2 flawed, now IETF-deprecated (RFC 6176)
  - SSLv3 still widely supported (since 1996)
- TLS
  - IETF-standardized version of SSL.
  - TLS 1.0 in RFC 2246 (1999)
    - Based on SSLv3 but NOT interoperable
  - TLS 1.1 in RFC 4346 (2006).
  - TLS 1.2 in RFC 5246 (2008).

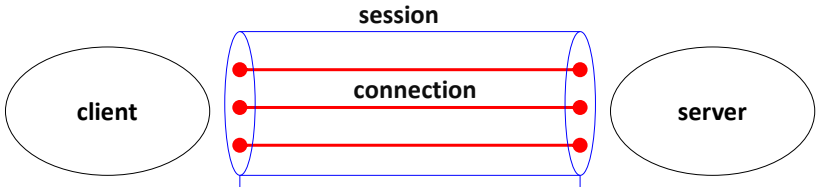
Session vs connection



- A **session** is a *logical association* between a Client and a Server
  - Created by the **Handshake protocol**
  - Define a set of **crypto pars** that can be shared by multiple connections
  - Avoid **expensive** negotiation of crypto pars for each connection

7

Session vs connection

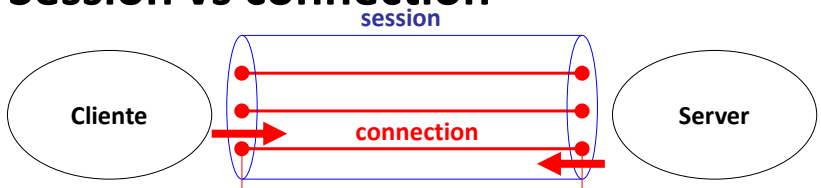


SESSION STATE

- session identifier
- Peer certificate (X.509v3)
- compression method
- cipher spec
- master secret (48 bytes)
- Is resumable

8

Session vs connection



As long as Session is alive you can establish connection

CONNECTION STATE

- Server random number (nonce) } For connection freshness to avoid replay from past connection
- Client random number (nonce) }
- Server write MAC secret } keys for MAC implementation. 2: we use one key in one dir, one for other direction.
- Client write MAC secret }
- Server write key } For encryption, one for server → Client, one for Client → Server
- Client write key }
- Initialization vectors: if needed
- Sequence numbers: against replay attacks

9

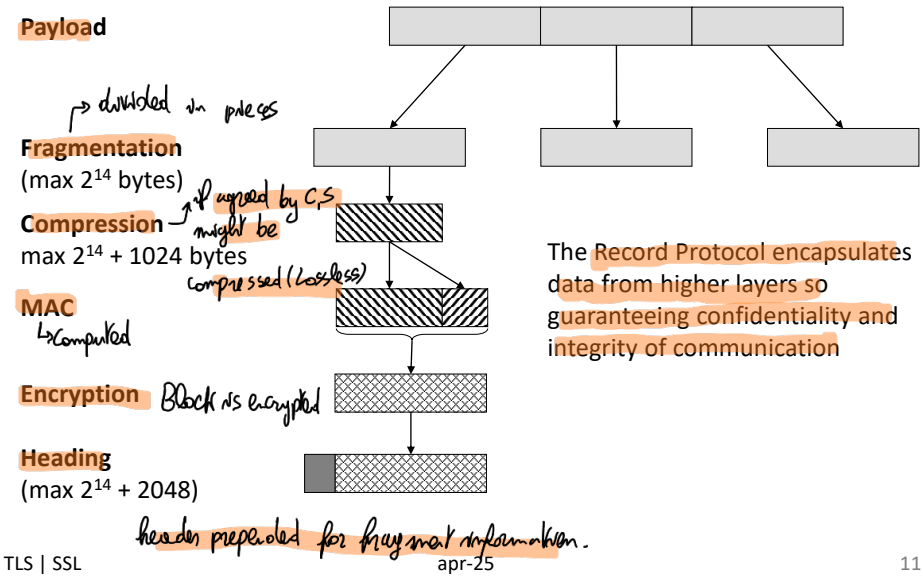
Keys and values generated by themselves

SSL | TLS

THE RECORD PROTOCOL

10

The Record Protocol



The Record Protocol encapsulates data from higher layers so guaranteeing confidentiality and integrity of communication

11

The Record Protocol

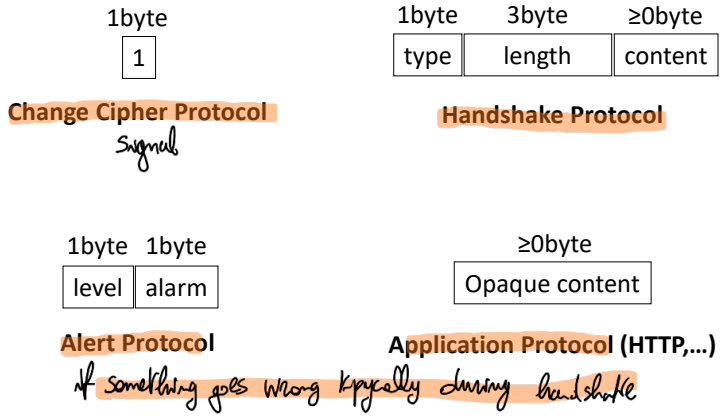
- **Fragmentation** fragments application data in blocks whose size ≤ 2^{14}-bytes
- **Compression** must be lossless and must not increase the block size more than 1024 bytes (default = null)
- **MAC** uses the [Server|Client] write MAC key, sequence number, compressed block, padding
- **Encryption** uses the [Server|Client] write key
  - Block and steam ciphers
  - Does not increase the content size more than 1024 byte
- Total length of a fragment must be ≤ 2^{14} + 2048 bytes

