Crittografia avanzata
Lezione del 2 Maggio 2011

SSL/TLS

Storia di TLS/SSL

- SSL: Sviluppato da Netscape a partire dal 1993
  - Versioni principali: 2.0 e 3.0
- TLS: sviluppato dal 1996 a partire da SSL da un WG dell'IETF
  - TLS 1.0: poco diverso da SSL 3.0 (anche indicato come SSL 3.1)
  - Successivamente: versioni 1.1 e 1.2 (anche indicate come SSL 3.2 e 3.3)
    - Modifiche nella parte crittografica
    - Ognuna rende obsoleta la precedente
- DTLS: su datagram, es. UDP (RFC 4347)

Caratteristiche e funzionalità

- SSL fornisce autenticazione, riservatezza e integrità
- Non fornisce non-repudiation
  - Anche registrando la sessione, dato che vengono usate chiavi effimere condivise, le due parti potrebbero produrre registrazioni false
  - Comunque non è pensato per questo uso
- Essendo pensato per trasportare dati di un livello applicativo superiore, divide i dati in “fragments” trattati separatamente; ognuno può essere:
  - Autenticato
  - Compresso
  - Cifrato
- Self-delimiting: non si basa sul protocollo sottostante per capire inizio e fine dei propri messaggi
- Funzionalità “aggiuntive”: session renegotiation, session resumption,

Record SSL

- Un fragment può essere autenticato (con MAC), compresso, cifrato, e poi utilizzato per costruire un SSL record che comprende:
  - Type field
  - Version field
  - Lenght field
  - Fragment field

+	Byte +0	Byte +1	Byte +2	Byte +3
Byte 0	Content type			
Bytes 1..4	Version		Length	
	(Major)	*(Minor)*	*(bits 15..8)*	*(bits 7..0)*
Bytes 5..(*m*-1)	Protocol message(s)			
Bytes *m*..(p-*1*)	MAC (optional)			
Bytes *p*..(q-*1*)	Padding (block ciphers only)			

Sottoprotocolli

- SSL Handshake protocol (core)
- SSL Change chipher spec. protocol
- SSL Alert protocol (segnalazione)
- SSL Application Data protocol
- Definiti nel campo “Content type”

Attivazione di SSL

- Server che accettano connessioni cifrate o non cifrate hanno due opzioni:
  - Porte separate: una porta in chiaro (es. 80) e una su SSL (es. 443)
  - Upward negotiation: protocollo per l'attivazione di SSL (es. in SMTP, STARTTLS; in HTTP, RFC 2817)
    - Fonte di diverse vulnerabilità
    - È la strategia preferita: non consuma porte, è più flessibile

SSL session/connection

- SSL session: logica analoga alle Security Associations
  - Creata da un SSL Handshake, definisce un insieme di parametri crittografici
  - Viene coordinato e mantenuto uno stato della sessione
- SSL connection: utilizzata per trasmettere dati, ad es. su una connessione TCP
- Più SSL connections possono condividere la stessa sessione
  - Può migliorare molto l'efficienza
  - Fra due entità ci possono essere molte connessioni contemporanee, raramente ci sono più sessioni contemporanee

Stato

- Ci sono due stati: corrente e pendente
- Vengono mantenuti separati lo stato di lettura e quello di scrittura
  - Totale: quattro
- Durante l'handshake vengono scambiati dei messaggi di ChangeCypherSpec:
  - Chi li riceve copia il pending read nel current read
  - Chi li invia copia il pending write nel current write

Stato: parametri di sessione

- **session identifier**: Arbitrary byte sequence chosen by the server to identify an active or resumable session state (maximum length is 32 bytes)
- **peer certificate**: X.509v3 certificate of the peer (if available)
- **compression method**: Data compression algorithm used (prior to encryption)
- **cipher spec**: Data encryption and MAC algorithms used (together with cryptographic parameters, such as the length of the hash values)
- **master secret**: 48-byte secret that is shared between the client and the server
- **is resumable**: Flag indicating whether the SSL session is resumable, meaning that it can be used to initiate new connections
-

Stato: parametri di connessione

- **server and client random**: Byte sequences that are chosen by the server and client for each connection
- **server write MAC key**: Secret used in MAC operations on data written by the server
- **client write MAC key**: Secret used in MAC operations on data written by the client
- **server write key**: Key used for data encrypted by the server and decrypted by the client
- **client write key**: Key used for data encrypted by the client and decrypted by the server
- **initialization vectors**: If a block cipher in CBC mode is used for data encryption, then an IV must be maintained for each key. This field is first initialized by the SSL Handshake Protocol. Afterwards, the final ciphertext block from each SSL record is preserved to serve as IV for the next record.
- **sequence numbers**: SSL message authentication employs sequence numbers. This basically means that the client and server must maintain a sequence number for the messages that are transmitted or received on a particular connection. Each sequence number is 64 bits long and ranges from 0 to 264 − 1. It is set to zero whenever a CHANGE CIPHER SPEC message is sent or receved.

Pre-master secret

- Il pre-master secret viene concordato nella fase di handshake
  - Se si usa RSA viene inivata cifrata con la chiave pubblica del server
  - Altrimenti, si possono ad esempio usare diverse versioni di DH
    - Fixed (i parametri del server sono fissati dal certificato)
    - Ephimeral (i parameteri sono generati e autenticati, ad es. firmandoli)
    - Anonymous (MitM)
  - Esistono altre possibilità (es. Fortezza)

Ephimeral Diffie Hellman

- Ephimeral
  - Evita il riutilizzo di parametri
  - Fornisce la Perfect Forward Secrecy (ripsetto alla master key, non alla chiave pubblica)
- Ogni versione comporta piccole differenze nella fase di handshake

Master secret

- master_secret =

  MD5(pre_master_secret + SHA('A' + pre_master_secret + ClientHello.random + ServerHello.random)) +

  MD5(pre_master_secret + SHA('BB' + pre_master_secret + ClientHello.random + ServerHello.random)) +

  MD5(pre_master_secret + SHA('CCC' + pre_master_secret + ClientHello.random + ServerHello.random))
- A, BB, CCC sono parametri: 0x41, 0x4242, e 0x434343, e "+" è la concatenazione
- I random seed servono a rendere più difficile la criptoanalisi

Key blocks

- Il master secret è usato come seed per una funzione pseudo causale per generare chiavi effimere
- key_block =

  MD5(master_secret + SHA('A' + master_secret + ServerHello.random + ClientHello.random)) +

  MD5(master_secret + SHA('BB' + master_secret + ServerHello.random + ClientHello.random)) +

  MD5(master_secret + SHA('CCC' + master_secret + ServerHello.random + ClientHello.random)) +

  [...]
- Proseguendo si generano tutti i byte necessari per gli altri parametri di sessione: client_write_MAC_secret, server_write_MAC_secret, client_write_key, server_write_key, client_write_IV, server_write_IV

SSL Record protocol

- Frammentazione: blocchi di 2^{14} byte (16k) o meno
- Compressione
- Autenticazione (aggiunta di un MAC)
- Cifratura
- Aggiunta dell'header
- Autenticazione e cifratura sono fatte in base a un unico CipherSpec: es.

