

Crittografia e sicurezza delle reti

Autenticazione

- Protocolli di autenticazione
- X.509, Kerberos

Autenticazione

Alice vuole provare la sua identità a Bob per ottenere un servizio, avere accesso ad una risorsa ecc.

Bob può chiedere:

- Chi sei? (Dimostra che sei Alice)
- Chi è mai Alice?

Trudy cerca di impersonare Alice:

- Una sola volta
- Sempre

Scenari di Autenticazione

Locale

- autenticatore umano
- dispositivo di autenticazione

Remota

- autenticatore umano
- ambiente aziendale (es. LAN)
- ambiente E-commerce
- TV via cavo/Satellite: Pay-per-view;
(verifica sottoscrizione)
- login remoto o e-mail da internet caffè

Ambiente

- chiuso (es. aziendale): chiave segreta
- aperto: chiave pubblica

Ambienti chiusi

Autenticazione remota in un'azienda/società

- Autenticazione iniziale è risolta da una terza persona di fiducia, Carolina
- Carolina distribuisce quanto serve all'autentica in modo sicuro, es. a mano o usando linee autenticate e confidenziali
- Gli attacchi di Trudy utilizzano la connessione Alice-Bob

Autenticazione di persone

Utilizzano

- cosa sai (passwords)
- cosa hai (smart card)
- chi sei (dispositivi biometrici)
- dove sei (luogo fisico)

Basata su indirizzo

.rhosts

node, username

/etc/hosts.equiv

hosts fidati

Pericoli:

un attacco con successo su un utente mette in
pericolo l'intero sistema

di solito: A si fida di B, implica che B creda a A
address spoofing

Gettoni di Autenticazione

Cosa hai

- Smart cards:
 - Challenge/response
- Calcolo crittografico :
 - Interazione

Tecniche Biometriche

Accuratezza:

- errori nell'accettare e nel rifiutare.
- avversario può imporre impostori? (Gemelli Identica, familiari, etc.)

Esame della retina, viso, impronta digitale (fingerprint reader), o basate su altre caratteristiche (es. voce, tempi di battitura, firma autografa ecc.)

Impronta digitale

Vulnerabile:

- Impronte false e/o di persone morte

Non stabile:

- Bambini
- Persone con alta probabilità di riportare danni alle dita

Riconoscimento Vocale

Singola frase:

- uso registratore per falsificazione

Stabilità:

- Rumore di fondo
- Mal di gola
- Uso telefono

Tempo di battitura (keystroke timing)

Ogni persona ha stile e velocità di battitura diversi

- movimenti della mano e delle dita

Adeguatezza:

- Autenticazione "locale" (si evitano ritardi della rete)

Firma autografa

Il riconoscimento automatico della firma è di qualità inferiore a quella delle macchine

Si aggiunge informazione dinamica dei movimenti

- Firma su electronic tablet

Autenticazione con chiave

Scenari di Autenticazione basata su conoscenza di una chiave (pubblica o segreta- possibilmente memorizzata su smart card)

1. utenti che condividono una password fra loro
2. utenti che condividono una password con autorità fidata (authentication servers)
3. utenti che hanno una chiave pubblica

Password e chiavi

- Generazione sicura: utilizzo di casualità, dove? (lezione successiva)
 - Necessità di molte chiavi
 - Chiavi di qualità
- Memorizzazione sicura: come garantire sicurezza fisica?
- Distribuzione di chiavi
 - come scambiare chiavi segrete fra due utenti (Diffie-Hellman)
 - come distribuire chiavi pubbliche

Password e chiavi

Esseri umani :

- Chiavi brevi, usate direttamente o come chiavi per ottenere chiavi più lunghe
- Necessità di molte chiavi (es. sistema distribuito)

Computers:

- chiavi segrete di qualità (lunghe e non prevedibili)
- chiavi nascoste e non accessibili ad utenti del sistema
- chiavi non sono memorizzate in chiaro nel server (crittografate con password, one-time password)

Attacchi su password

- Password umane sono spesso facili da indovinare (attacco del dizionario: attaccante prova password più probabili; facile in un insieme di persone ampio)
- Cavallo di Troia (Trojan horse): Una finestra falsa di login per catturare passwords (oggi su web noto come phishing)
- **Contromisure:**
 - Rendi difficile apparire prompt login
 - Usa interruzioni
 - Limita login da programmi utenti (blocco dopo un certo numero di accessi rifiutati)

Password: problemi

Eavesdropping: attaccante è in ascolto sul canale

- password non si devono trasmettere in chiaro
- procedure di autenticazione devono essere ogni volta diversa (per evitare attacchi di replay)