12

The Record Protocol - Header

- Fields of the header
  - **Payload type** (8 bit): change cipher, alert, handshake, application_data
  - **Major Version** (8 bit)
  - **Minor Version** (8 bit)
  - **Compressed length** (16 bit): size of the cleartext fragment (max val = 2^{14} + 2048)

Payload types



The other protocols in the SSL suite

- The **change cipher spec protocol** consists in one single message (cleartext) to make the negotiated crypto suite operational

15

The other protocols in the SSL suite

- The **alert protocol** notifies alarms to peers

FATAL ALARMS	OTHER ALARMS
unexpected_message	no_certificate
bad_record mac	bad_certificate
decompression_failure	unsupported_certificate
handshake_failure	certificate_revoked
illegal_parameter	certificate_expired
	certificate_unknown
	close_notify

16

SSL

THE HANDSHAKE PROTOCOL

17

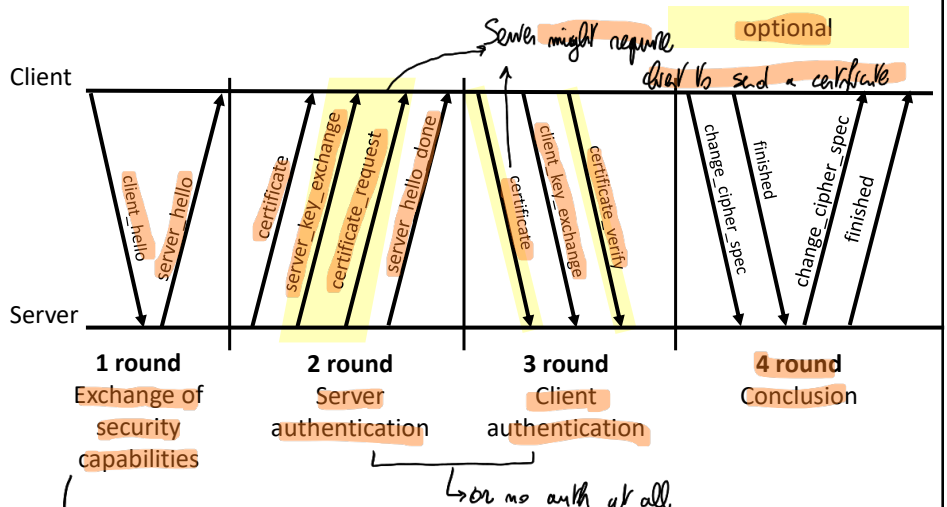
The Handshake Protocol

- Establish a secure session
  - Client and server authenticate each other (in general, usually only server is authenticated by client, that gets auth. at higher level)
  - (1) Client and server negotiate the cipher suite
    - Key establishment scheme;
    - Encryption scheme (used in the Record Protocol)
    - MAC (used in the Record Protocol)
  - Client and server establish a shared secret
    - E.g., pre-master secret
- Before any application data exchange
- The most complex part of SSL

(1) You want to identify risk to all customers from all countries (that might have different standards nationally)