  SSL DH RSA WITH 3DES EDE CBC SHA
  - RSA per key exchange
  - SHA-1 per il MAC
  - 3DES CBC per la cifratura (Encrypt/Decrypt/Encrypt)

EtA o AtE?

- Prima cifrare e poi autenticare, o prima autenticare e poi cifrare?
- Generalmente, è più sicuro EtA
- AtE è sicuro se si usa un algoritmo a blocchi in CBC o uno stram cihper (XOR)
-

MAC

- SSL MAC_k (SSLCompressed) =h(k || opad || h(k || ipad || seq_number || *type* || *length* || *fragment*))
- La parte in italico è compressa
- È un “predecessore” dell'HMAC
  - Fra chiave e ipad o opad usa la concatenazione anziché l'XOR
  - Ipad e opad sono diversi da HMAC
  - Nota: manca “Version”

Cifratura

- Semplice con uno stream cihper:
  - Niente padding o IV, solo uno stato da mantenere
- Block cipher:
  - Padding, con byte ognuno dei quali indica la lunghezza del padding in byte (l'header indica la lunghezza del fragment)
  - IV (parte dello stato)
- Attacco al padding: http://www.cs.bham.ac.uk/~mdr/teaching/modules03/security/students/SS8a/SSLTLS.html

Handshake protocol

- Nota: la struttura generale è quella del Record
- Il type è 22

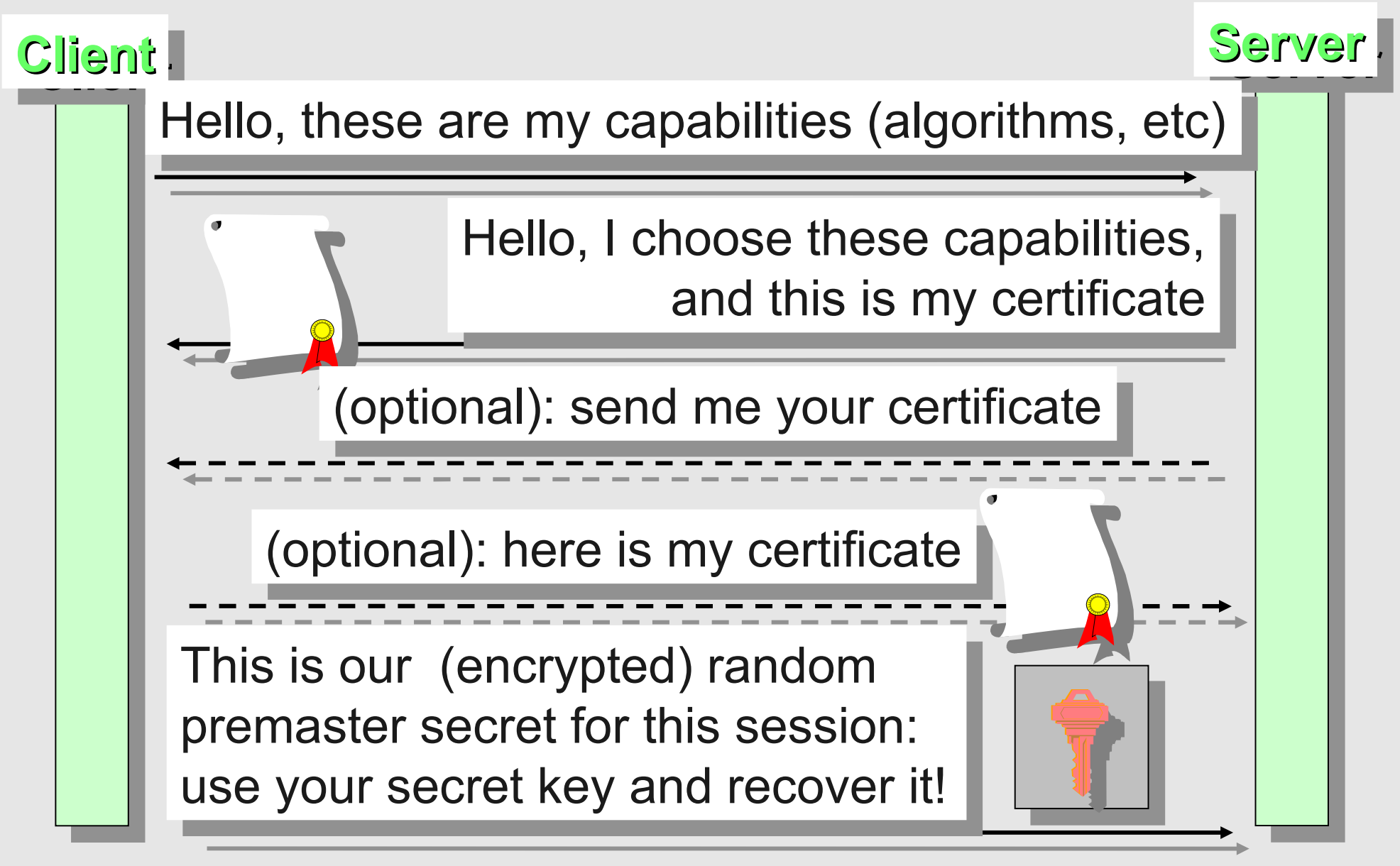
+	Byte +0	Byte +1	Byte +2	Byte +3
Byte 0	22			
Bytes 1..4	Version		Length	
	(Major)	*(Minor)*	*(bits 15..8)*	*(bits 7..0)*
Bytes 5..8	Message type	Handshake message data length		
		(bits 23..16)	*(bits 15..8)*	*(bits 7..0)*
Bytes 9..(*n*-1)	Handshake message data			
Bytes *n*..(*n*+3)	Message type	Handshake message data length		
		(bits 23..16)	*(bits 15..8)*	*(bits 7..0)*
Bytes (*n*+4)..	Handshake message data			