Sicurezza delle password:

- attaccante può accedere al data base delle password: password crittate in memoria
- password utenti brevi; generazione password sicure

Sicurezza fisica: Unix

Idea: non si memorizza la chiave ma un'informazione ottenuta dalla chiave. Sia K la chiave

- Unix: memorizza codifica DES (modificato) con chiave K della stringa $00\dots0$

Problema: attacco del dizionario: le chiavi usate sono prevedibili e quindi utente che accede ad un database trova facilmente una password

per evitare attacchi del dizionario si usa SALE (Salt): si memorizza codifica di $0000.00\langle\text{num.casuale}\rangle$ (sale)

Sicurezza fisica: Lamport

Si usa funzione hash crittografica H ; utente A sceglie chiave K e intero n e si memorizza nel database:

$A, n, H^n(K) = H(H(H..(K)...))$ (hash calcolato n volte)

- Ad ogni richiesta utente il **Sistema** fornisce n
- **Utente** calcola $H^{n-1}(K)$ e lo invia al sistema;
- **Sistema** calcola $H(H^{n-1}(K))$ e verifica; se OK n è decrementato di uno e si memorizza $H^{n-1}(K)$

Quando n diventa zero : nuova password

Nota: 1. sistema non conosce password;

2. se si incrementa invece di decrement. non funziona.

Tecniche di autenticazione

Autenticazione basata su conoscenza di una chiave (pubblica o segreta)

Si utilizzano

- **timestamp**
- **nonce** (o **sfida**) : numero casuale scelto da chi deve autenticare per sfidare chi deve essere autenticato che conosce la chiave
- **numeri di sequenza** nei protocolli

Crittografia vs. autenticità

Gli attacchi si basano sul fatto che uno o più messaggi inviati non sono autentici (non sono stati inviati durante l'esecuzione del protocollo sotto attacco)

- o sono vecchi messaggi o sono falsi
- garantire l'autenticità del mittente e l'integrità dei messaggi

Ricorda: la codifica crittografica di un messaggio non ne garantisce l'autenticità

Autenticazione - Symm.Key

A e B condividono una chiave K comune
Autent. unilaterale con timestamps

1. A invia a B: $K(t,A,B)$ - B decodifica e verifica t (verifica che t non sia scaduto)

1. A invia a B: $\text{Hash}(K,t,A,B)$ - B calcola hash e verifica t (verifica se t è ragionevole)

come B verifica correttezza? prova tutti i possibili timestamps? No meglio

1. A invia a B: $t, \text{Hash}(K,t,A,B)$ - B t (verifica se t è ragionevole) calcola $\text{Hash}(K,t,A,B)$ e verifica uguaglianza

Autenticazione - Symm.Key

A e B condividono una chiave K comune

Autent. unilaterale con timestamps

1. A invia a B: $t, K(t, A, B)$ - B decodifica e verifica t
efficiente e semplice

MA

- necessita orologi sincronizzati
- se Alice usa la stessa chiave con più server, cosa succede?
- se attaccante convince Bob a aggiustare/modificare clock?

Autenticazione - Symm.Key

Utenti condividono una chiave K comune fra A e B

- Autent. unilaterale con nonce (sfida)
 1. B invia a A N
 2. A invia a B $K(N, B)$ - B verifica che A risponde alla sfida
- Oppure
 1. B invia a A N
 2. A invia a B $\text{Hash}(K, N, B)$ - B verifica che A risponde alla sfida
- Oppure
 1. B invia a A $K(N, B)$
 2. A invia a B N - B verifica che A risponde alla sfida

Autenticazione - Symm.Key

Nonce devono essere generati in modo casuale.
E' importante che nonce siano non predicibili. Infatti

1. B invia a A $K(N,B)$
2. A invia a B N - B verifica che A risponde alla sfida

Ad esempio se B incrementa N di uno ogni volta
allora T osserva una autentica e poi può predire
N si autentica al posto di A (facile)

Autenticazione - Symm.Key

Nonce devono essere generati in modo casuale.

- Autent. unilaterale con nonce (sfida)
 1. B invia a A N
 2. A invia a B $K(N,B)$ - B verifica che A risponde alla sfida

Assumi che B incrementa di uno ogni volta

In questo caso

T aspetta che A si autentichi con T stessa.

quando succede fornisce N+1 a A e riceve la risposta
che può usare con B!