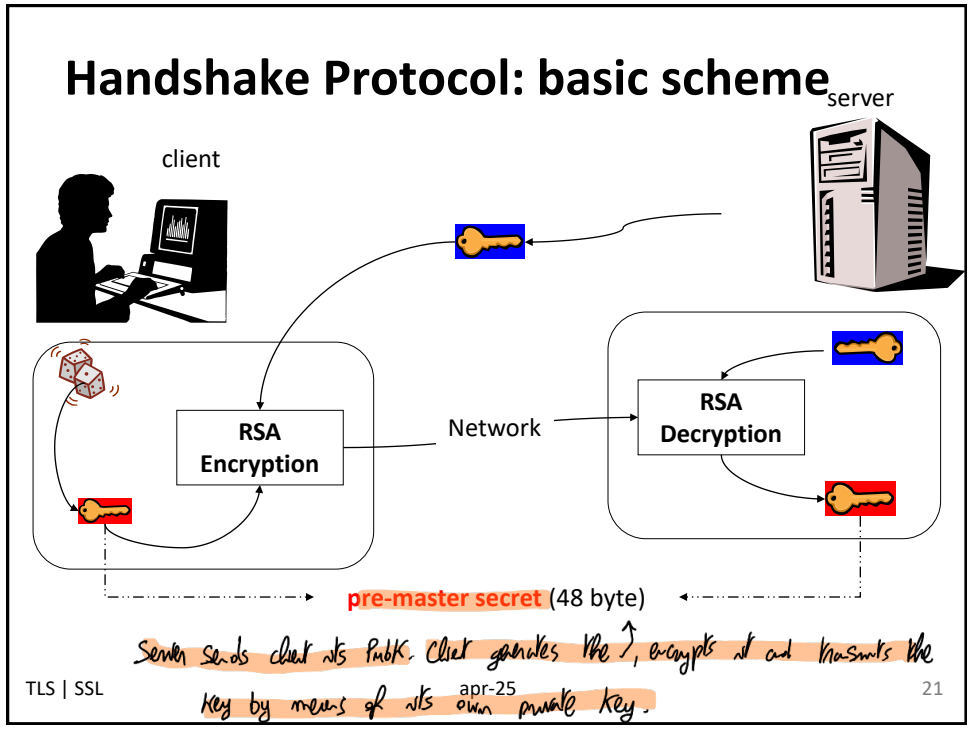
18

Handshake protocol: an overview

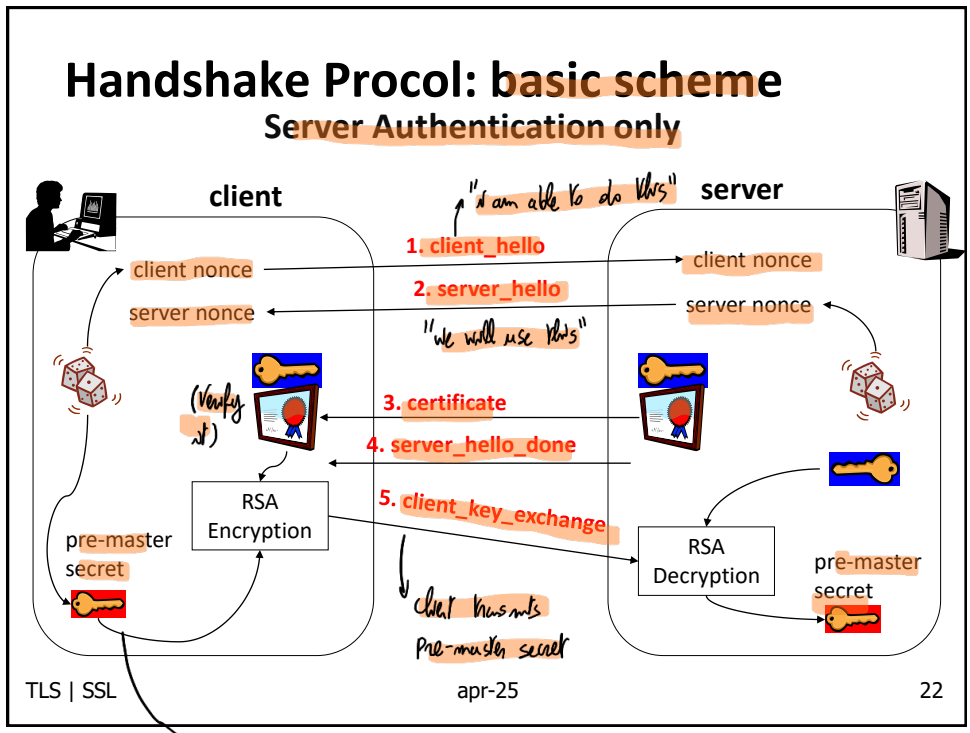


Set of messages

Type	Contents
hello_request	No pars
client_hello	version, nonce, session id, cipher suite, compression method
server_hello	version, nonce, session_id, cipher suite, compression method
certificate	Certificate X.509v3
server_key_exchange	parameters, signature
certificate_request	Type, authority
server_hello_done	No pars
certificate_verify	signature
client_key_exchange	Parameterss, signature
finished	hash


Handshake Protocol: basic scheme
server
client
RSA Encryption
Network
RSA Decryption
pre-master secret (48 byte)
Server sends client its PubK. Client generates the ↑, encrypts it and transmits the key by means of its own private key.
TLS | SSL
apr-25
21

21


Handshake Procol: basic scheme
Server Authentication only
client
server
"I am able to do these"
1. client_hello
client nonce
client nonce
2. server_hello
server nonce
server nonce
"we will use this"
(Verify it)
3. certificate
4. server_hello_done
RSA Encryption
5. client_key_exchange
RSA Decryption
pre-master secret
pre-master secret
client transmits pre-master secret
generated by client nonce
TLS | SSL
apr-25
22

22

Hello message

- By means of Hello msgs, Client and Server tell each other what they are able to do
  - SSL version
  - Random: timestamp (32 bit) + random bytes (28 bytes)
  - Session id
  - Cipher suite
  - Compression method

- **ClientHello random**: A 32-byte random number generated by the client.
- **ServerHello random**: A 32-byte random number generated by the server.

Now, the random fields are used mainly for two things:

1. **They get mixed into the key derivation process.**
   When the client and server eventually compute the **master secret** (that "special sauce" that protects the whole session), they mix in the client random and server random to make sure that every session key is *unique* and *cannot be replayed*.

   Even if you somehow used the same pre-master secret (which shouldn't happen), having different randoms guarantees different master secrets and session keys.
2. **They help with defending against replay attacks and certain types of cryptographic attacks.**

23

There's another fun secret hidden inside:
The first 4 bytes of each random field are the current UNIX timestamp.
Yes — TLS designers thought it would be useful for debugging and loose freshness checks!
So, the random field looks like:

- 4 bytes = current time (seconds since 1970)
- 28 bytes = random garbage

But from the cryptographic point of view, it's treated as just a 32-byte random blob.