Messaggi

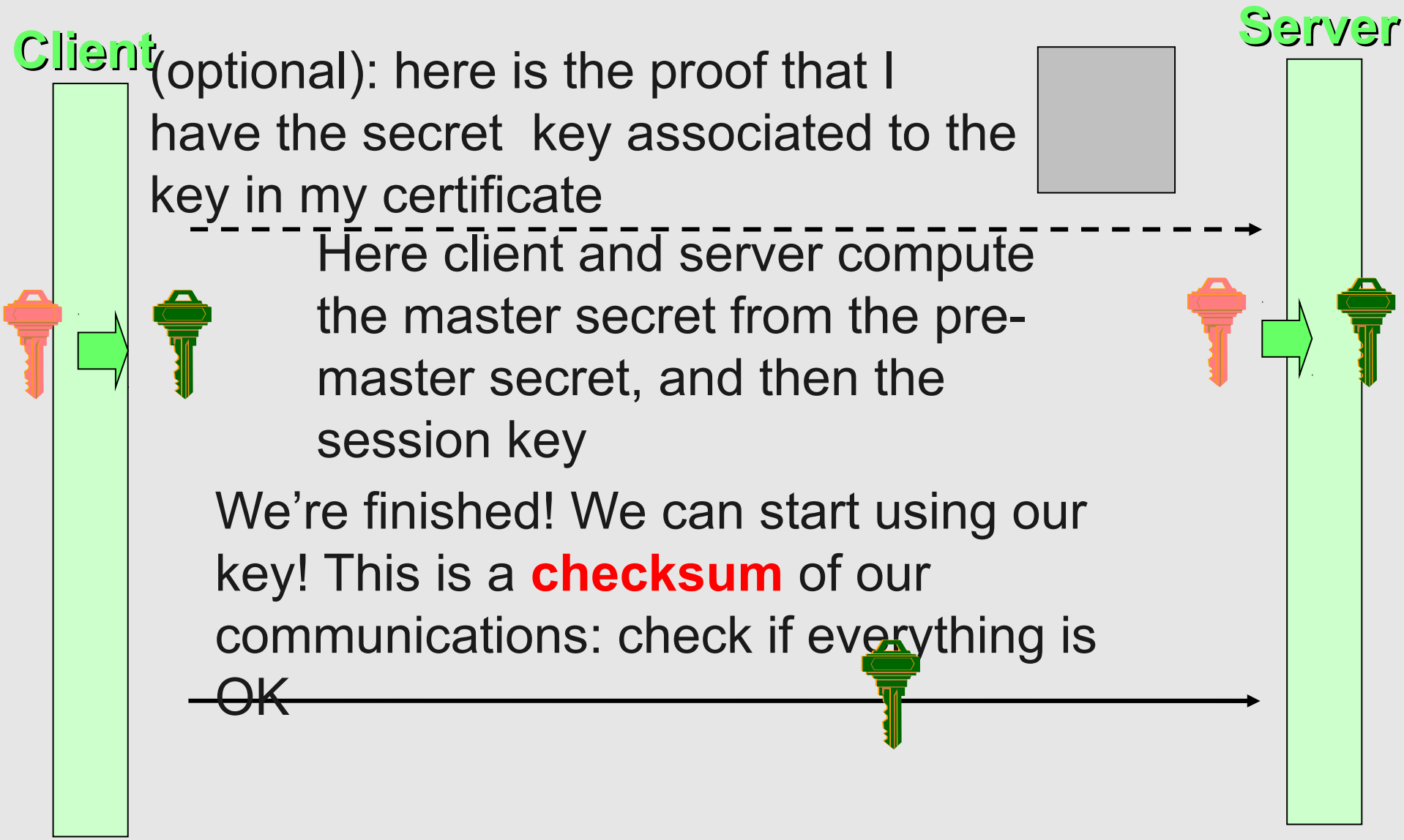
- Permettono di effettuare le varie fasi dell'handshake

Message Types	
Code	**Description**
0	HelloRequest
1	ClientHello
2	ServerHello
11	Certificate
12	ServerKeyExchange
13	CertificateRequest
14	ServerHelloDone
15	CertificateVerify
16	ClientKeyExchange
20	Finished

SSL session opening (1)



SSL session opening (2)



Sequenza

Client Hello

Server hello
Certificate
Server Key Exchange
Certificate Request
Server Hello Done

Certificate
Client Key Exchange
Certificate Verify
Change Cipher Spec
Finished

Change Cipher Spec
Finished

Finished

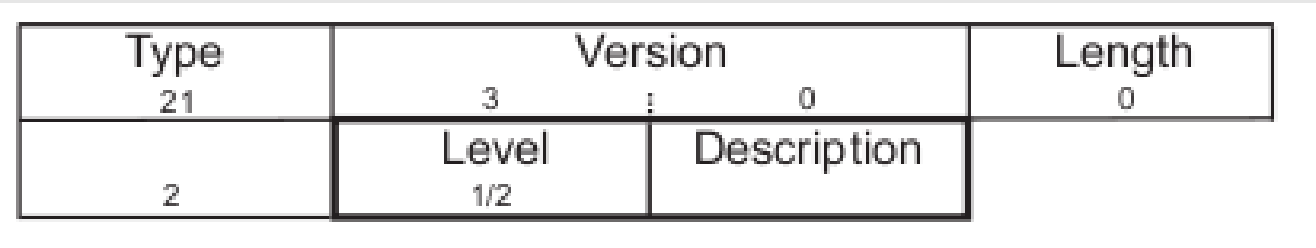
- È il primo messaggio protetto con il nuovo stato
- Contiene una checksum:

  h(k || opad || h(handshake_messages || sender || k || ipad))
- handshake_messages è la concatenazione di tutti i messaggi scambiati fino a quel punto
- Sender è un codice che indica il client o il server

CertificateVerify

- Proof of possession della chiave privata
- Si cifra con la chiave privata un hash derivato dalla concatenazione dei messaggi dell'handshake fino a quel punto
  - Simile al Finished
- Nei messaggi ClientHello e ServerHello sono inviati dei numeri casuali, quindi il PoP è unico per la sessione

Alert Protocol



Code	Level type	Connection state
1	warning	connection or security may be unstable.
2	fatal	connection or security may be compromised, or an unrecoverable error has occurred.

