

applicazioni della crittografia

sommario

- certificati e public key infrastructure
- protocolli di trasporto ssl, tls e ssh
- protocolli di autenticazione di livello 2
- virtual private networks
- altre applicazioni
 - one time passwords
 - documenti crittografati
 - posta elettronica
 - filesystem

certificati e public key infrastructure

certificato

- un **certificato** è un messaggio o documento firmato che stabilisce che una **chiave pubblica** è relativa ad un certo “**nome**”
- il nome deve poter essere ricondotto ad un **soggetto**
- esempio
 - $[\text{name=pippo, key=823764}]_{\text{pluto}}$

certification authority (CA)

- una CA è un'entità che emette certificati firmati con la sua chiave privata
 - l'emissione avviene su richiesta del subject
 - esempio: [name=pizzonia,key=034985]_{CA_ROMA3}
- una CA deve verificare che...
 - il subject sia identificabile chiaramente col nome dichiarato
 - deve chiedere delle prove al subject (es. carta di identità)
 - dovrebbe verificare che non ci siano nomi simili con cui un utente possa fare confusione (es. www.microsoft.com)
 - il subject sia in possesso della corrispondente chiave privata
 - una CA è tanto più affidabile quanto più le procedure per le verifiche sono stringenti
 - una CA non è un server, è un ufficio che si avvale di strumenti tecnologici

terminologia

- **issuer** di un certificato: chi emette il certificato (tipicamente una CA)
- **subject** di un certificato: il soggetto a cui fa riferimento il nome contenuto nel certificato
- **verifier** o **relying party**: chi cerca una prova che una chiave pubblica è associata ad un certo nome
 - es. un browser

certificazione per delega, catene di certificati

- CA1 può delegare CA2 nell'emissione dei certificati (di un certo tipo)
 - si presume che CA1 verifichi che CA2 esegua tutti i controlli che CA1 farebbe direttamente
 - la delega può essere a più livelli (CA2 può a sua volta delegare)
- un verifier si fida di CA2 poiché si fida di CA1 e di come controlla che CA2 svolga bene il proprio lavoro
- la delega è espressa mediante una catena di certificati
 - $[\text{name=pippo, key=234212}]_{\text{CA2}}$
 - $[\text{name=CA2, key=309485}]_{\text{CA1}}$
 - $[\text{name=CA1, key=208372}]_{\text{CA1}}$ (self signed)
 - un **certificato self signed** è solo un modo di memorizzare una chiave pubblica associata ad un nome, non vi è nessuna semantica di fiducia associata
 - comodo per omogeneità con la memorizzazione delle altre chiavi pubbliche

tipi di certificati

- che differenza c'è tra questi due certificati?
 - $[\text{name=pippo, key=234212}]_{\text{CA2}}$
 - $[\text{name=CA2, key=309485}]_{\text{CA1}}$
- questo schema permetterebbe a pippo di firmare certificati!
- il certificato deve contenere indicazione sulla possibilità di firmare certificati
 - $[\text{name=pippo, key=234212, usage=signData}]_{\text{CA2}}$
 - $[\text{name=CA2, key=309485, usage=signCert}]_{\text{CA1}}$

terminologia

- **trust anchor:** un issuer di cui ci fidiamo senza bisogno di avere un certificato per questo
 - es. i browser hanno un insieme di CA di cui si fidano già preconfigurato (certificati self-signed)
 - in realtà è il produttore del browser che si fida delle CA
 - gli utenti si fidano delle scelte fatte dal produttore del browser e del fatto che nessuno abbia modificato le configurazioni
 - infatti nessuno controlla che non ci siano trust anchor spurie configurate nel browser
- **target:** un nome per cui vogliamo trovare una catena di certificati che parta da un trust anchor
 - es. il browser che naviga in un sito “sicuro” ha come target il “nome del sito”

modelli di fiducia

- monarchia
 - una sola CA
- monarchia + registration authority (RA)
 - una CA e più RA a cui la CA delega le verifiche, i certificati sono comunque firmati da CA
- monarchia + deleghe
 - una trust anchor che delega la firma di certificati e crea catene di certificati

modelli di fiducia

- oligarchia (+ eventuali deleghe)
 - varie trust anchor
 - è il modello usato dai browser
- anarchia
 - l'utente sceglie le trust anchor
 - chiunque può emettere certificati
 - è il modello usato da PGP
- vincoli sul nome
 - la delega sull'emissione dei certificati può essere vincolata in modo da permettere firme solo di certificati relativi a certi domini
 - es. la CA PerfectSign delega CA_UNIROMA3 per emettere certificati solo per nomi che terminano con uniroma3.it
 - [name=CA_UNIROMA3, key=9345, usage=signCert, **constraint=*.uniroma3.it**]_{PerfectSign}

semantica di un certificato e pubblicazione di chiavi private

- un certificato asserisce che tutto ciò che è firmato con la corrispondente chiave privata è firmato dal subject del certificato
- che succede se la chiave privata viene pubblicata?
 - per errore umano
 - per attacco ai sistemi
 - per attacco crittoanalitico
- chi ha la chiave privata riesce a impersonare il subject con il minimo sforzo

validità di un certificato

- la probabilità che una chiave privata sia pubblicata aumenta con il tempo
 - es. è più facile che il sistema di `www.securebank.com` venga penetrato nell'arco di un anno anziché di una settimana
 - in un certo senso la chiave privata “si deteriora”
- diamo un tempo di vita alle chiavi private e quindi una scadenza ai certificati
 - tipicamente i certificati hanno anche un tempo di inizio di validità
 - es. `[name=CA_UNIROMA3, key=9345, usage=signCert, constraint=*.uniroma3.it, validity=(from 2007.01.01 to 2007.12.31)]`_{PerfectSign}
- il verifier ha l'onere di **verificare** che i certificati della sua catena siano validi

revoca di un certificato

- se la chiave privata viene pubblicata durante il tempo di validità del certificato il certificato deve essere **revocato**
- il verifier ha l'onere di **verificare** che i certificati della sua catena siano non revocati
- il subject ha l'onere, in caso di pubblicazione della chiave privata, di **richiedere immediatamente la revoca** all'issuer

certificate revocation list (CRL)