Cipher suite

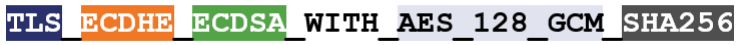
- Cipher suite is a list of algorithm *tuples*
- A *tuple* specifies
  - Key exchange algorithm (RSA, DH, DHE, ECDHE, PSK)
  - Digital Signature Algorithm (RSA, ECDSA, DSA)
  - Bulk encryption (AES, DES, 3DES, IDEA, RC4,…) Ciphers, both block and stream ciphers
  - MAC Algorithm (MD5, SHA-1, SHA-256,…)
  - Cypher type, IV size, isExportable
  - Hash size

→ or if DH for ex. negotiate prime, key size etc.

24

Cipher suite tuple

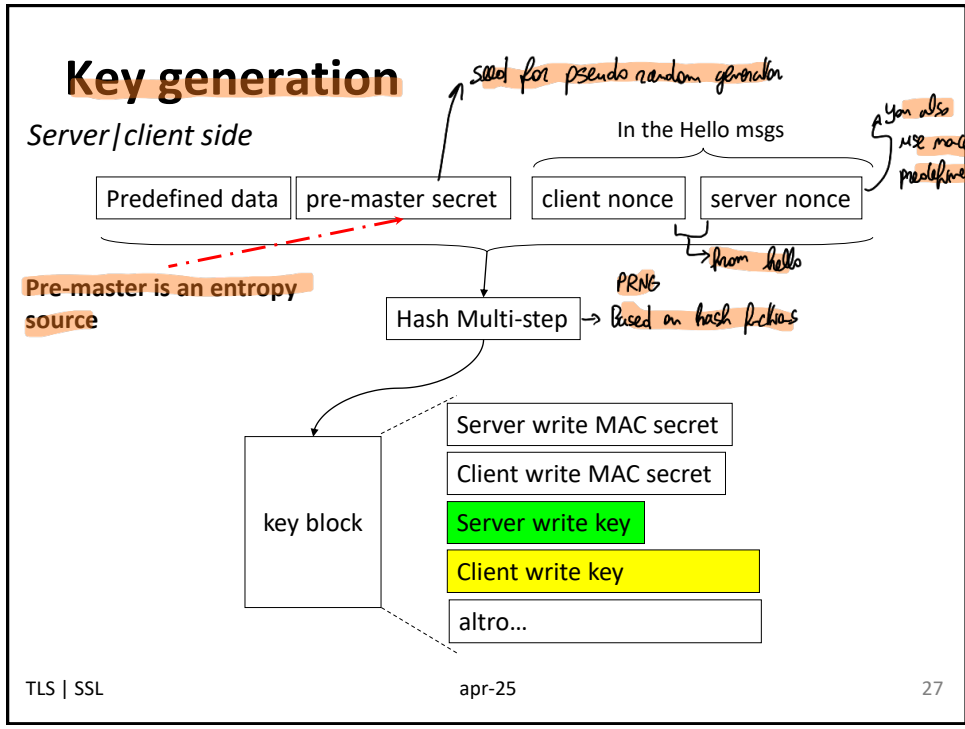
- An example



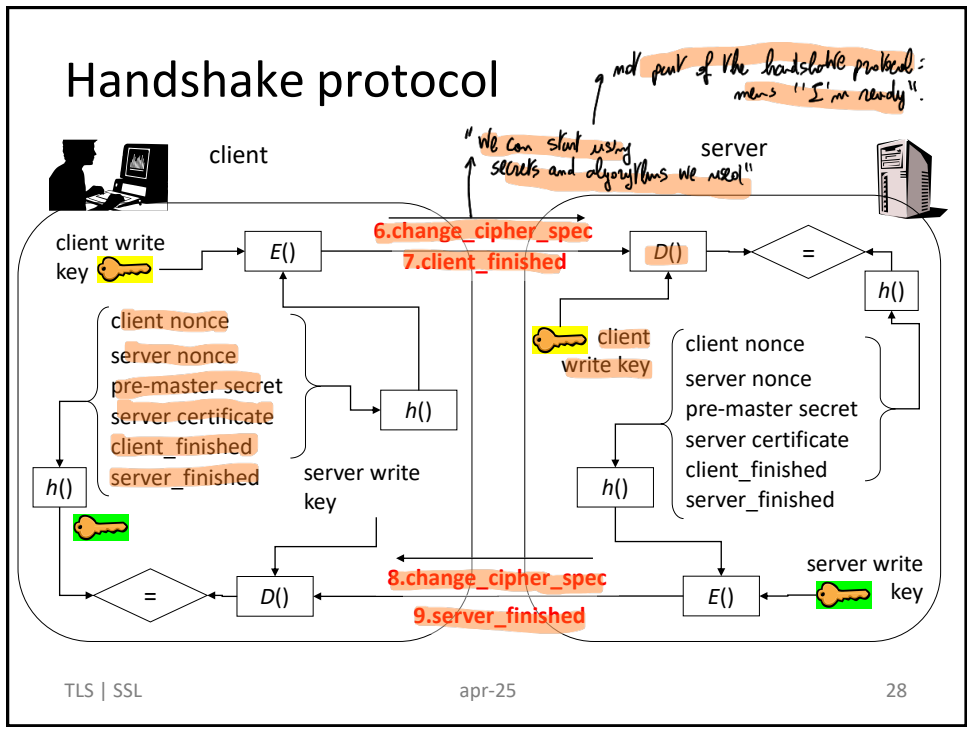
- Some tuples are recommended
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
  - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
  - …

Cipher suite

- Supported key establishment schemes
  - RSA (certified)
  - Fixed Diffie-Hellman (certified; fixed pub pars)
  - Ephemeral Diffie-Hellman (signed, dynamic pub pars)
  - Anonymous Diffie-Hellman (non authenticated)
- Supported ciphers
  - RC4, RC2, DES, 3DES, IDEA, …
- Supported MAC
  - MD5, SHA-1, SHA-2,…



27



28 ⑦ Client takes a transcript of the protocol, takes hash and encrypts it via client write key. The server receives it and check whether it is ok or not: this is to prove client owns session key and a proof that client has the same transcript, no MITM. Server does the same. This proves they agree on the same protocol no MITM.