Autenticazione - Symm.Key

Nonce devono essere generati in modo casuale. In alcuni casi la generazione di un nonce può essere prevedibile e il protocollo è affidabile

Ad esempio si assume di avere due chiavi K e K'

- Autent. unilaterale con nonce (sfida)
 1. B invia a A $K(N,B)$
 2. A invia a B $K'(N,B)$ - B verifica che A risponde alla sfida

Assumi che B incrementi N di uno ogni volta

In questo caso T che non conosce le chiavi K e K' non sa calcolare N e quindi anche $K'(N,B)$

Autenticazione - Symm.Key

Utenti condividono una chiave K comune fra A e B

- Mutua autent. con nonce
 1. B invia a A N
 2. A invia a B $K(N,N',B)$
 3. B invia a A $K(N,N')$

NOTA:

- è possibile usare sia nonce che timestamps contemporaneamente
- condivisione di chiave fra tutte le coppie di utenti (esplosione combinatorica del numero di chiavi)

Autenticazione EKE: Encrypted Key Exchange

Problema: attacco dizionario su chiavi deboli

EKE:

- protegge rispetto ad attacco dizionario
- autenticazione bilaterale
- permette di definire chiave di sessione

Scenario:

- Utente e server condividono una chiave
- Utilizzano la password per definire una chiave di sessione (simile a Diffie Hellman)

Autenticazione -EKE

1. sia $w = \text{Hash}(\text{password})$
 2. Sia p primo e g generatore di Z_p
 3. A al server: $A, E_w(g^a \bmod p)$
 4. Server a A: $E_k(\text{sfida-1}), E_w(g^b \bmod p)$
 5. A al server: $A, E_k(\text{sfida-1}, \text{sfida-2})$
 6. Server a A: $E_k(\text{sfida-2})$
- Chiave di sessione $k = g^{ab} \bmod p$

Autenticazione -EKE

Fornisce soluzione per

- attacchi di replay (a diverso di volta in volta)
- passo 1: attacchi dizionario (anche se la password è debole a casuale implica che non si può calcolare a)
- passi 3 e 4: permettono di convincere A e server che l'altro conosce la chiave di sessione k

Nota: se attaccante indovina password comunque può sostituirsi a A con un attacco attivo

Scenari di autenticazione

Autenticazione basata su conoscenza di una chiave (pubblica o segreta- possibilmente memorizzata su smart card)

1. utenti che condividono una password fra loro
2. utenti che condividono una password con autorità fidata (authentication servers)
3. utenti che hanno una chiave pubblica

Intermediari fidati

Problema: Non è possibile la condivisione di una chiave fra tutte le coppie di utenti (troppe chiavi)

Soluzione: Utenti che condividono una password con autorità fidata - *C* è server di autenticazione anche

Key distribution center (KDC)

- A e B condividono ciascuno una chiave (K_{AC} e K_{BC}) con *C* che è autorità fidata
- A e B non sono necessariamente utenti umani ma, in genere, entità del sistema
- obiettivi
 - autenticazione di A (opzionale anche di B)
 - opzionale: stabilire chiave di sessione fra A e B (si

Server di Autenticazione

Obiettivo: Alice e Bob devono generare una chiave segreta K - di sessione, temporanea

Alla fine del protocollo

- solo A e B conoscono K (a parte *C* intermediario fidato)
- ciascuno è certo che l'altro conosce K
- K è una chiave nuova (non utilizzata prima)

Server di Autenticazione

Possibilità di Trudy (attaccante)

- T è un utente legittimo del sistema
- può intercettare e fare sniffing e spoofing di messaggi
- può partecipare in una o più sessioni con A e B
 - eseguire le diverse esecuzioni in modo interleaved (ad es, inizia A poi T si sostituisce e attiva altre autenticazioni)
 - indurre A e/o B a iniziare sessioni con T
- può conoscere vecchie chiavi di sessione

Server di Autenticazione

Limiti dell'attaccante.

Trudy

- non può indovinare numeri casuali (generati da A o B)
- non conosce chiavi KAC e KBC (in generale non conosce le chiavi pubbliche o private di altri utenti)
- non è in grado di decodificare messaggi codificati con chiavi non note

Protocollo Needham-Schroeder

N, nonce: numero casuale

Protocollo N.S. basato su Challenge-Response :

1. A genera N (nonce) e invia a C $\langle A, B, N \rangle$
2. C genera K e manda a A $KAC(N, K, B, KBC(K, A))$
3. A decodifica, verifica N e identità B e manda a B $KBC(K, A)$
4. B riceve, decodifica, verifica identità A, crea nonce N' e invia a A $K(\text{sono } B, N')$
5. A invia a B $K(\text{sono } A, N'-1)$