- la CA emette liste firmata di certificati validi ma revocati
 - i certificati hanno un ID
 - le CRL contengono gli ID dei certificati validi ma revocati
- l'emissione è strettamente periodica e dotata di timestamp
 - bisogna ottenere l'ultima CRL prima di fidarsi di un certificato
 - nessun ci deve poter far credere che una CRL vecchia è la più recente (timestamp)
 - attenzione al real time clock del sistema
- la distribuzione viene fatta tipicamente tramite ftp o http (rfc 2585)
 - il certificato può indicare un url da cui scaricare la CRL

on-line revocation server

- per applicazioni con stringenti requisiti di tempo è possibile avere un on-line revocation server per ottenere lo stato di un certificato in tempo reale
- Online Certificate Status Protocol (OCSP)
 - rfc 2560

X.509 e PKIX

- il formato standard dei certificati è stabilito da X.509 (ITU, l'ultima versione è la 3, usata comunemente) e dal profilo PKIX (rfc 4212)
- specifiche date mediante ASN.1
 - standard (ITU/ISO) per descrivere protocolli a livello astratto
- codifica: tipicamente .der (binaria)
 - standard per codificare protocolli descritti con ASN.1
- variante: .pem (ascii)
 - è una codifica ascii del corrispondente .der
 - simile a uuencode
- per ulteriori info vedi
 - <http://en.wikipedia.org/wiki/ASN.1>

name e common name

- struttura dei nomi X.500 (ITU)
 - C=US (country)
 - ST=Rohde Island (state)
 - L=Providence (location, città)
 - O=SecureBank Inc. (organization)
 - OU=Loan Dept. (organization unit)
 - CN= John Taylor (common name del subject)
- per i server web, per convenzione, CN deve essere il DNS name o il numero IP
 - molti browser verificano questo ed eventualmente avvertono l'utente

X.509v3

- esempio con alcuni campi comuni, molti altri sono possibili

Version: 3 (0x2)

Serial Number: 8a:5f:f6:85:21:f9:20:41

Signature Algorithm: md5WithRSAEncryption

Issuer: C=IT, ST=Italy, L=Rome, O=ROMA3,
OU=Network Research,
CN=PerfectSign Inc./emailAddress=....

Validity

Not Before: Jun 16 07:34:45 2006 GMT

Not After : Jun 13 07:34:45 2016 GMT

Subject:

C=US, NY=Italy, L=Rome, O=SecureBank Inc.,
OU=Loan,
CN=www.securbank.it/emailAddress=none

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public Key: (1024 bit)

Modulus (1024 bit):

00:d3:b2:17:09:54:ee:50:e1:28:cc:70:e3:f3:d0:

....

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Basic Constraints: CA:TRUE

PKCS #1 ... #15

- Public-Key Cryptography Standard
 - pubblicate da RSA
- fissano codifiche precise per varie tipologie di “dati crittografici” per RSA
- alcuni esempi
 - PKCS#1: RSA firma e cifratura: aspetti matematici e algoritmici
 - PKCS#5: Password-Based Cryptography Standard
 - PKCS#7: RSA firma e cifratura, formati messaggi
 - PKCS#8: Codifica delle chiavi private, formato
 - PKCS#10: Certificate Signing Requests per una CA
 - PKCS#12: formato coppia <chiave privata, certificato>

punti critici nelle PKI: il verifier

- il comportamento del verifier (cioè es. web browser) è quasi mai ortodosso
 - verifica il common name?
 - quasi tutti lo fanno
 - può ignorare gli intervalli di validità
 - configurabile? qual'è il default?
 - può ignorare il problema della revoca
 - configurabile? qual'è il default?
 - chi seleziona le trusted anchor?
- conosci il comportamento del tuo browser preferito?

punti critici nelle PKI: l'issuer

- il comportamento dell'issuer è conforme alle esigenze di sicurezza?
 - quando un utente fa affidamento su un certificato non conosce le politiche di sicurezza della CA
 - che controllo ha una CA sulle CA delegate o sulle RA?
- prova a cercare in Internet le politiche di una CA
- Normativa europea: EU Directive 1999/93/EC

punti critici nelle PKI: l'utente

- l'utente non sa che cosa è ...
 - un “sito sicuro”
 - un certificato
 - un common name
 - una CA
 - una trust anchor

punti critici nelle PKI: interazione utente-browser

- l'utente tende a rispondere OK
 - Il certificato è firmato da una CA sconosciuta. Lo vuoi accettare lo stesso?
 - accetto questo certificato per sempre?
 - il common name non corrisponde al dns name, accetto questo certificato?
- l'utente non verifica il nome del subject
 - si limita ad osservare il lucchetto chiuso
 - raramente ci clicca sopra per vedere chi è il subject del certificato