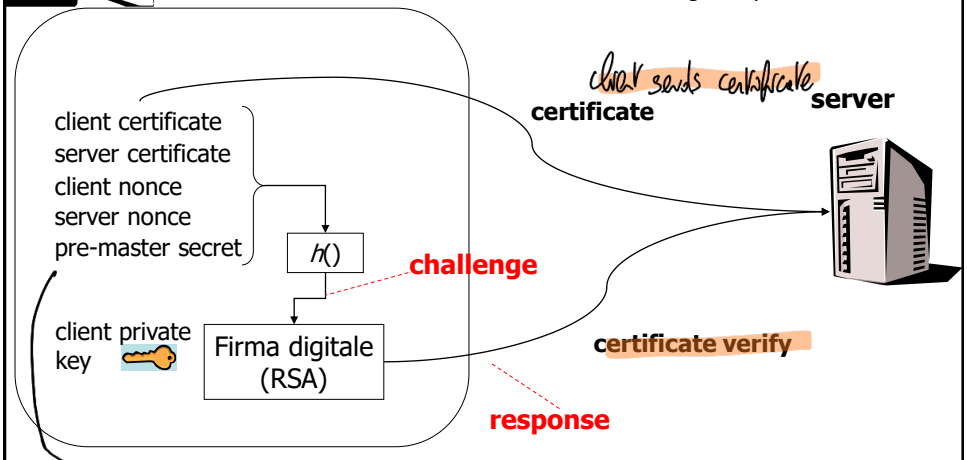
Client authentication

- Handshake Protocol authenticates the server by default
- How can the client be authenticated?
  - Typically, the client is authenticated at the application level (password, credit card number, …)
- However, SSL also supports client authentication w.r.t. the server

29

Client authentication

- Server requires client authentication by means of the certificate request message after server_hello
- Authentication is based on *challenge-response*



30

Certificate & certificate_verify message

- Client sends the a **certificate** message if the server requested it
  - No_certificate alert if required certificate is not available
- The client sends **certificate_verify** message to provide explicit proof of signing privK possession

31

Security

- Handshake Protocol
  - Nonces in client hello and server hello
    - Nonces make it possible generate a fresh master secret and avoid replay attacks
  - Certificates
    - Avoid MIM
  - Random quantities
    - Pre-master secret and nonces must be unpredictable
- Record Protocol
  - A block is numbered, authenticated and encrypted
  - Avoid block replay, reordering and substitution
  - Cipher “protects” the MAC

32

Certificate & server_key_exchange

- **Certificate**
  - Always requested but anonymous Diffie-Hellman
- **Server_key_exchange**
  - Not requested in Fixed Diffie-Hellman and RSA
  - When requested, the format depends on the chosen key exchange algorithm
  - Requested in
    - Anonymous Diffie-Hellman (p, g, Y_{svr})
    - Ephemeral Diffie-Hellman <p, g, Y_{svr}>$_{svr}$
    - RSAE when the server has RSA-signing-key <TempPubK$_{svr}$>$_{svr}$

Client_key_exchange message

- The message format depends on the chosen key establishment
  - RSA → pre-master secret
  - ANONYMOUS OR EPHEMERAL DH → (p, g, Y_{clnt})
  - FIXED DH → void payload, public pars will be sent in a certificate message (client → server)

certificate_request message

- Server may issue a **certificate_request** unless anonymous Diffie-Hellmann is used
- The message has two parameters
  - **Certificate_type**: type of digital signature and its use
    - (RSA | DSS) + (only signature | fixed Diffie-Hellmann | Ephemeral DH)
  - **Certificate_authorities:** acceptable certification authorities

SSL

HISTORY: PITFALLS AND ATTACKS

Random generator in SSL v2.0

- Pseudo-Random Bit Generator
  - keystream = PRBG(tod || pid || ppid)
  - tod = time of day; pid = process id; ppid = parent process id
- Entropy of the triple is 47-bit → seed can be guessed in 25s
- A more sophisticated attack based on system observation may be even more effective

2^{47} seconds: time of day can be guessed, if you are looking at server side it is likely to be a low value associated at bootup and ppid even more (0 or 1).

38

Attacks against implementation

Protocol was neither secure, mainly attacks against implementation

- 2011: Browser Exploit Against SSL/TLS (BEAST) attack (1) [sort of padding oracle]
  - Weakness of CBC in TLS 1.0
- 2012: Compression Ratio Info-leak Made Easy (CRIME)
  - Side-channel attack based on the compressed size of HTTP request
- 2013: Lucky13 attack
  - Timing side-channel attack with CBC (2013)
- 2014: Heartbleed attack
  - Buffer over-read attack (2014)

39

(1) You do MAC then Encrypt, so opposite is Decrypt then verify, so oracle for padding works.