Alert Protocol: types

Code	Description	Level types	Note
0	Close notify	warning/fatal	
10	Unexpected message	fatal	
20	Bad record MAC	fatal	Possibly a bad SSL implementation, or payload has been tampered with e.g. FTP firewall rule on FTPS server.
21	Decryption failed	fatal	TLS only, reserved
22	Record overflow	fatal	TLS only
30	Decompression failure	fatal	
40	Handshake failure	fatal	
41	No certificate	warning/fatal	SSL 3.0 only, reserved
42	Bad certificate	warning/fatal	
43	Unsupported certificate	warning/fatal	E.g. certificate has only Server authentication usage enabled and is presented as a client certificate
44	Certificate revoked	warning/fatal	
45	Certificate expired	warning/fatal	Check server certificate expire also check no certificate in the chain presented has expired
46	Certificate unknown	warning/fatal	
47	Illegal parameter	fatal	

- 0: usato per chiudere la connessione, per evitare attacchi di troncamento

Analisi

- Nel 1996, l'analisi di Wagner e Schneier su SSL 3.0 ha trovato vulnerabilità minori, ma nel complesso è stato un buon protocollo
  - Con alcuni problemi di implementazione, ad esempio nel PRNG
- Alcuni limiti e problemi sono stati trovati negli algoritmi crittografici utilizzati
- Un attacco (semi) teorico di chosen ciphertext contro RSA e PKCS#1 1.0 richiede un numero di prove dell'ordine di un milione
  - Facilmente riconocibile
  - Richiede di usare il server come oracolo
  - Il server, di fronte a un errore nella formattazione di un blocco RSA, deve rispondere con una stringa casuale, non facendo l'oracolo
  - Ha portato a PKCS#2.0 e poi, per un ulteriore attacco, a PKCS#1 2.1
  - Ha dimostrato che algoritmi anche provabilmente sicuri possono essere attaccati per i limiti delle implementazioni pratiche
- Vulnerabilità utilizzabili praticamente nel CBC block padding (vedi 148)
  - Corretta in TLS 1.1

Differenze con TLS

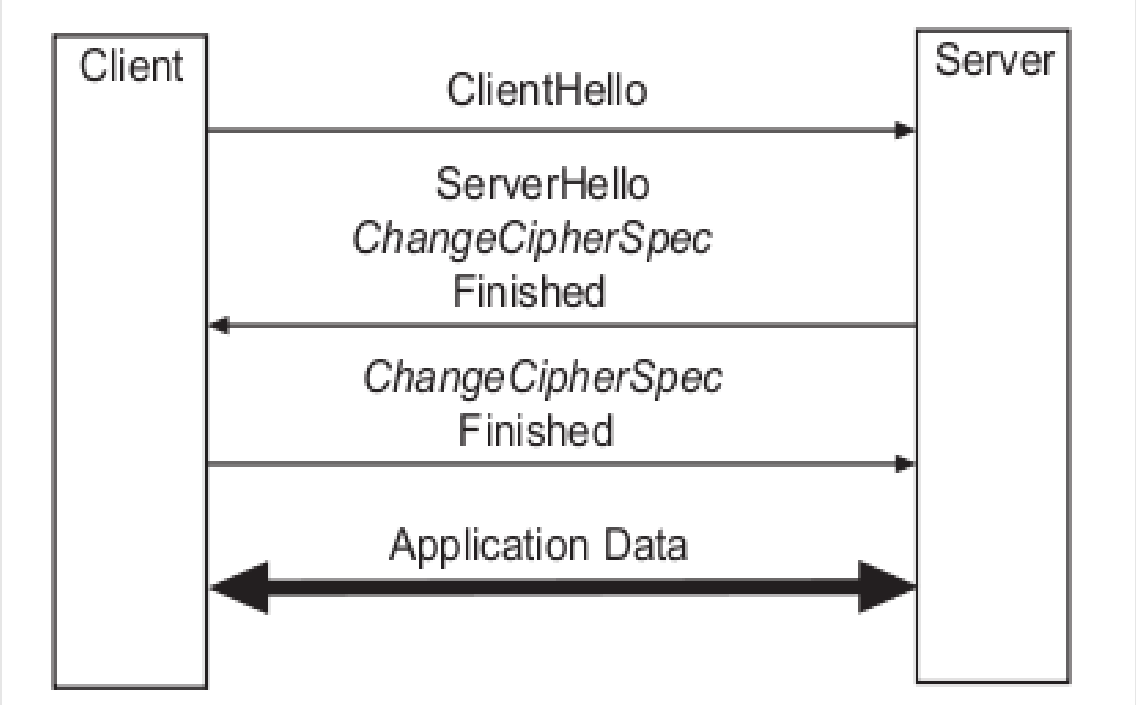
- 1.0: poche differenze:
  - Funzioni di generazione del key material
  - HMAC
  - Diverse ciphersuite
  - Certificate chain parziali (fino a una CA intermedia)
  - Nuovi alert type
  - Semplificazione del CertificateVerify (solo concatenazione e hash)
  - ...

TLS 1.1 e 1.2

- 1.1
  - Nuove ciphersuite
  - Protezione per il CBC padding
  - ...
- 1.2:
  - PRF è un parametro
  - Nuove ciphersuite (in particolare, SHA-256)
  - ...