punti critici nelle PKI

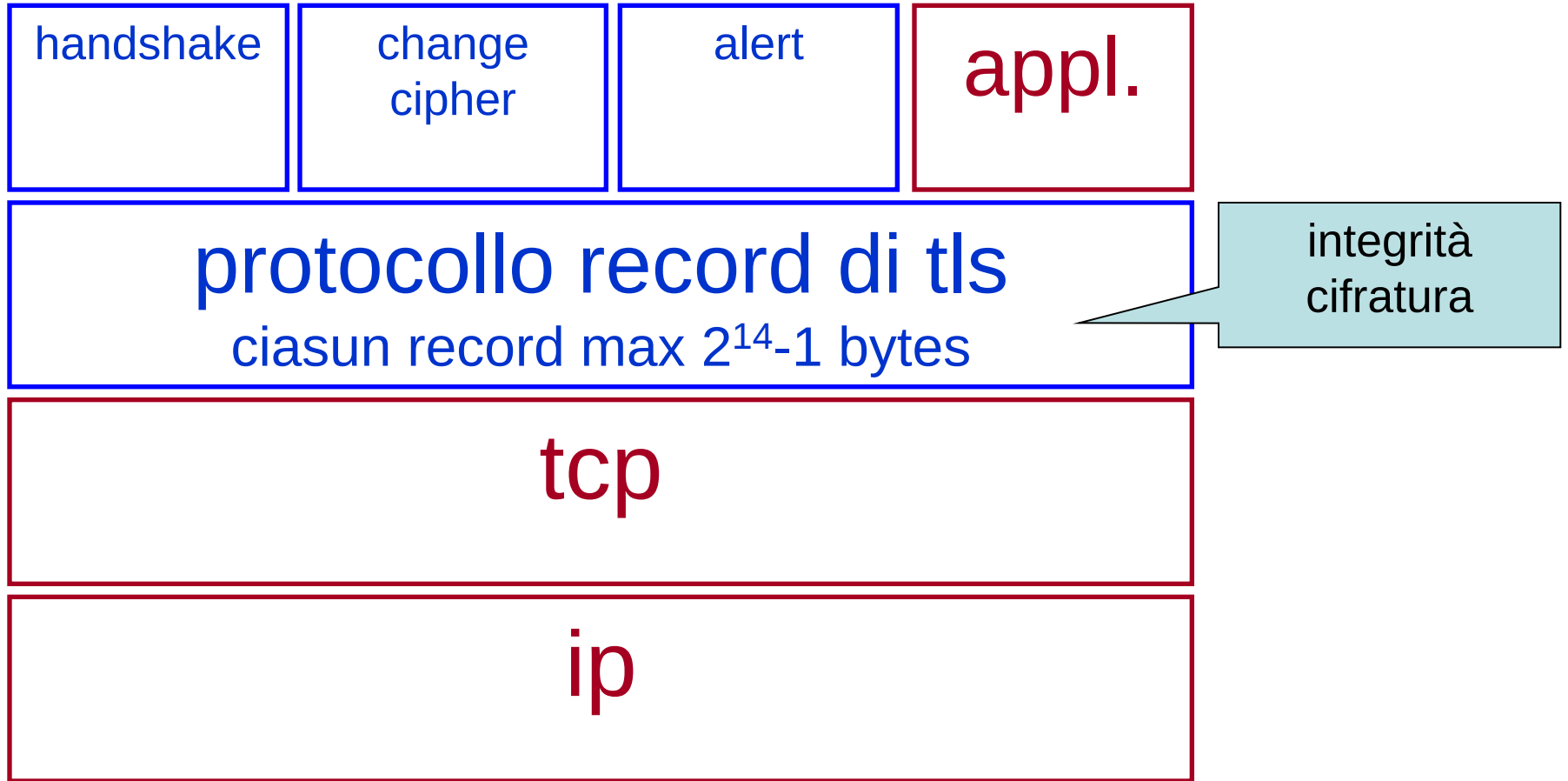
- Ellison, Schneier. “Ten Risks of PKI”.
Computer Security Journal. Nov 1, 2000

