Quando vengono generate delle chiavi pubbliche, sorgono 2 dubbi:

1. Come vengono distribuite le chiavi pubbliche?
2. Chi ci assicura che una chiave pubblica è quella di un prefissato utente?

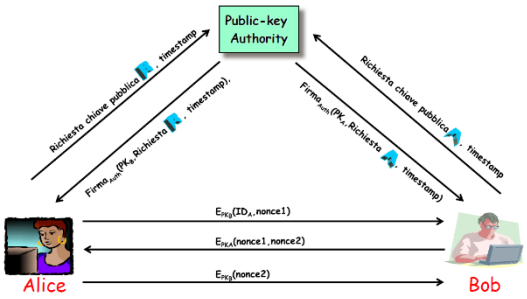
Ci sono alcune tecniche che rispondono a queste 2 domande: **annuncio pubblico**, **directory pubblica**, **autorità per le chiavi pubbliche** e **certificati per le chiavi pubbliche**.

L'**annuncio pubblico**, come descrive lo stesso nome, permette di mostrare a tutti la chiave pubblica ad esempio su un sito web personale, in allegato alle email. Il problema di questa tecnica è se possiamo fidarci di un annuncio del genere che vediamo sul web.

Si può creare una **directory di chiavi pubbliche** in cui ogni utente scrive e legge la propria chiave pubblica e prende nota delle chiavi pubbliche di altre persone. Ogni partecipante può registrare la chiave pubblica e aggiornarla se usata da troppo tempo o compromessa. Il problema di questa tecnica, cosi come per la tecnica precedente, è se possiamo fidarci di questa directory pubblica.

L'autorità per le chiavi pubbliche gestisce le directory di chiavi pubbliche, ogni utente chiede la chiave pubblica desiderata e l'autorità la invia. In questo modo viene risolto il problema di fiducia. Lo svantaggio di questa tecnica è che i server sono on-line e ci può essere collo di bottiglia.

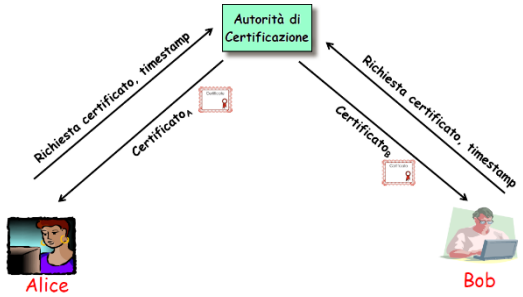
Un esempio di protocollo che si basa sull'autorità per le chiavi pubbliche è il seguente:



Se Alice e Bob vogliono dialogare tra di loro, devono sapere la chiave pubblica dell'altro. Poiché è l'autorità che gestisce le chiavi pubbliche, Alice chiede la chiave pubblica di Bob e Bob chiede la chiave pubblica di Alice. L'autorità invia la chiave pubblica ai 2 utenti mediante una firma: ***Firma$_{Auth}$(PK$_{BOB}$, Richiesta, timestamp)*** dove timestamp indica che all'ora in cui è stata fatta la richiesta, la chiave di Bob è la PK$_{BOB}$ che viene mandata in questo caso ad Alice.

Una volta ricevute le chiavi, può iniziare il dialogo tra Alice e Bob. Il dialogo è stabilito con un protocollo su chiave privata che garantisce che Alice sta parlando veramente con Bob e viceversa.

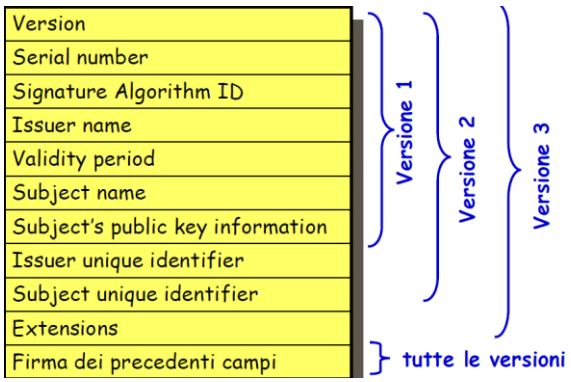
Per evitare di basarsi esclusivamente su un server on-line ci sono i **certificati**, uno strumento off-line che non interroga un server costantemente. Questo è l'approccio più utilizzato. Un certificato stabilisce che una chiave pubblica è di un utente. Nella vita reale un certificato è una carta di identità, nel mondo digitale è il certificato digitale che sono un insieme di informazioni che includono l'identità della persona, dell'organizzazione etc.. tutte queste informazioni sono firmate da un'autorità che tutti conoscono. Le proprietà dei certificati sono che ogni utente può leggerli e determinare nome e chiave pubblica, ognuno può verificarli ed assicurarsi dell'autenticità e solo l'autorità può crearli ed aggiornarli. Il formato più diffuso è quello definito nello standard internazionale ITU-T X.509 che ha le seguenti caratteristiche:



