05 - Key Management

Problemi:

- Distribuzione di chiavi pubbliche
- Distribuzione di chiavi private tramite servizi sicuri
- Distribuzione di chiavi private tramite protocolli a chiave pubblica

Distribuzione di chiavi pubbliche

- Public announcement
- Public directory
- Certificati

Public Announcement

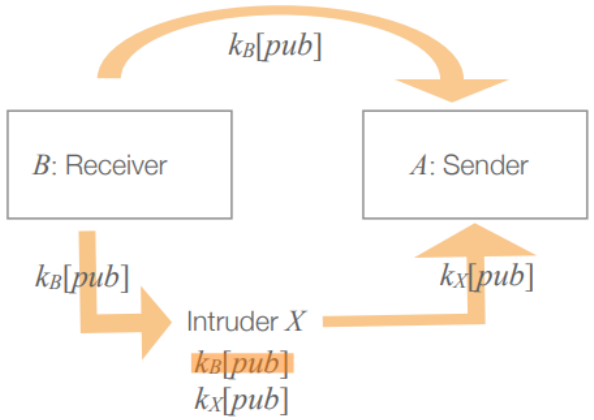
L'utente rende la sia chiave pubblica accessibile mettendola in uno spazio pubblico
Esempio: l'utente invia la chaive pubblica come allegato a tutte le email che invia. L'utente posta la chiave pubblica sul suo tsito o profiso social.
Tutti possono pubblicare la loro chiave pubblica, tutti possono accedere alle chiavi pubbliche degli altri.

Vantaggi:

- Semplice, veloce, non richiede interventi di terze parti
  Svantaggi:
- Nessuna garanzia: l'informazione pubblica può essere facilmente alterata
- Vulnerabile all'attacco **man-in-the-middle** (un utente può pubblicare la propria chiave come se fosse di qualcun'altro)

Esempio attacco man in the middle

- Avviene durante la pubblicazione della chiave
- X si inserisce nella comunicazione tra A e B
- Verso B egli pretende di essere A
- Verso A pretende di essere B
- X si preoccupa che tutte le comunicazioni tra A e B passino attraverso di lui



Public directory

La cartella è una lista di coppie **<utente, chiave_pubblica>**
La cartella deve essere mantenuta da una trusted party
Per pubblicare un record l'utente deve essere registrato (di persona o tramite un metodo sicuro)
Acesso:

- Consultare l'ultima local copy della directory ricevuta (periodicamente) dall'autorità (come un registro telefonico)
- Consultare la copia mantenuta dall'autorità **remotamente** (richiede autenticazione e protocolli di comunicazione sicuri)

Svantaggi:

- Richiede un'autorità imparziale e fidata
- La directory può essere compromessa
- Richiede protocolli di conicazione per pubblicare e accedere alle chiavi

Certificti

Autenticità della chiave certificata dall'autorità aggiungendo la propria firma
Garantisce l'identità delle parti e la validità delle chiavi pubbliche
Elimina l'attacco man in the middle (un malientenzionato non può sostituire la propria chaive pubblica con quella di qulcun'altro perchè non può firmarla senza conoscere la chiave privata dell'autorità)
Richiede un autorità imparziale e fidata

Management of (private) secret keys

n parti (clienti, server, utenti...) necessitano di comunicare in privato
Utilizzo di crittografie con chiave privata per stabilire canali sicuri di comunicazione
Se ogni comunicazione tra coppie è possibile e deve essere privata necessiteranno di $O(n^2)$ chiavi private
Per grandi n questo può diventare poco pratico visto che le chiavi private devono essere rimpiazzate dopo un po' di tempo.
Si può ridurre il numero di chiavi private a $O(n)$ se si utilizzano terze parti fidate.

Si assuma di avere un (fidato) Key Distribution Server (KDS) che condivide una chiave segreta diversa tra ogni parte.
A e B vogliono stabilire un canale di comunicazione sicuro tra di loro
Uno di loro chiede al KDS di generare una one-time session key da usare per la durata della comunicazione
Comunicazioni future tra A e B genereranno e useranno chiavi diverse

Protocollo base

- A e KDS condividono K_A
- B e KDS condividono K_B
- A invia a un messaggio a KDS: {A, "richiesta chiave di sessione per B"}
- KDS genera una nuova chiave di sessione K_S e la manda ad A:

$$C(K_A\{K_S, c(K_B, K_S)\}$$

- A salva K_S e la invia a B: $C(K_B, K_S)$
- B salva K_S
- A e B possono scambiarsi messaggi confidenziali usando K_S

Commenti:
B non riceve K_S direttamente dal KDS ma da A
Quindi:

- B non può sapere con sicurezza se il messaggio è stato davvero inviato da A
- Vulnerabile a un replay attack (un malintenzionato può registrare e inviare un messaggio, già cifrato, in un futuro, come se fosse nuvo)