protocolli di trasporto ssl, tls e ssh

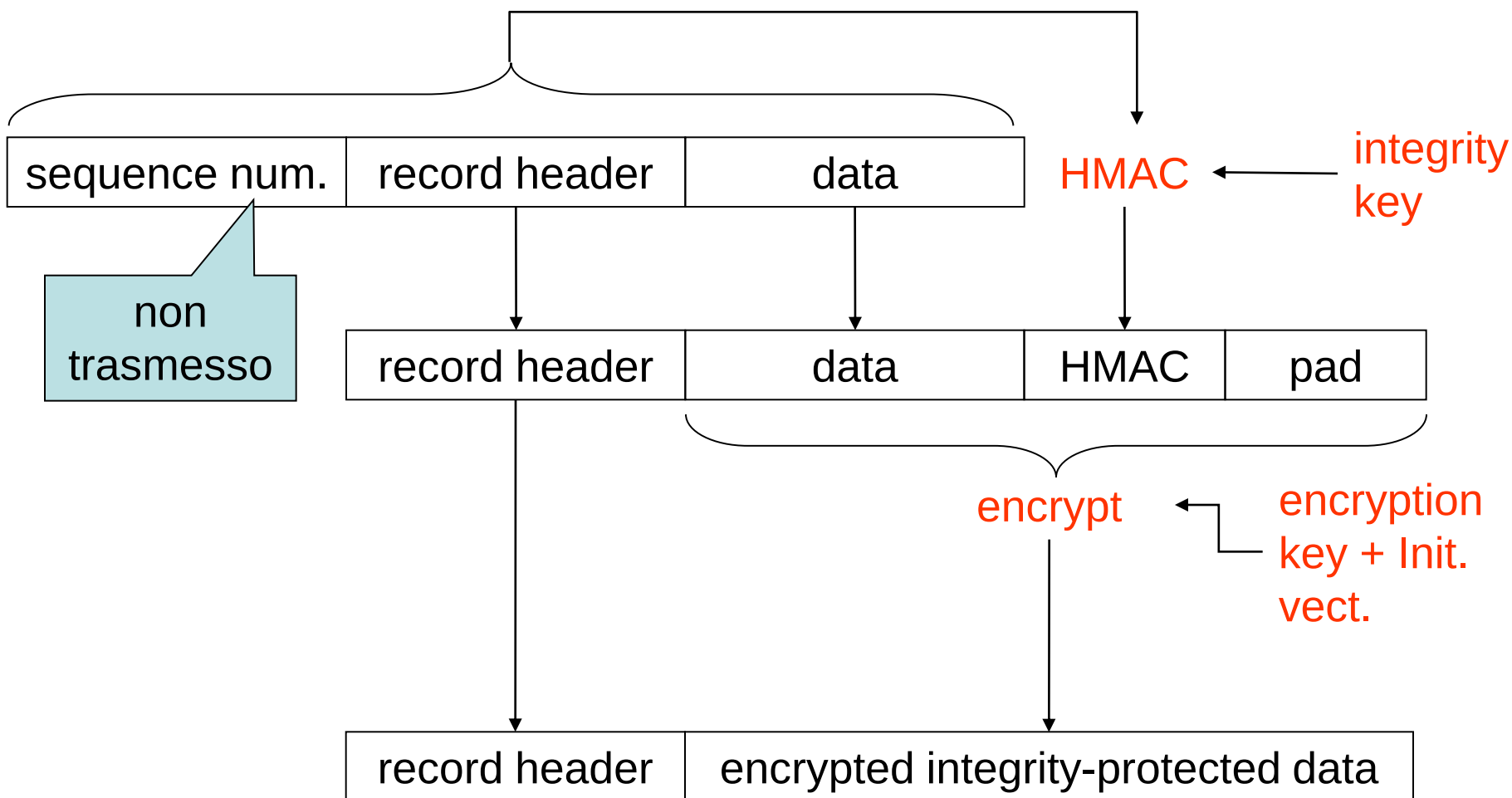
descrizione generale

- Secure Socket Layer (Netscape)
 - versione 2, obsoleta, qualche vulnerabilità
 - versione 3
- Transport Layer Security (IETF, rfc2246)
 - versione 1, molto simile a SSLv3 ma incompatibile
- protocolli del tutto generali
 - usati spesso per http (https su porta 443)
 - usati anche per imap, pop, telnet
- supportati dalle applicazioni più diffuse
- sono protocolli piuttosto complessi

il rapporto con la pila osi



cifratura dei record



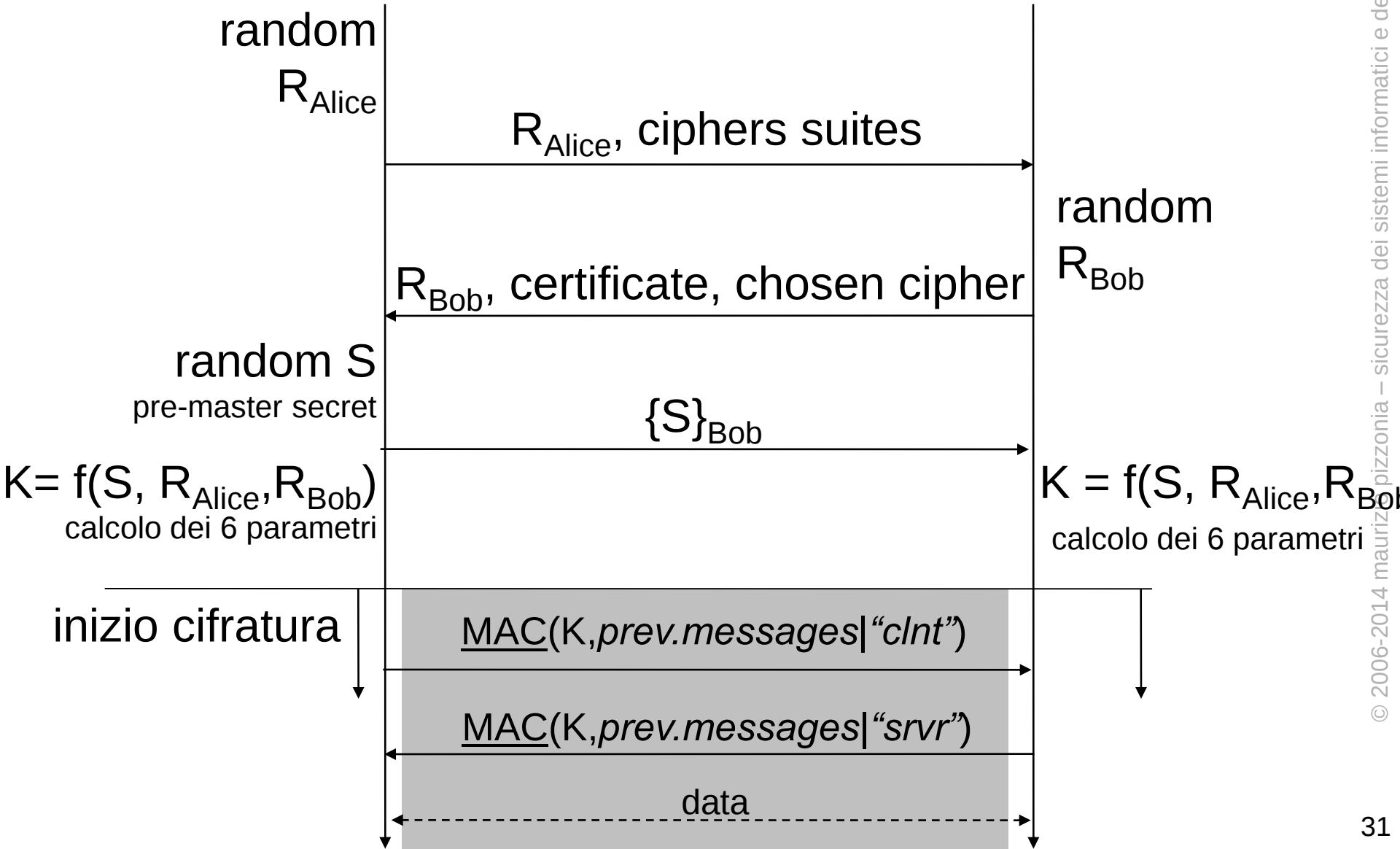
stato della connessione

- per iniziare una connessione criptata i due devono accordarsi su...
 - algoritmo di cifratura
 - hash function per HMAC
 - come scambiare “la chiave” (pre-master secret)
- ... e su i seguenti 3 segreti per ciascuna direzione (totale 6)
 - integrity protection key
 - encryption key
 - Initialization Vector (necessario per molti algoritmi di criptazione a blocchi, es. DES)
 - sono tutti calcolati a partire dal pre-master secret

scambio rsa

Alice

Bob



esercizio

- perché SSLv3/TLS inseriscono un controllo di integrità per l'handshake?