Chi vuole un certificato fa una richiesta e l'Autorità rilascia il certificato. Una volta che Alice e Bob hanno i certificati, per dialogare in futuro non dovranno ricontattare l'Autorità, ma potranno dialogare con il certificato rilasciato in passato. Il dialogo è molto semplice: Alice manda il suo certificato a Bob e Bob manda il suo ad Alice.

Che succede se la chiave privata viene compromessa? In questo caso l'utente può richiedere la revoca del certificato. Una revoca può avvenire per vari motivi, come compromissione della chiave privata, non più utile per lo scopo prefissato, perdita o malfunzionamento di security token, perdita di password o PIN. Se un certificato non è più valido, come si fa effettivamente a sapere che vale più? Ci sono varie possibilità, ma quella più utilizzata è di avere un file pubblico di chiavi revocate (Certificate Revocation List (CRL))

Mostriamo ora il funzionamento del formato del certificato più utilizzato al giorno d'oggi: il formato **X.509**



Si compone di diversi campi. Con il passare del tempo sono uscite ben 3 versioni e per ogni versione sono stati aggiunti nuovi campi.

Il primo campo è il **Version** che può essere 1,2,3 a seconda del numero di campi del certificato. C'è poi un **Serial number** che è un valore intero, unico per ogni CA e identifica senza ambiguità il certificato. **Signature Algorithm ID** è l'algoritmo che l'autorità usa per firmare tutti i campi del certificato. **Issuer name** indica il nome di chi ha firmato il certificato. Il **Validity period** indica, come suggerisce il nome, il periodo di validità del certificato. In **Extensions** si può motivare il perché si ha questa certificazione.

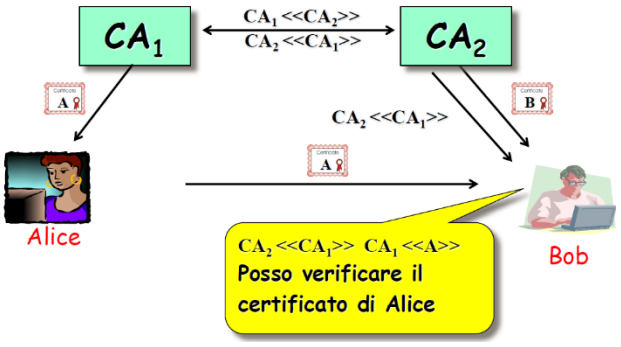
```
Certificate:
    Data:
        Version: v3 (0x2)
        Serial Number: 3 (0x3)
        Signature Algorithm: PKCS #1 MD5 With RSA Encryption
        Issuer: OU=Ace Certificate Authority, O=Ace Industry, C=US
        Validity:
            Not Before: Fri Oct 17 18:36:25 1997
            Not After: Sun Oct 17 18:36:25 1999
        Subject: CN=Jane Doe, OU=Finance, O=Ace Industry, C=US
        Subject Public Key Info:
            Algorithm: PKCS #1 RSA Encryption
            Public Key:
                Modulus:
                    00:ca:fa:79:98:8f:19:f8:d7:de:e4:49:80:48:e6:2a:2a:86:
                    ed:27:40:4d:86:b3:05:c0:01:bb:50:15:c9:de:dc:85:19:22:
                    43:7d:45:6d:71:4e:17:3d:f0:36:4b:5b:7f:a8:51:a3:a1:00:
                    98:ce:7f:47:50:2c:93:36:7c:01:6e:cb:89:06:41:72:b5:e9:
                    73:49:38:76:ef:b6:8f:ac:49:bb:63:0f:9b:ff:16:2a:e3:0e:
                    9d:3b:af:ce:9a:3e:48:65:de:96:61:d5:0a:11:2a:a2:80:b0:
                    7d:d8:99:cb:0c:99:34:c9:ab:25:06:a8:31:ad:8c:4b:aa:54:
                    91:f4:15
                Public Exponent: 65537 (0x10001)
        Extensions:
            Identifier: Certificate Type
                Critical: no
                Certified Usage:
                    SSL Client
            Identifier: Authority Key Identifier
                Critical: no
                Key Identifier:
                    f2:f2:06:59:90:18:47:51:f5:89:33:5a:31:7a:e6:5c:fb:36: 26:c9
    Signature:
        Algorithm: PKCS #1 MD5 With RSA Encryption
        Signature:
            6d:23:af:f3:d3:b6:7a:df:90:df:cd:7e:18:6c:01:69:8e:54:65:fc:06:
            30:43:34:d1:63:1f:06:7d:c3:40:a8:2a:82:c1:a4:83:2a:fb:2e:8f:fb:
            f0:6d:ff:75:a3:78:f7:52:47:46:62:97:1d:d9:c6:11:0a:02:a2:e0:cc:
            2a:75:6c:8b:b6:9b:87:00:7d:7c:84:76:79:ba:f8:b4:d2:62:58:c3:c5:
            b6:c1:43:ac:63:44:42:fd:af:c8:0f:2f:38:85:6d:d6:59:e8:41:42:a5:
            4a:e5:26:38:ff:32:78:a1:38:f1:ed:dc:0d:31:d1:b0:6d:67:e9:46:a8:dd:c4
```