Session resumption

- Si utilizza un handshake in cui viene solo ripetuta la fase di hello/ChangeCipherSpec
  - Anche per una nuova connessione



TLS e smart card

- Quando si usa una smart card, la chiave privata rimane **sempre** sulla smart card
- La creazione di una nuova sessione richiede un accesso alla chiave privata
  - Un problema frequente è che non riutilizzando correttamente il session-context, ad ogni accesso al server venga richiesto il PIN della carta (eventualmente dalla cache, ma non è detto che sia un'opzione desiderabile)

Attacchi

- TLS renegotiation
  - http://www.educatedguesswork.org/2009/11/unde
- Papers:

Certification Authorities

Il concetto di Trust

- Nella sicurezza si usano due tipi di “fiducia” diversi:
  - Quello al quale siamo abituati, dotato di molte sfumature, ad esempio quando valutiamo la fiducia in una persona, in uno strumento, in un report...
  - Quello al quale si riferiscono i modelli matematici, molto più preciso:
    - Ci si fida del fatto che un'entità svolga un preciso compito in modo corretto;
    - La fiducia è incondizionata: o c'è o non c'è

Il “trucco” delle Trusted Third Parties

- Molti problemi di sicurezza sono difficili o impossibili da risolvere completamente
- Molti problemi si risolvono magicamente con l'introduzione di una terza parte fidata:
  - È terza, non ha interesse nel problema
  - È fidata, svolgerà incondizionatamente bene il proprio compito
- Ma quando si passa alla pratica...

Esempio: sicurezza multilaterale

- Indica un contesto in cui entità diverse hanno requisiti di sicurezza diversi (contrastanti) e il sistema/protocollo deve soddisfarli il più possibile
- Prima possibilità: self enforcing protocols
- Seconda possibilità: si affida la TCB a una terza parte fidata

Problemi pratici

- La parte fidata spesso non è terza
- Non svolge il suo compito in modo perfetto
  - Responsabilità?
- Fidata per chi?
  - Nella maggior parte dei casi, chi sceglie la TTP non è chi si deve fidare

Firma digitale ≠firma autografa

- La firma digitale è legata all'uso della chiave privata
- La firma autografa è legata all'uso della mano
  - La mano è parte inseparabile della persona
  - La chiave privata no, ogni utilizzo reale è sempre mediato da strumenti complessi

Cosa significa firmare?

- Nella normativa italiana, il concetto è talmente scontato da non trovare traccia di una definizione in più di un secolo di normativa
  - La firma digitale può indicare:
  - Un autore
  - Una volontà
  - Una certificazione
    - Di integrità
    - Di origine (web of trust)
    - “Quanto” si certifica?

Chiave pubblica?

- Problemi:
  - come distribuire la chiave pubblica?
  - come essere sicuri che una chiave pubblica sia effettivamente associata al mittente del messaggio?
  - Come accertarsi se una chiave pubblica è ancora valida?

Distribuzione delle chiavi

- Scambio di chiavi personale
- Repository
- Verifica in linea
- Il problema principale è che le due entità non sono state precedentemente in contatto fra loro, serve una terza parte fidata (Trusted Third Party) che faccia da intermediario
  - È un problema generale dell'autenticazione

Web of trust (1)

- A conosce B ed è disposto a garantire sulla sua identità; A firma la chiave pubblica di B
- C conosce A e la sua chiave pubblica; C “si fida” di A per quanto riguarda la firma delle chiavi
- C si fida dell’autenticità della chiave di B firmata da A

Web of trust (2)

- Vantaggi:
  - non serve alcun tipo di infrastruttura
  - gli utenti scelgono di chi fidarsi
- Svantaggi:
  - la transitività non è ragionevole oltre due-tre passaggi
  - non è utilizzabile per contatti con entità con le quali non si hanno “conoscenze comuni”
  - responsabilità? (non sono parte del meccanismo)

Legare la chiave alla persona

- Le Certification Authority sono delle “Terze parti fidate” che garantiscono l’associazione fra un’entità e una coppia di chiavi
- Generano un certificato, cioè un documento firmato contenente:
  - un’identificazione dell’entità certificata
  - la chiave pubblica dell’entità
  - un periodo di validità
- È un Web of trust a due soli livelli

Certification Authority (1)

- Viene scelta un'entità della quale "tutti si fidano": la Certification Authority (CA)
- La CA ha il compito di assicurarsi dell'identità degli utenti e di firmarne le chiavi pubbliche (certificati)
- Per gli utenti è necessario conoscere solo la chiave pubblica della CA
- Per la CA è necessario conoscere solo la chiave pubblica degli utenti

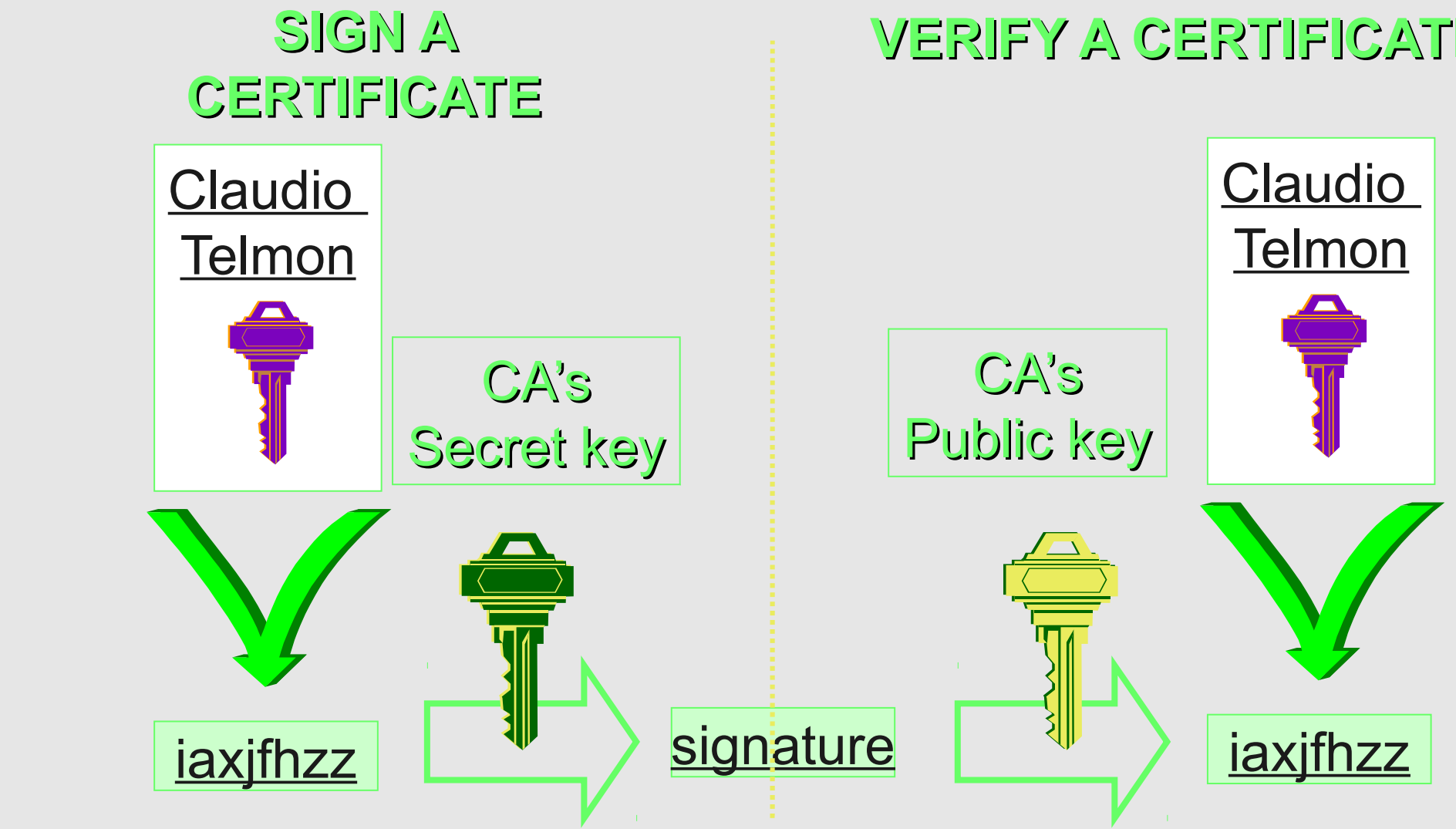
Certification Authority (2)