varianti

- il client può fornire un proprio certificato per essere autenticato
- diffie-hellman
 - il server interviene nella creazione di S
 - autenticato con RSA o DSS
- diffie-hellman ephemeral
 - le chiavi vengono generate per la sessione e poi dimenticate
 - forward secrecy
- session resumption

cipher suites di TLS

- una cipher suite è un insieme di algoritmi da usare per la cifratura, l'integrità, e lo scambio di chiavi
- esempio di stringa identificativa di una cipher suite

TLS_RSA_WITH_3DES_EDE_CBC_SHA

chiavi
scambiate
con RSA

cifratura:
3DES nella
variante EDE
CBC

integrità:
HMAC con
SHA-1

default, usato
solo per
handshake

cipher suites di TLS

forward
secrecy

i più usati

TLS_NULL_WITH_NULL_NULL

TLS_RSA_WITH_NULL_MD5

TLS_RSA_WITH_NULL_SHA

TLS_RSA_EXPORT_WITH_RC4_40_MD5

TLS_RSA_WITH_RC4_128_MD5

TLS_RSA_WITH_RC4_128_SHA

TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5

TLS_RSA_WITH_IDEA_CBC_SHA

TLS_RSA_EXPORT_WITH_DES40_CBC_SHA

TLS_RSA_WITH_DES_CBC_SHA

TLS_RSA_WITH_3DES_EDE_CBC_SHA

TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA

TLS_DH_DSS_WITH_DES_CBC_SHA

TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA

TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA

TLS_DH_RSA_WITH_DES_CBC_SHA

TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA

TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA

TLS_DHE_DSS_WITH_DES_CBC_SHA

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA

TLS_DHE_RSA_WITH_DES_CBC_SHA

TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA

TLS_DH_anon_EXPORT_WITH_RC4_40_MD5

TLS_DH_anon_WITH_RC4_128_MD5

TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA

TLS_DH_anon_WITH_DES_CBC_SHA

TLS_DH_anon_WITH_3DES_EDE_CBC_SHA

deprecati perché DH
non autenticato è
vulnerabile a MitM

ssh

- ssh è un concorrente di ssl/tls
 - v1 (vulnerabile), v2 attualmente in uso
- del tutto generale
 - usato soprattutto come telnet criptato
 - si può fare tunneling di qualsiasi cosa in ssh (opzioni –L e –R)
 - ma ora si può fare anche con ssl (vedi “stunnel”)
- companion protocols/commands
 - scp, sftp
- diffusione
 - famoso (implementazione open: openssh)
 - ampiamente supportato
 - recente standardizzazione
 - rfc 4250-4256 e seguenti
 - Supporta autenticazione RSA ma **non supporta i certificati**

virtual private networks

VPN

- Internet e le reti degli ISP costano poco ma sono insicure
- le VPN sono reti private “sicure” ricavate da infrastrutture pubbliche
 - gli ISP offrono VPN con dei Service Level Agreement per garantire una certa QoS
- usate per
 - accesso da Internet alla “rete aziendale”
 - Intranet geograficamente distribuite
 - cioè collegamento di sedi distanti della stessa azienda

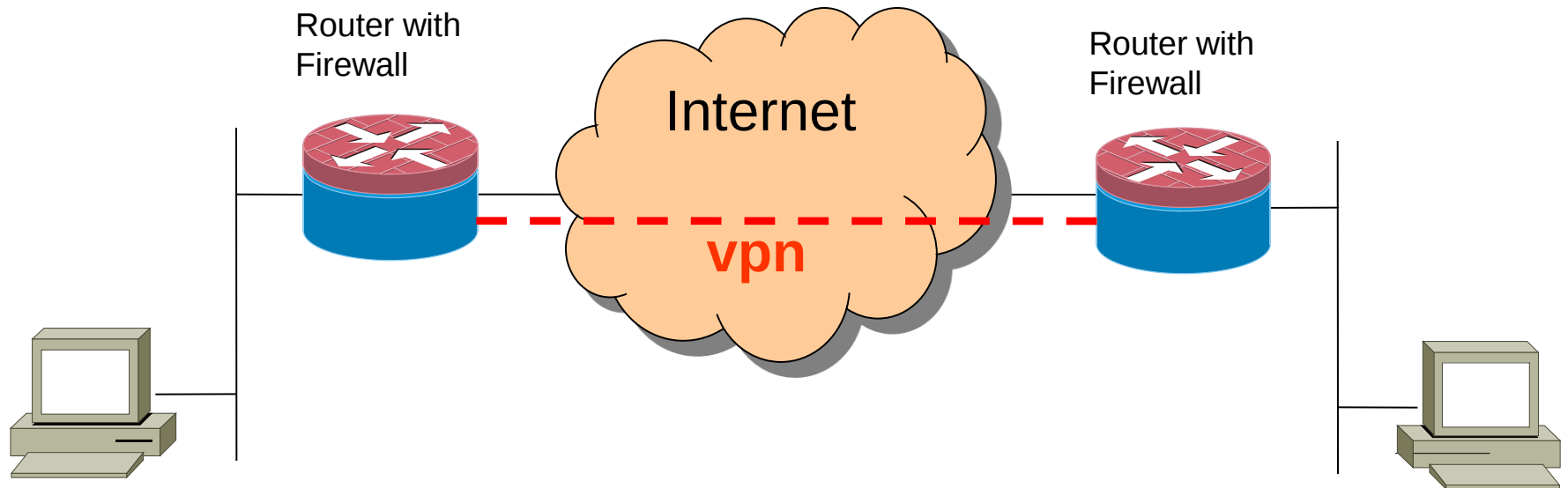
strumenti

- ssl/tls (OpenVpn),
- ssh (opzioni -L -R -W -D ecc.)
- IPsec
- altro (PPTP, L2TP/IPsec)