SSL Quality Test

- https://www.ssllabs.com/ssltest

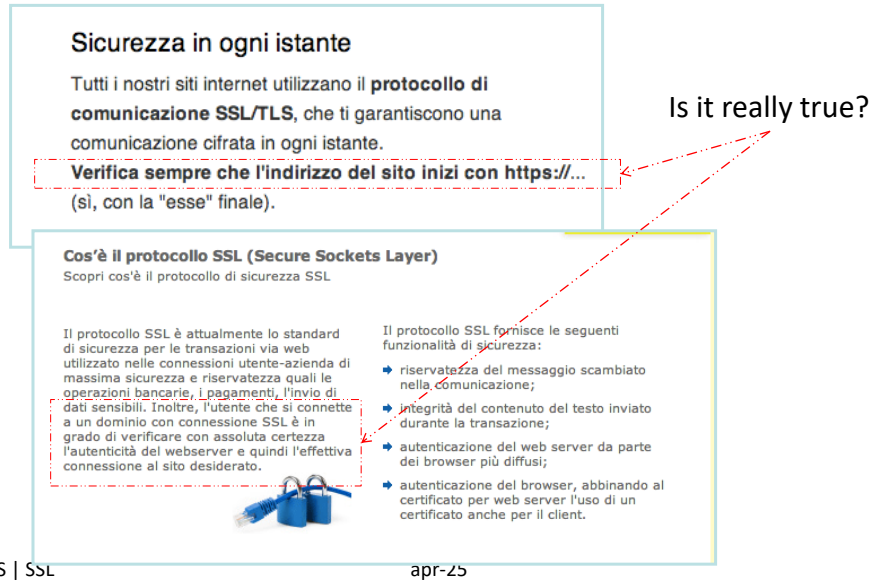
40

SSL

ON USING SSL IN E-COMMERCE

41

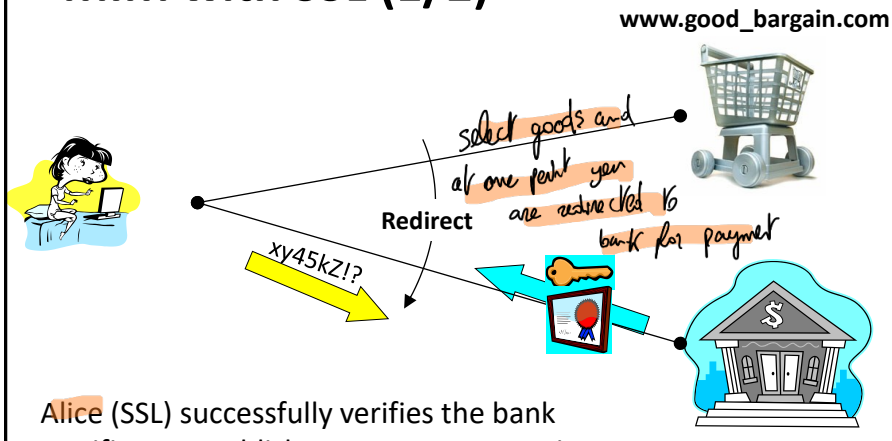
SSL in action



Is it really true?

42

MIM with SSL (1/2)

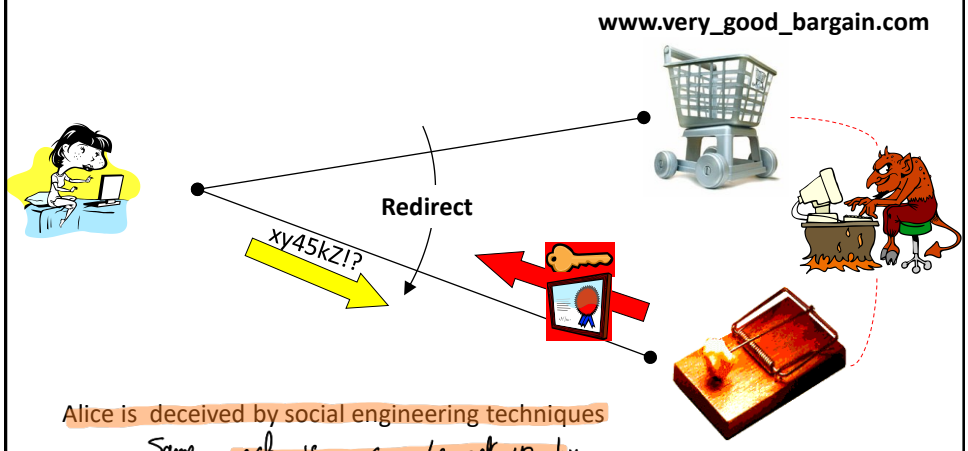


Alice (SSL) successfully verifies the bank certificate, establishes a secure connection, and sends her pwd/PIN along the connection

43

MIM with SSL (2/2)

Phishing



Alice is deceived by social engineering techniques

Same mechanism can be set up by a fraud.

Under the control of the adversary.

44

Sure sends a certificate (and it could be valid or self signed, or signed by someone unknown (the browser will alert user)). If user accepts, the connection is established and pw is transmitted.

Is it the right certificate?