Needham-Schroeder Protocol

- A e KDS condividono K_A
- B e KDS condividono K_B

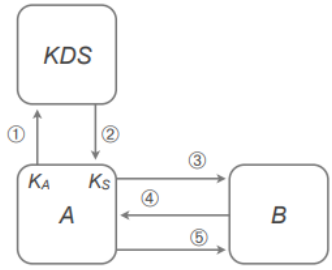
1. A invia a un messaggio a KDS: {A, "richiesta chiave di sessione per B", N_1}
2. KDS genera una nuova chiave di sessione K_S e la manda ad A:

$$C(K_A\{K_S, A, B, N_1, C(K_B, \{K_S, A\})\}$$

3. A salva K_S e la invia a B: $C(K_B, \{K_S, A\})$
4. B salva K_S e invia ad A: $C(K_S, N_2)$ (challenge)
5. A risponde a B: $C(K_S, N_2 + 1)$ (response)
6. A e B possono scambiarsi messaggi confidenziali usando K_S

N_1 e N_2 sono chiamati "*nonces*" (number used once) e prevengono i replay attack
La "*challenge-response*" handshake negli step 4 e 5 servono a confermare che A e B sono presenti e voglion comunicare oltre a sincronizzare le comunicazioni per usare la stessa session key.
Questa è la base della **Kebernos authentication protocol**

Commenti
A si fida di KDS ed è certo che ha riccevuto la session key da KDS perchè il messaggio è criptato con K_A
Il nonce N_1 serve per verificare la corrispondenza della session key ricevuta con la richiesta fatta da A nello step 1
A è certo di rivelae K_S solo a B perchè manda K_S criptato con K_B che solo B può decriptare
B si fida di KDS e KDS garantisce a B che la chiave può solo essere usata per comunicare con A
B può rilevare un replay attack ed è sicuro di stare comunicando con A



Attacchi possibili

- Il messaggio 3: $C(K_B, \{K_S, A\})$ non è protetto da nonce
- Supponendo che X sia stato in grado di craccare la session key K_S dal run del protocollo della settimana prima e che abbia salvato il messaggio 3
- X può ora rimandare quel messaggio e far credere ad A di stare parlando con B

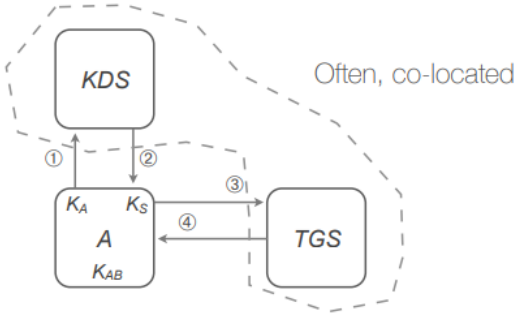
3. X invia a B: $C(K_B, \{K_S, A\})$
4. B invia ad X: $C(K_S, N_2)$ (challenge)
5. X risponde a B: $C(K_S, N_2 + 1)$ (response)

- Non esiste alcun modo per B di sapere che K_S che riceve nel messaggio 3 è quello corrente.
  Si sitema aggiungendo un nonce nel paso 3.

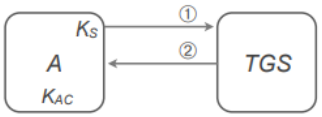
Se A vuole comunicare con un diverso principal C, deve riavviare il protocollo con KDS per generare nuove session key K_S usando la sua chaive segreta K_S
Visto che lo scambio di chiavi è basato su un segreto, questo può risultare in un maggiore rischio che può essere compromesso.

Kerberos

Sviluppato dal MIT nel 1980 per essere usato come un **authentication service** distribuito in un ambiente accademico.
Ogni principal inizialmente condivide un secret key (password) con KDS
Per ridurre l'esposizione della chaive segreta, KDS viene usato solo una volta per login
Tutte le comunicazioni in una singola sessione sono sicure tramite chiavi ottenute da un Ticket Garanting Server (TGS)



Se A vuole comunicare con un diverso principal C, deve riavviare il protocollo con TGS (non KDS) per generare una nuova session key K_{AC} usando la chiave K_S (non la chiave condivisa K_A)



In sistemi davvero grandi, KDS potrebbe essere un bottleneck
KDS può essere replicato per ottenere un'aumento di performance e reliability usando uno schema master-slave.

Vantaggi:

- garantisce confidenzialità e autenticazione
- Raggiunge buone performance anche in presenza di più parti e frequenti scambi di chiave
- Per n parti riduce il numero di chiavi segrete a $O(n)$
  Difetti:
- richiede l'esistenza di un fidato e reliable KDS

Private key vs Public-Key Cryptography

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Soluzioni ibride

La crittografia con chaive pubblica è circa 1000 volte più lenta della crittografia con chiave privata.
Soluzioni ibride:

- utilizzo di **asymmetric criptography** solo una volta (all'inizio) per concordare su una chiave privata.
- Quindi passare a **symmetric cryptography** (usando la chiave concordata prima) per le comunicazioni future

1. A genera $(K_A[\text{pub}], K_A[\text{priv}])$
2. A invia a B: $\{K_A[\text{pub}], A\}$
3. B genera la session key K_S
4. B invia a A: $C(K_A[\text{pub}], K_S)$
5. A decripta per ottenere $K_S = D(K_A[\text{priv}], C(K_A[\text{pub}], K_S))$
6. A cancella $(K_A[\text{pub}], K_A[\text{priv}])$ B cancella $K_A[\text{pub}]$
7. A e B passano alla crittografia simmetrica usando la session key K_S

Garantisce cofidenzialità e autenticazione
Rimane soggetto ad attacchi man in the middle
Soluzione generale basata su **certificati** per garantire mutua autenticazione evitando man in the middle attacks
Base per SSL