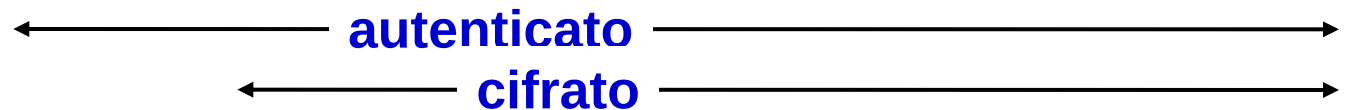
ipsec

- due protocolli crittografici per IP
 - Encapsulating Security Payload (ESP , rfc 4303)
 - confidenzialità (opzionale), e integrità e dei dati
 - Authentication Header (AH, rfc 4302)
 - integrità dei dati e di parte dell'header IP
 - raramente usato, non c'è motivo di autenticare l'header IP
- due modalità
 - tunnel mode
 - dati in ip(originale) in ipsec in ip(nuovo)
 - transport mode
 - dati in ipsec in ip
 - non strettamente necessario
 - più efficiente perché ha un header in meno (mtu maggiore)

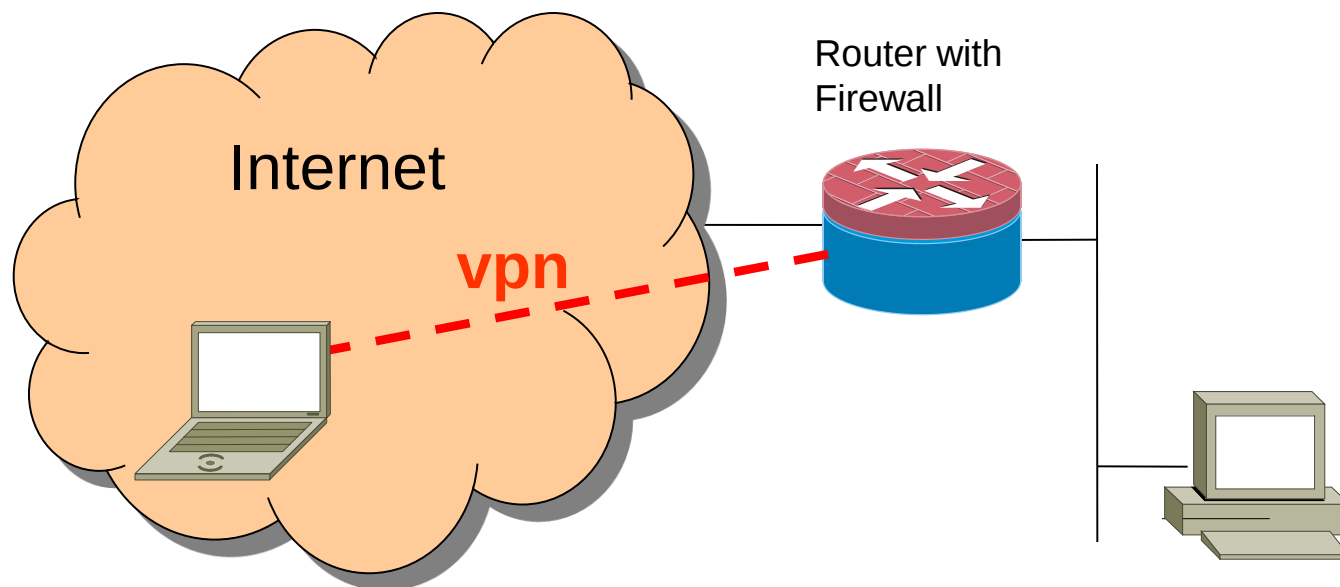
ipsec tunnel mode



**pacchetto
originale**



ipsec transport mode



pacchetto originale



← autentificato →

← cifrato →

concetti ipsec

- security association (SA)
 - tra due macchine (addr, addr, modo, algoritmi, chiavi, SPI)
 - spi: security parameter index
 - identifica la SA, non bastano gli indirizzi poiché più security association possono essere instaurate tra le stesse macchine
 - l'spi viene inviato negli header ipsec
- security policy
 - quali pacchetti sono ammessi per essere instradati nel tunnel

chiavi di sessione

- le chiavi di sessione possono essere configurate manualmente o automaticamente
- Internet Key Exchange (IKE)
 - autenticazione
 - supporta sia chiavi pubbliche che shared secret
 - security association
 - negoziazione degli algoritmi di crittazione e di verifica di integrità
 - scambio chiavi di sessione
 - key rollover
 - protocollo molto complesso (forse troppo)
 - v1 (rfc 2407-2409, qualche problema di sicurezza)
 - v2 proposto recentemente (rfc 4306, 4307)

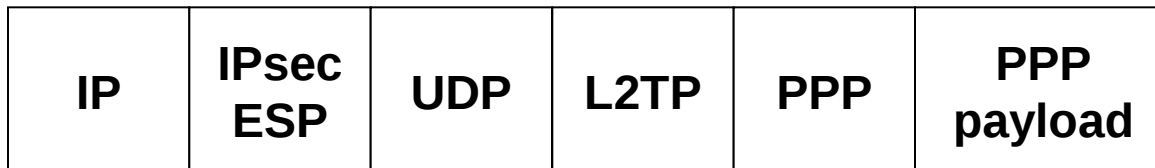
pptp

- Microsoft
- Generic Routing Encapsulation (GRE, rfc 2784)
 - un protocollo per fare tunnel generici in IP
- protocollo di management del tunnel
 - tcp port 1723
 - problematico per i firewall
- ppp in gre in ip
- autenticazioni MSCHAP (basta una password) o EAP-TLS
 - challenge/response in chiaro
- crittografia opzionale
 - Microsoft Point-to-Point Encryption (MPPE)
 - RC4 chiavi 40-128 bit

IP	GRE	PPP	PPP payload
-----------	------------	------------	------------------------

