applicazioni della crittografia certificati e public key infrastructures (PKI)

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
  - $[name=pippo,key=823764]_{pluto}$

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.mircosoft.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
  - $[\text{name=pippo, key=234212}]_{CA2}$
  - $[\text{name=CA2, key=309485}]_{CA1}$
  - $[\text{name=CA1, key=208372}]_{CA1}$ (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?
  - $[\text{name=pippo, key=234212}]_{CA2}$
  - $[\text{name=CA2, key=309485}]_{CA1}$
- questo schema permetterebbe a pippo di firmare certificati!
- il certificato deve contenere indicazione sulla possibilità di firmare certificati
  - $[\text{name=pippo, key=234212, usage=signData}]_{CA2}$
  - $[\text{name=CA2, key=309485, usage=signCert}]_{CA1}$

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, con certificati self-signed (*root certificates*)
    - 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 autority (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]$_{\text{PerfectSign}}$

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 criptoanalitico
- 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)]$_{\text{PerfectSign}}$
- 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 dever 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 Iland (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 (CSR) per una CA
  - PKCS#12: formato usato per configurare i web server, contiene chiave privata e catena di certificati

requesting a certificate: workflow

1. the subject creates a CSR (PKCS #10)
    1. it creates a public/private key pair (PKCS #1,7,8)
    2. it creates an unsigned certificate for the public key (X509v3)
    3. it signs the CSR by the private key (self signed)
2. the issuer (i.e., the CA)
    1. …checks CSR signature by public key contained in the CSR itself
    2. …performs a number of checks on the content of the certificate (e.g., if it conform to subject identity and to what subject has payed like validity, domain etc.)
    3. …signs the unsigned certificate of the CSR with the private key of the CA
    4. … sends it to the subject.

openssl

- openssl is a tool (a *de-facto* standard) for certificate creation and management
- e.g., it allows a user to…
  - …create a private/public key pair
    - RSA, DSA, and many other schemes
  - …create an unsigned certificate from public key
  - …create a certificate signing request (CSR) to be sent to a CA
  - …create pkcs12 files to be used used with TLS-enabled web servers
- e.g., it allows a CA to…
  - …manage its private keys and root certificates
  - …process CSRs and issue certificates
  - …tracking issued certificates
  - …manage revocation

- openssl support many other cryptographic-related operations
  - cryptographic hashes, encryption and eecryption, SSL/TLS client and server tests, S/MIME mail, timestamping requests

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 (verifier) fa affidamento su un certificato non conosce le procedure di verifica della CA (issuer)
  - che controlli fa una CA sulle CA delegate o sulle RA?
  - conflitto di interesse
    - le procedure di verifica prima delle emissione dei certificati devono tutelare il verifier, ma procedure complesse hanno costi elevati
    - una CA ha interesse a vendere un certificato ad un subject, il subject tende a preferire certificati a basso costo e semplici da ottenere poiché il verifier raramente analizza le procedure di verifica per l'emissione dei certificati
- prova a cercare in Internet le politiche di una CA
- regolamento europeo: eIDAS
  - electronic IDentification, Authentication and trust Services
  - in vigore in tutta l'UE dal 1 luglio 2016

ACME

- Automatic Certificate Management Environment
  - RFC 8555
- a protocol for automatic verification of domain possession
- adopted by very-low-cost or free certification authorities
  - no other checks are performed

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?
- attualmente i browser bloccano la navigazione
  - e l'opzione "continue at your own risk" è poco evidente

punti critici nelle PKI: interazione utente-browser

- 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

bibliografia

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