Questo è un esempio di certificazione. Il periodo di validità va dal 17 ottobre ore 18:36:25 del 1997, fino al 17 ottobre ore 18:36:25 del 1999. La chiave pubblica usa PKCS #1 RSA Encryption. È presente l'esponente pubblico 65537 (0x10001) che corrisponde a 2^{16}-1, viene usato questo valore perché è facile farne l'esponenziazione. La firma dell'autorità si trova sotto a tutto e viene fatta utilizzando l'algoritmo PKCS #1 MD5 With RSA Encryption

Ci sono diverse codifiche per quanto riguarda i certificati, come DER in qui i caratteri sono binari e dunque non visibili con un text editor. Per leggere un file di questo genere bisogna usare un certificato codificato in Base64, che fa una codifica di 6 bit alla volta utilizzando valori Hash. Tutti i bit sono di lunghezza 6 → 2^6=64 e da questo deriva il nome Base64.

Nel mondo, non esiste una sola autorità di certificazione (chiamata CA per semplicità) che emette tutti i certificati. Ne esistono molte che si dividono in gerarchie. Essenzialmente esiste una CA che emette certificati per le CA di livello inferiore che a loro volta emettono CA per gli utenti finali o a CA che si trovano più in basso della gerarchia. Ad esempio:

Se Alice e Bob posseggono due certificati che derivano da 2 autorità di CA diverse, per poter dialogare tra loro, Bob dovrebbe mandare il suo certificato ad Alice, ma Alice non conosce l'autorità di certificazione di Bob e di conseguenza non può aprire il dialogo. Per poter verificare il certificato di Bob, l'idea è che CA_1 e CA_2 stabiliscano un accordo con cui entrambe le autorità mandano all'altro un certificato che stabilisce che CA_1 possa verificare certificati CA_2 e viceversa. In questo modo c'è la fiducia da entrambe le parti che consente ad Alice e Bob di poter verificare i certificati.



Gli usi comuni dei certificati sono su:

- **HTTPS**
- **Code Signing** (firma del codice)

Con **HTTPS** si rende sicura la connessione che avviene tra Browser e http Server. Un tipico scenario che può creare problemi si ha quando l'attaccante può controllare il sito web in cui l'utente sta navigando per poter intercettare informazioni private dell'utente(login, password, informazioni finanziarie, dati personali). L'utente può andare in un sito ingannevole ad esempio cliccando su un link malevolo presente in una mail di phishing. Può anche succedere quando l'utente è collegato a qualche HotSpot e poiché la connessione non è fidata, l'attaccante può riuscire a controllare varie informazioni, in maniera da reindirizzare l'utente in un sito che non è quello reale.

Con HTTPS, il server possiede un certificato in maniera tale che il browser quando tenta di collegarsi ad un sito ha la certezza che il sito è quello originale.

Per verificare che un sito abbia un certificato SSL ci sono quattro indizi visivi: lucchetto a fianco di una URL, prefisso https dell'URL invece di http, un sigillo di fiducia, alcune componenti degli indirizzi sono di colore verde (lucchetto, prefisso https, etc.)

Un **Code Signing Certificate** permette agli sviluppatori di firmare digitalmente il loro software prima della distribuzione via web. Questo processo implica:

- **Origine del contenuto**: gli utenti finali possono avere conferma che il software provenga dalla legittima fonte che lo ha firmato.
- **Integrità del contenuto**: gli utenti finali possono verificare che il software non è stato modificato da quando è stato firmato.