L2TP/IPsec

- Microsoft
- Layer2 Transport Protocol (L2TP)
 - derivato da ppp incapsulato in udp
 - non prevede autenticazione
- IPsec ESP transport mode
- più sicuro di pptp
 - autenticazione strong tra macchine (IPsec/IKE)
 - autenticazione di utente (su ppp ma criptata)
- poco pratico
 - richiede setup di ipsec (shared secret o certificato)
- poco efficiente
 - mtu ridotto



livello 2

autenticazione a livello 2 point to point

- gli estremi di una connessione ppp sono tipicamente autenticati
 - vedi connessioni dial-up e adsl
- protocolli famosi:
 - Password Authentication Protocol (PAP)
 - richieste di autenticazione con password in chiaro!
 - Challenge-Response Handshake Protocol (CHAP)
 - il server invia un challenge, il client risponde con un MAC del challenge (shared secret)
 - richiesta ripetuta durante la sessione (anti hijacking)
 - MS-CHAP— versione Microsoft di CHAP
 - lo shared secret è derivato dalla password
 - Extensible Authentication Protocol (EAP)

EAP

- RFC 3748
- framework per la negoziazione di meccanismi di autenticazioni arbitrari
- prevede una negoziazione del metodo di autenticazione
 - metodi diversi prevedono protocolli di autenticazione diversi (e quindi una sequenza di messaggi diversa)
- eap methods (sono oltre 40)
 - es. eap-md5: autenticazione one-way,
 - es. eap-tls: usa tecniche simili a tls
- è possibile il supporto per token card, dispositivi biometrici, OneTimePasswords, Smart Card, certificati digitali ...

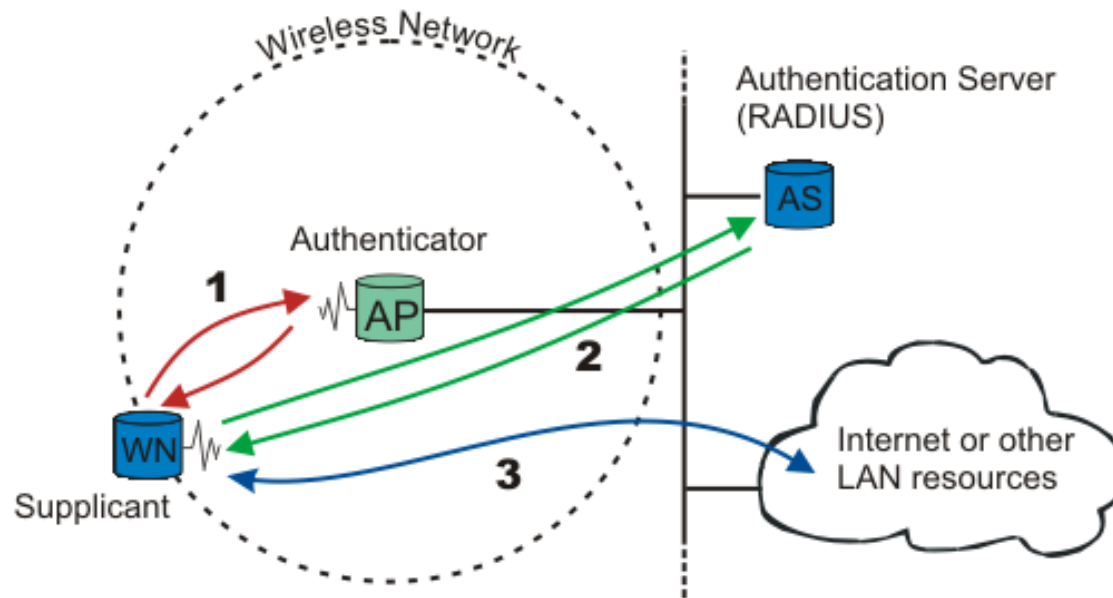
autenticazione a livello 2 per LAN

- ieee 802.1X
- è un modo di incapsulare EAP in frame su LAN
 - detto anche EAPoL (EAP over LAN)
- il server è tipicamente un apparato di rete
 - switch
 - access point
 - ...
- scomodo avere uno user db in un apparato di rete...

radius

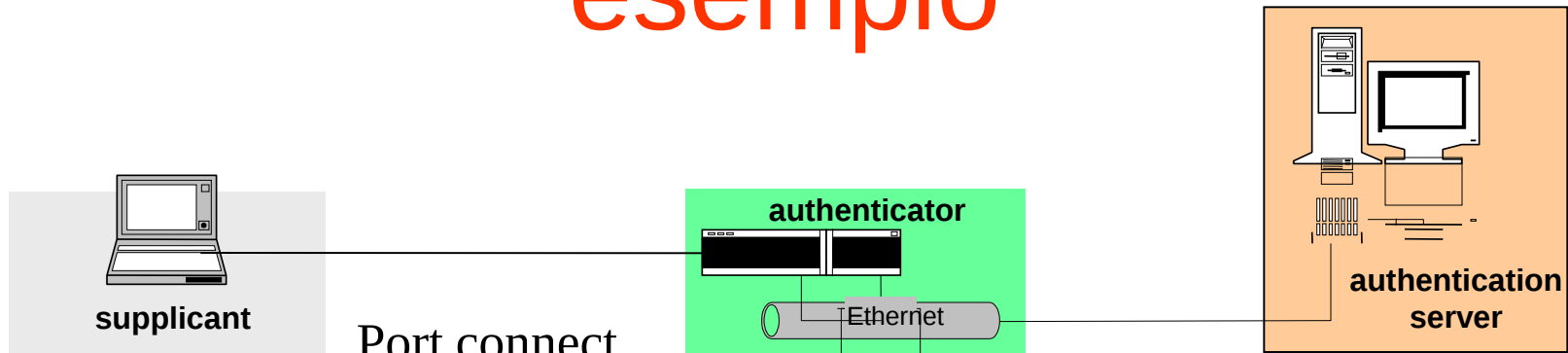
- rfc 2865, rfc 2866
- su udp
- supporto per EAP (rfc 3579)
- elementi
 - User (pc, laptop, telefono, ecc)
 - Radius server
 - autorizza o meno l'accesso alla rete
 - User database
 - ldap, dbms, ecc
 - Network Access Server (NAS, client del radius server)
- protocolli analoghi e concorrenti
 - diameter (rfc 3588, recente)
 - tacacs, tacacs+ (più vecchi)