- Vantaggi:
  - è possibile contattare entità sconosciute;
  - è chiaro in chi si ha fiducia
  - la gestione è più semplice
- Svantaggi
  - ci si deve fidare di chi si fidano gli altri
  - serve un'infrastruttura per distribuire i certificati

I certificati

- Associano un'identità (un server, una persona) a una chiave pubblica
- Sono firmati da una CA
- Contengono i dati necessari per rendere unica l'identità
- Contengono opzionalmente altri dati:
  - un periodo di validità
  - l'associazione fra l'identità e un'entità reale
  - …

Certificati



Cosa garantisce la CA?

- La CA può garantire che un'entità è in possesso di una chiave
  - Proof of possession (importante in molti schemi)
  - Sia "legittimamente" registrata (es. per un ruolo)
- La CA non può garantire che la chiave
  - non venga consegnata anche a un'altra persona
  - non venga "rubata"da un'altra persona
  - non venga usata da un'altra persona

Revoca dei certificati

- Problema:
  - le chiavi private possono essere compromesse
  - entità possono cambiare caratteristiche (es. licenziamenti)
  - non è possibile prevedere questi eventi all'atto della creazione del certificato
- Che ne è di un certificato associato a una coppia identità/chiave non più valida?

Revoca dei certificati

- Se una chiave privata di utente o di CA è compromessa, viene revocata e il certificato è inserito in una Certificate Revocation List…
- …solo quando la compromissione è scoperta

Certificate Revocation Lists: CRL

- Elencano i certificati che sono stati revocati
- Devono essere aggiornate e disponibili
- Richiedono un meccanismo di sincronizzazione, o la verifica online
- Si scarica una grande quantità di dati per verificare un solo certificato
- Espongono inutilmente informazioni sulle revoche