- SSL operates at the transport level
- Browser controls
  - Browser warns user if the URL known to the browser is not equal to that in the certificate (mismatch)
  - Browser warns user whether a certificate is signed by an unknown CA (self-signed certificates)
  - The user has the last word
    - The clickthrough phenomenon: does the user understand security? Usability vs security
- These controls may be not sufficient for all web applications
- Browser have a largely variable behaviour in this respect (what to warn; when to warn)

45

2 problems: user auth at application level and user unaware of security concern

If by means of this fraud PIN is stole, risk allocation is on user.

E-payment: risk allocation

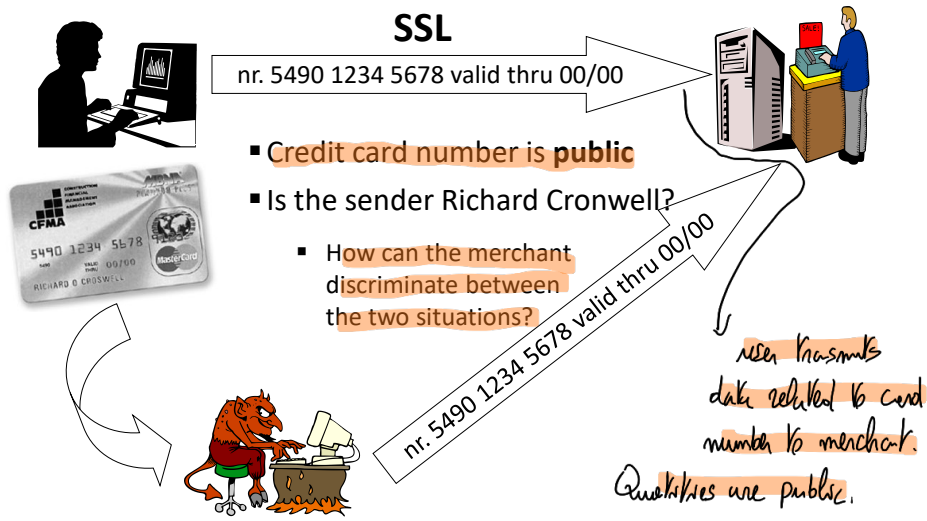
- PIN/PWD is a shared secret
- In a home banking contract, the user commits himself to protect the PIN/PWD confidentiality
- In a fraud it is evident that the PIN/PWD confidentiality has been violated
- Who is liable for?

You! So you won't be refunded

You assume responsibility of protecting PIN

In case of credit card transaction, risk allocation is different.

E-payment by credit card



user transmits data related to card number to merchant. Credentials are public.

Merchant should decide whether request is from card holder or someone else. There is no way to have certainty. In case of fraud, merchant is going to lose money, there is no way to discriminate. You cannot put responsibility on client because otherwise nobody would use it. And card issuer does not take responsibility.

E-payment by Credit Card

Decreto legislativo 22 maggio 1999, n. 185, di attuazione della direttiva 97/7/CE

Art. 8 - Pagamento mediante carta

1. Il consumatore può effettuare il pagamento mediante carta ove ciò sia previsto tra le modalità di pagamento, da comunicare al consumatore al sensi dell'articolo 3, comma 1, lettera e), del presente decreto legislativo.

2. L'istituto di emissione della carta di pagamento riaccredita al consumatore i pagamenti dei quali questi dimostri l'eccedenza rispetto al prezzo pattuito ovvero l'effettuazione mediante l'uso fraudolento della propria carta di pagamento da parte del fornitore o di un terzo, fatta salva l'applicazione dell'articolo 12 del decreto-legge 3 maggio 1991, n. 143, convertito, con modificazioni, dalla legge 5 luglio 1991, n. 197. L'istituto di emissione della carta di pagamento ha diritto di addebitare al fornitore le somme riaccreditate al consumatore.

You as a merchant have insurance

E-payment by Credit Card

- **Gli istituti di emissione**, cui compete l'autorizzazione dell'operazione di pagamento, nonché i soggetti che rendono tecnicamente possibile la transazione on-line, **sono tenuti a controllare la correttezza del numero della carta e la data della sua scadenza ma non anche la corrispondenza tra il numero fornito e l'effettivo titolare**
- Gli istituti di emissione verificano la corrispondenza tra numero della carta di credito comunicato per effettuare una transazione on-line ed il nominativo fornito da colui che la effettua.

  Ad esempio, l'**Address Verification Service (AVS)** verifica che l'indirizzo di consegna sia quello con cui il possessore della carta è registrato
- In Europa il grado di sicurezza nelle transazioni on-line è minore e quindi il commercio elettronico è destinato ad incontrare resistenze anche da parte dei fornitori di che sopportano rischi elevati

E-payment by Credit Card: risk allocation

- Il fornitore di beni o servizi on-line **è tenuto ad accollarsi il rischio** della rivalsa degli istituti di emissione qualora, in caso di uso fraudolento delle carta, questi riaccreditano le corrispondenti somme al legittimo titolare.
- La legge **non consente** al fornitore di liberarsi dall'obbligo della restituzione delle somme agli istituti di emissione qualora dimostri
  1. di avere usato tutte le cautele necessarie e possibili ad evitare l'uso fraudolento della carta di credito
  2. che il fatto è stato causato dal caso fortuito.
- I fornitori dovranno usare tutte le cautele del caso per potere, nel caso di uso fraudolento di carte di credito, perlomeno rintracciare l'illegittimo utilizzatore e rivalersi su questo.