802.1X e radius

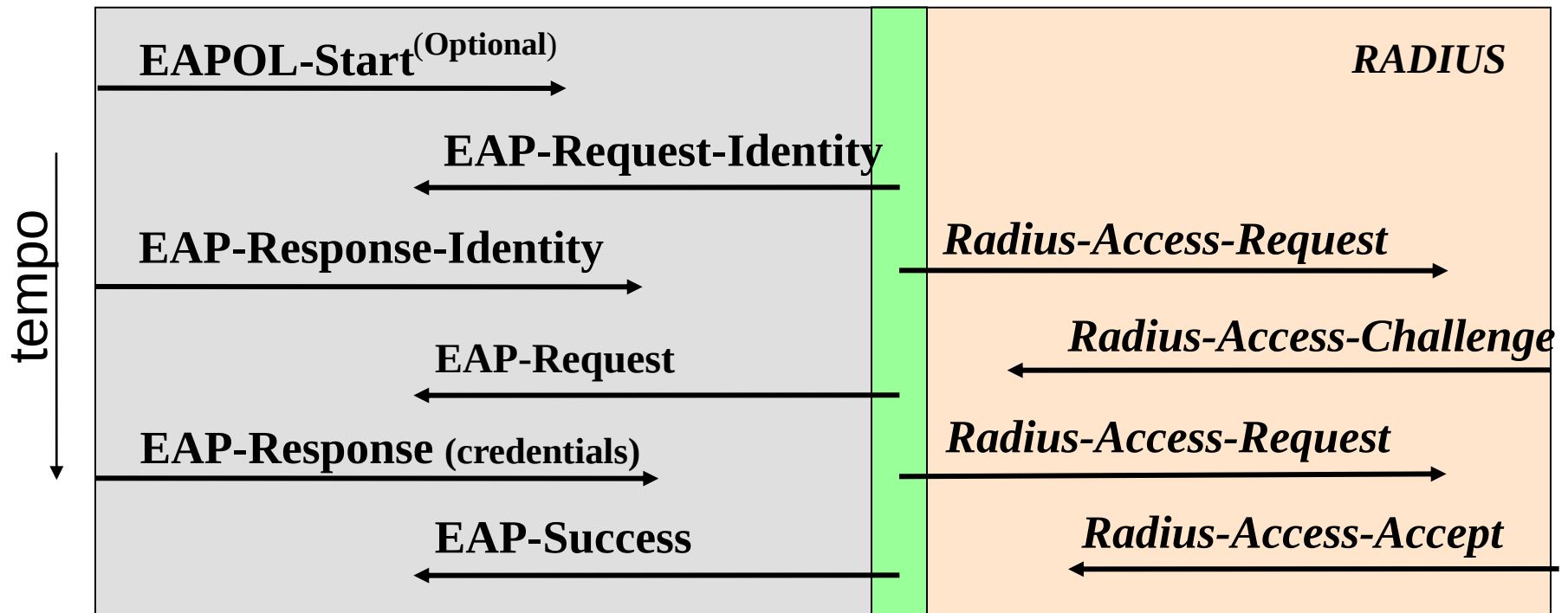


- supplicant (user) usa EAPoL con l'autenticator (in questo caso un access point che fa da NAS)
- l'autenticator passa i messaggi EAPoL al authentication server usando RADIUS e il supporto per EAP
- quando l'autenticator riceve la conferma dall'authentication server che il supplicant è autenticato allora permette al traffico del supplicant di raggiungere la rete

esempio



Accesso bloccato solo traffico EAPoL



Accesso concesso (con eventuali vincoli) dall'autentication server

wireless

- wep (obsoleto e vulnerabile)
 - rc4 (40bit key, 24bit iv), integrity con crc-32
- wpa
 - ha bisogno IEEE 802.1X server
 - distribuisce pre-shared secret diversi a ciascun utente, mutual autenticazione
 - rc4 (128bit key, 40bit iv)
 - key rollover
 - integrity con mac e frame counter (no replay attack)
- 802.11i (wpa2)
 - evoluzione di wpa
 - tra le altre cose usa AES

altre applicazioni

posta elettronica

- pretty good privacy (PGP)
 - obsoleto
- Privacy Enhanced Mail (PEM)
 - IETF, obsoleto
 - usato come formato file (.pem)
- S/MIME (rfc 3850-3851)
 - creato da RSA
 - mime (rfc 2045-2049) + pkcs#7
- Posta Elettronica Certificata
 - in italia ha lo stesso valore legale di una raccomandata con ricevuta di ritorno

documenti crittografati

- criptati con una chiave simmetrica S
- S è cifrata con la chiave pubblica di ciascun soggetto autorizzato alla lettura
 - S/MIME
 - più destinatari ciascuno con la sua chiave pubblica
 - EFS (encrypted filesystem windows XP)
 - chiave privata e pubblica associata all'utenza
 - chiave privata è persa quando l'utenza viene cancellata
 - più soggetti possono essere autorizzati alla lettura di un file (agente di recupero)

confidenzialità dei files

- criptazione a livello di
 - file
 - directory
 - filesystem
 - disco
- windows: encrypted filesystem (EFS)
 - più utenti possono aprire un file criptato
 - encryption/decryption trasparente all'utente durante l'uso del file
 - disponibili prodotti commerciali
- Window Vista: BitLocker
 - a livello di disco
 - basato su TPM
- linux
 - encfs: directory level
 - cryptofs: directory level
 - rasier4: filesystem level
 - ecc.

one time passwords (OTP)

- poiché la password può essere rivelata facciamo in modo che si possa usare una sola volta
- generazione di un insieme di passwords
 - Lamport: $h(p)$, $h(h(p))$, $h(h(h(p)))$,
 - le passwords vengono chieste a partire dall'ultima
 - sniffare un telnet o vedere un login non aiuta a entrare nel sistema
 - l'utente deve tenere privato l'insieme di passwords!
 - per ciascun utente si memorizza l'hash dell'ultima password
- time-synchronized OTP
 - dispositivo hardware con clock sincronizzato con il server
 - genera password che dipendono dal tempo