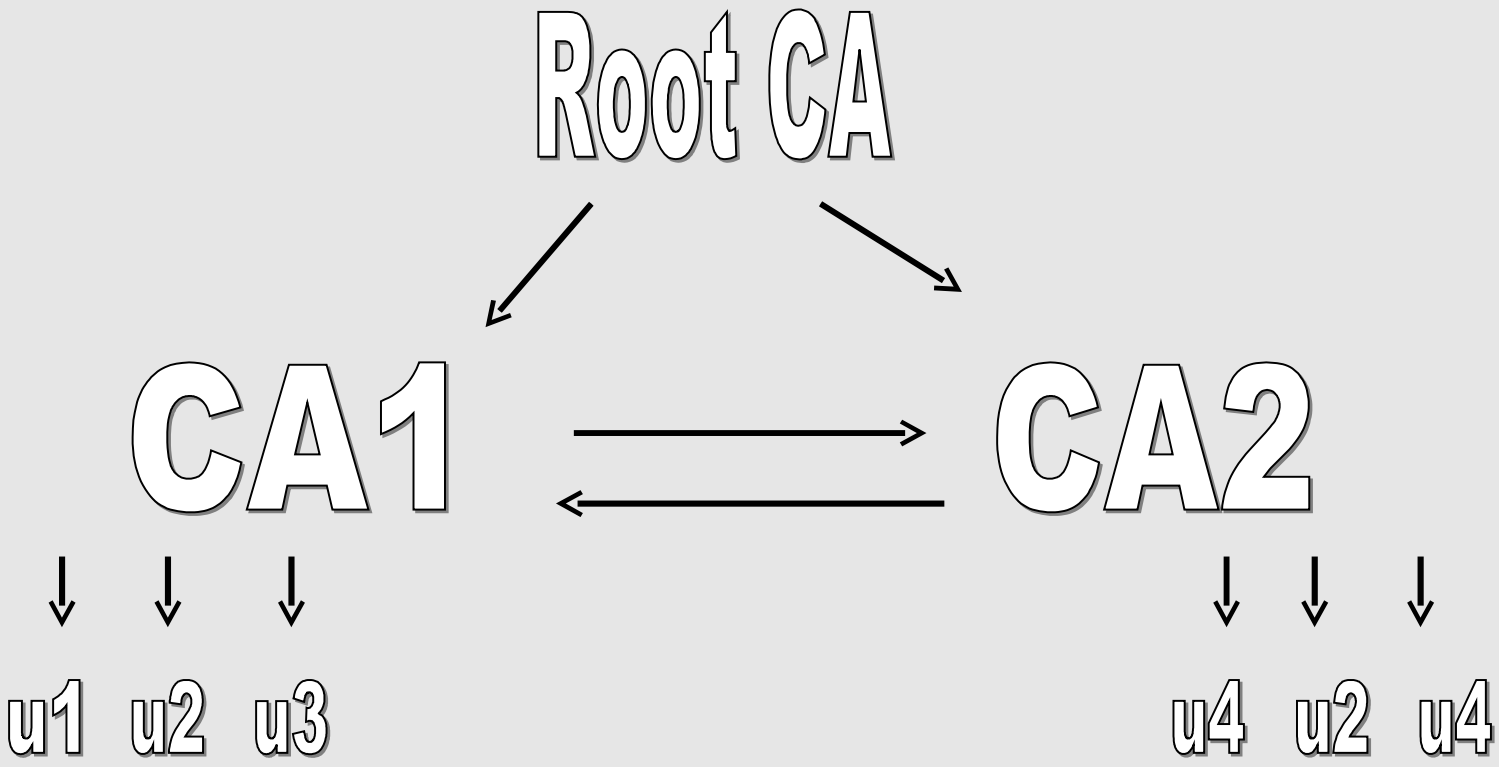
OCSP

- Online Certificate Status Protocol (RFC 2560)
- Permette la verifica online del singolo certificato richiedendo online ad un server autorizzato
  - Tipicamente della CA che ha emesso il certificato
  - Indicato nel certificato mediante il campo “Authority Information Access”
- Permette alla CA di fornire informazioni aggiuntive
- Fornisce al servizio l'informazione che un certificato è stato usato in un certo istante su un server/servizio/...

Certification path

- Le CA possono firmare certificati per altre CA, creando delle catene di certificati
  - Tipicamente, si tratta di CA “sussidiarie”, es. specifiche per tipologia di certificato
- Per fidarsi di un certificato bisogna fidarsi di tutta la catena a monte

Firme reciproche: modello gerarchico e piatto



Servizi di timestamp

- Un servizio di timestamp firma una coppia (hash del documento, orario)
- Garantisce l'esistenza del documento a una data
- Limita i danni della compromissione di una chiave
  - in particolare la chiave di una CA

Politiche di certificazione

- Definiscono:
  - chi genera le chiavi
  - lunghezze minime e massime delle chiavi
  - requisiti per l'identificazione delle parti
  - requisiti di sicurezza per la generazione di certificati e CRL nonché i controlli
  - frequenza di generazione delle CRL
  - meccanismi di revoca delle chiavi
  - vincoli sui nomi o attributi degli utenti
  - meccanismi di audit

PKI: Public Key Infrastructure

- È un'infrastruttura per:
  - generare certificati
  - distribuire i certificati
  - rendere disponibili le condizioni (politiche di sicurezza) di creazione dei certificati e di cross certification
  - generare e distribuire le CRL
  - gestire il ciclo di vita delle chiavi
  - gestire il timestamp

Entità principali coinvolte in una PKI

- Utente:
  - richiede la creazione di un certificato per una propria chiave
  - richiede il certificato (o una CRL) di un altro utente per verificarne la chiave
- Certification Authority:
  - genera certificati per altri utenti, anche altre CA, secondo una certa politica

Entità coinvolte in una PKI (2)

- Registration Authority:
  - è un tramite fra gli utenti e la CA
  - identifica gli utenti ma non produce certificati
- Altre CA
  - può essere necessario accettare certificati emessi da un'altra CA
  - per questo le due CA effettuano una Cross Certification
  - non è un semplice riconoscimento reciproco ma l'accettazione delle rispettive politiche

Protocolli e formati

- Tutte le interazioni fra le varie entità sono effettuate secondo protocolli standard (tipicamente PKIX) e con formati standard (tipicamente PKCS)
- Esiste però una grande varietà, eccezioni, famiglie di standard alternative...
  - Sopratutto nell'interazione con l'utente, la parte meno “sotto controllo”

Compiti della Registration Authority

- controlla l'identità degli utenti che richiedono la generazione di certificati
- fornisce alla CA i dati sull'identità del richiedente e la richiesta
- riceve e verifica i certificati forniti dalla CA
- consegna i certificati agli utenti

Generazione delle chiavi

- Le chiavi possono essere generate:
  - dall'utente; la CA dovrà verificarne la corrispondenza alla politica
  - da un apposito meccanismo che le fornisce all'utente e poi distrugge la propria copia
- La chiave privata deve essere nota solo all'utente salvo meccanismi di key escrow/recovery

Verifica delle firme

- Per la verifica di una firma un utente si deve procurare tutta la catena di certificati
- Ogni certificato deve essere verificato rispetto alla più recente CRL
  - O mediante OCSP; in pratica, dato che comporta dei ritardi, molto spesso è disattivato...
- I certificati possono essere accumulati localmente in una cache

Compromissione delle chiavi

- Se si sospetta la compromissione di una chiave privata questa deve essere subito revocata
- La CA deve verificare la correttezza e legittimità delle richieste di revoca
- La compromissione della chiave di una CA comporta l'invalidazione di tutti i certificati firmati
- La compromissione può essere scoperta in ritardo
  - Le responsabilità di quanto accade nel frattempo sono definite più dalle normative che dagli standard

Rigenerazione dei certificati

- La durata di validità di un certificato non coincide in generale con quella della firma della CA
- Allo scadere della validità della firma della CA è necessario “rigenerare” i certificati
  - È un'attività onerosa, è meglio schedularla in modo scaglionato

Archiviazione di certificati e CRL

- È necessario archiviare certificati e CRL
  - Per verifiche anche dopo la data di scadenza
- L'archiviazione oltre la data di scadenza pone molti problemi:
  - possono essere usati per dimostrare la validità di una firma? Se nel frattempo la chiave o l'algoritmo non sono più validi...
  - Possono essere modificati dopo la compromissione della firma della CA?
  - ...