  Le conseguenze derivanti dall'addebito delle somme riaccreditate al titolare della carta potrebbero poi essere annullate contraendo una **assicurazione** a copertura dei danni (economici) derivanti da tale circostanza.

E-payment by Credit Card

Foglio informativo sulle operazioni e servizi offerti alla clientela (CariPrato)

Caratteristiche e rischi tipici

Struttura e funzione economica

CARTE DI DEBITO e CARTE DI CREDITO

Strumenti di pagamento rilasciabili a clienti della Banca che consentono:
- Acquisto di beni;
- Prestazione di servizio presso esercenti convenzionati.
- Ottenimento di contante presso sistemi automatici o sportelli bancari convenzionati.

Funzione Bancomat: è il servizio in forza del quale la banca (emittente), attraverso il rilascio di una Carta, consente al correntista (c.d. "titolare") di effettuare prelievi di denaro — entro massimali di utilizzo stabiliti dal contratto - presso sportelli automatici (ATM) contraddistinti dal marchio Bancomat, digitando un codice segreto (c.d. P.I.N., "Personal Identification Number").

Funzione PagoBANCOMAT: è il servizio in forza del quale il correntista può compiere acquisti di beni e servizi presso esercizi commerciali convenzionati che espongono il marchio "PagoBANCOMAT", digitando il citato codice segreto.

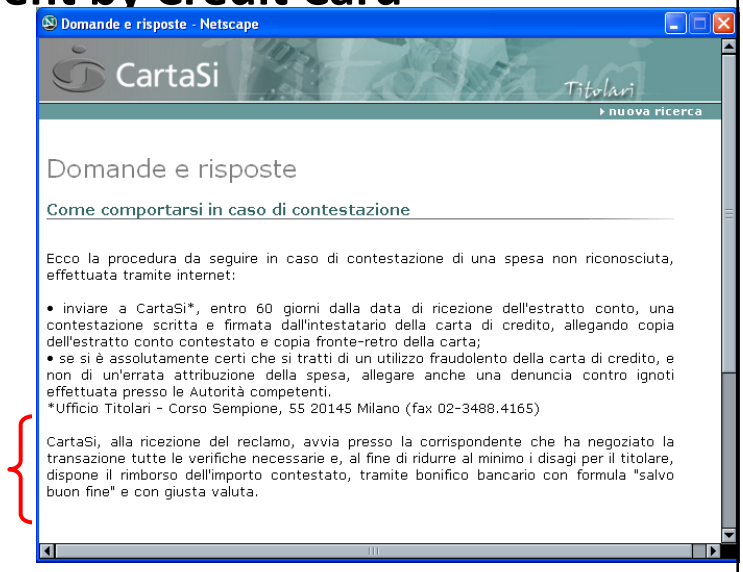
L'utilizzo del sistema di pagamento è consentito nei limiti giornaliero e mensile, entro limiti di importo contrattualmente previsti, determinato dal momento dell'emissione e dalla capienza di conto corrente al momento dell'addebito.

Principali rischi (generici e specifici)

Il rischio relativo ad eventuali utilizzi fraudolenti effettuati con le Carte di Pagamento è limitato a 150 € per evento se il Titolare ha ottemperato e rispettato quanto indicato dalla Raccomandazione della Commissione Europea del 30 giugno 1997 n. 97/489"

In sintesi il titolare è tenuto a:
- Firmare la carta nel caso che la stessa sia munita di apposita banda di scrittura;
- Osservare la massima attenzione nella custodia della carta e PIN e la massima riservatezza nell'uso del medesimo;
- Bloccare la carta nel caso di furto, smarrimento o uso fraudolento della medesima, confermando l'evento con denuncia o dichiarazione di smarrimento.

E-payment by Credit Card



SSL: Pros and Cons

- Pros
  - SSL is a well-designed, robust and secure protocol
- Cons
  - SSL protects communication only
  - User has to check security parameters
  - SSL is vulnerable to name spoofing

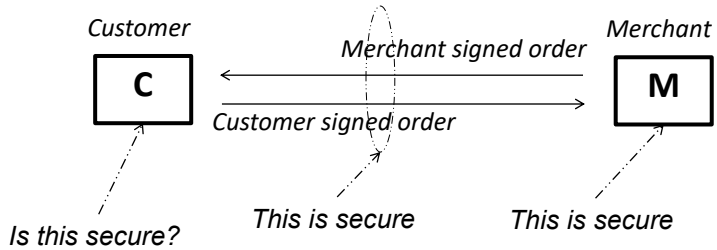
Secure Electronic Transactions

- SET was built to answer to these problems
- SET has been designed and implemented in the late 90's
  - Commissioned by Visa and Mastercard
  - Involves all (IBM, Microsoft,…)
- SET was a failure
  - Too "heavy" and too expensive
  - Specifications takes more than 1000 pages (!)
- We are interested in the risk allocation

54

Secure Electronic Transactions

- SET requires a PKI in place
- A (privK, pubK) pair is stored at M and C
- *If an order is signed by your key, you cannot repudiate it*
  - The risk is allocated on the customer
- M and C are assumed trusted devices!
  - Stealing a privK is equivalent to stealing a file



55

Secure Electronic Transactions

- Do smart cards help?
  - Loosing a piece of plastic vs. loosing a file
  - Is what you see what you sign?