Sospensione dei certificati

- È simile alla revoca, ma è temporanea
- •È meno critica, può essere accettata più facilmente (es. PIN e richieste autofirmate) e poi verificata
- •Non è implementata dai prodotti più diffusi e utilizzati
  - In generale si preferisce revocare ed eventualmente riemettere

Responsabilità delle CA

- Un comportamento della CA non corrispondente alle politiche o negligente può causare danni notevoli
- I contratti di servizio delle CA commerciali indicano sempre una responsabilità assai ridotta
  - Poco letti
  - Finora pochi casi noti, ma non è detto che le violazioni verrebbero scoperte

Esempio: falsi certificati Microsoft

- Nel Gennaio 2001 Verisign ha emesso due certificati “class 3 – code signing” a nome Microsoft Corporation ad una persona che si è fatta passare per un dipendente Microsoft autorizzato
- I certificati sono stati revocati in Marzo, l'advisory relativo è di Aprile MS01-017 (vale la pena leggerlo)
  - Non erano trusted in modo automatico
  - Non era banale verificare la revoca
  - Non è noto (pubblicamente) se e come siano stati usati

Formato dei certificati: X.509

- 1988: versione 1, come attributo nell'ambito dello standard X.500
- Revisioni nel 1993 (v.2) e nel 1996 (v.3)

Formato di X.509

- I campi base per tutte le versioni sono:
  - numero di serie
  - identificatore algoritmo di firma
  - nome di chi ha emesso il certificato, issuer (opz. identificatore unico)
  - validità (inizio, fine)
  - nome del soggetto (opz. id. unico)
  - chiave pubblica del soggetto
  - firma del certificato

Estensioni di X.509 v.3

- L’indicatore di criticità stabilisce che chi elabora il certificato deve scartare il certificato se non sa come interpretare questi campi
- I campi devono essere “registrati” in modo da essere riconosciuti univocamente
- Esistono estensioni standard ma sono possibili estensioni private

Estensioni di X.509 v.3 (2)

- Le estensioni sono composte da:
  - un identificatore di estensione
  - un indicatore di criticità
  - un valore in ottetti (byte)
- Si possono aggiungere estensioni senza modificare il formato di X.509

I formati “fisici”

- DER: Distinguished Encoding Rules: formato binario usato (anche) per i certificati X.509
- PEM: essenzialmente, DER in base64
- PKCS#7: dati firmati, spesso con il certificato, secondo la famiglia PKCS
- PKCS#12: può contenere la chiave privata oltre al certificato, usata ad es. per backup e Export/Import
  - Erroneamente chiamata spesso “certificato”

Informazioni sulla chiave e sulle politiche

- Identificatori di chiave della CA o del soggetto: un soggetto può avere più chiavi o più certificati associati a una chiave, ad es. consecutivi
- Uso della chiave:
  - firma digitale, non repudiation, cifratura di chiavi, cifratura di dati, key agreement, firma di certificati, firma di CRL

Certificati diversi per usi diversi

- Certificati diversi per ruoli diversi
- Non solo persone fisiche: es. SSL
- Limitazioni alla validità dei certificati
  - Tutto molto legato alle normative e scelte implementative nazionali
- Ruoli: molto importanti in particolare per le aziende

Key Recovery

- L’azienda si garantisce la possibilità di accedere ai documenti cifrati
- Offre un servizio agli utenti
- Non si garantisce la possibilità di falsificare le firme
  - Servono due coppie di chiavi per utente

Tutte le chiavi saranno compromesse

- Le capacità di calcolo aumentano
- A meno di chiavi estremamente lunghe, prima o poi si arriva a forzarle
- È opportuno usare chiavi molto lunghe
- Si deve supporre che le chiavi vecchie siano di fatto forzate
- È necessario “rigenerare” i timestamp

Compromissione della chiave di cifratura

- Permette l’accesso ai dati cifrati
- Non invalida la firma se fatta con un’altra chiave

Compromissione della chiave di firma

- Permette di creare firme false
- La revoca della chiave (quando scoperta) non lo impedisce
- Il timestamp permette di mantenere la validità dei documenti già firmati.

Compromissione della chiave della CA

- Permette di generare certificati falsi
- La revoca comporta la revoca dei certificati emessi
- Anche in questo caso il timestamp permette di limitare i danni
- La “compromissione” può anche essere un uso illegittimo, possibile anche se la chiave è su smart card

Compromissione della chiave di timestamp

- Permette di creare documenti predatati
- Quindi anche datati a quando erano usate chiavi ora falsificabili...
- È un servizio semplice ma estremamente critico
- Spesso l'autorità id timestamp è la CA stessa, almeno dal punto di vista organizzativo/gestionale
  - Se una è compromessa, è più facile che sia compromessa anche l'altra...

Inaccessibilità del repository

- Le chiavi sono verificate con le informazioni già ottenute (o altrimenti si blocca l'attività)
- Impossibilità di scaricare le CRL, ordinarie o urgenti, o di verificare via OCSP

Mercato, scelte e trust

- I certificati si sono diffusi con SSL, e il web è ancora l'uso principale
- Problema: chi sceglie la CA è il server, chi si deve fidare è il client
  - Quale CA scegliere?
- Alcune CA hanno accordi commerciali con i produttori di browser per essere “compresi” nella distribuzione

Problemi delle CA nei browser

- Il canale di distribuzione è sicuro?
- Chi verifica le politiche della CA e il loro rispetto?
  - Non il server, non il client, ma il produttore del browser
  - Nessuno è terzo quando ci sono accordi commerciali

Opzioni per il server

- Utilizzare una CA “già inclusa”
  - Nessun messaggio/errore al client
  - Tocca pagare
  - Le politiche possono non piacere (es. responsabilità)
- Usarne una non compresa
  - Far accettare la CA al client (possibile solo con un rapporto diretto)
  - Errori e warning in quantità
- Usare un certificato autofirmato
  - Come sopra, ma più warning
- Cosa conviene? ☺

Opzioni per client

- Nessuna: o accetta la CA scelta dal server, o non usa il servizio...
  - Ma non era il client a doversi fidare?
- Di fatto, la quasi totalità degli utenti Internet non ha nessuna idea di tutto questo, della presenza di certificati sul browser, né di “fidarsi” di qualcuno
  - Sa solo che alcuni siti sono “sicuri”
  - ... e che quando c'è un warning o è una frode o si ignora, più o meno a caso

Autenticazione del client

- È generalmente considerata impraticabile, perché si devono installare certificati sul client
  - Ed eventualmente su più browser...
- Qualche banca ci ha provato molto presto ed ha rinunciato
- Qualche altra li usa tuttora, almeno per i clienti business