NOTA: B non è sicuro di ricevere il messaggio da A (passo 4)

Attacco Protocollo NS

Problema: - B non è sicuro di ricevere il messaggio da A
- T usa vecchie chiavi di sessione (decodificate) e si finge A

- A genera N (nonce) e invia a C $\langle A, B, N \rangle$
- C genera K e manda a A $KAC(N, K, B, KBC(K, A))$
da qui in poi T si sostituisce a A
- A decodifica, verifica N e identità B e manda a B $KBC(K, A)$ - T intercetta e (come A) invia a B $KBC(K', A)$ - K' vecchia chiave decodificata da T
- B riceve, decodifica, verifica identità A crea nonce N' e invia a T (e non a A) $K'(\text{sono } B, N')$
- T invia a B $K'(\text{sono } A, N'-1)$

Protocollo modificato - timestamp

Si assume tempo universale:

1. A genera N invia a C A, N
2. C invia a A $KAC(N, B, K, KBC(K, A, t))$
3. A invia a B $\langle K(T), KBC(K, A, t) \rangle$
4. B invia a A $K(T+1)$

t : periodo di validità della chiave K

T : tempo a cui avviene autentica

Protocollo NS, esteso - nonce

1. A dice a B: voglio parlare con te
2. B genera N (nonce) e invia a A $KBC(N)$
3. A genera N' e invia a C $\langle N', KBC(N) \rangle$
4. C genera K e manda a A $KAC(N', K, B, KBC(K, A, N))$
5. A decodifica, verifica N' e identità B e manda a B $KBC(K, A, N)$ e $K(N'')$
6. B decodifica, verifica identità A crea nonce N''' e invia a A $K(N''-1, N''')$
7. A invia a B $K(N'''-1)$

NOTA: passo 4 C certifica A per B (la sfida è di B)

Passi 5,6,7 A e B eseguono mutua autenticazione con chiave comune

Protocollo NS - timestamp e nonce

Challenge-Response + timestamp (t):

1. A genera N (nonce) e invia a C A,B,N
2. C genera K e manda a A
 $\text{Sig}_C(N,K,B,t,KBC(K,A,t))$
3. A decodifica, verifica N e ident. B e manda a B $\text{Sig}_C(K,A,t)$
4. B decodifica, verifica id. A, crea nonce N' e invia a A K(sono B,N')
5. A invia a B K(sono A,N'-1)

Necessità di clock sincroni tra A e B

Challenge-response Symm.Key

Deve essere garantita l'integrità dei dati

- timestamps (un messaggio è valido solo all'interno di una piccola finestra temp.)
- numeri di sequenza (A e B ricordano numeri per evitare attacchi di replay)
- attento uso dei nonce

Autenticaz. con autorità fidata C: attacco-1

A e B condividono ciascuno una chiave (KAC e KBC) con C che è autorità fidata

1. A manda a C A,B
2. C genera K a caso, e manda a A KAC(K) e KBC(K)
3. A calcola K e manda a B C, A,KBC(K)
4. B decodif. KBC(K) e trova K e manda a A K(Hello A, sono B)

Problema: B non è certo di ricevere il messaggio originato da C

Attacco-1 (cont.)

Problema: B non è certo di ricevere il messaggio originato da C.

Assumi T intercetta conversazione

1. A manda a T (al posto di A manda a C) : A,B subito dopo T manda a C A,T
2. C genera K e manda a A KAC(K) e KTC(K)
3. A invia a B C,A,KTC(K) - T intercetta
4. T invia a A K(Hello A, sono B)

Correz: passo 1: A manda a C $\langle A, KAC(B) \rangle$ (C ha garanzia che A vuole parlare con B)

Attacchi - 2

Considera protocollo modificato

1. A manda a C $\langle A, KAC(B) \rangle$
2. C genera K, e manda a A $KAC(K)$ e $KBC(K)$
3. A calcola K e manda a B $C, A, KBC(K)$
4. B decodif. $KBC(K)$ e trova K e manda a A $K(\text{Hello } A, \text{ sono } B)$

Nota al passo 1 T non sa con chi vuol comunicare A

Attacchi - 2 (cont.)

Considera protocollo modificato

1. A manda a C $\langle A, KAC(B) \rangle$ - T (come A) intercetta messaggio di A e manda a C: $A, KAC(T)$ (noto da autenticazione precedente di A e T)
Nota T non sa con chi vuol comunicare A
2. C genera K a caso, e manda a A $KAC(K)$ e $KTC(K)$
3. A calcola K e manda a B $C, A, KTC(K)$
4. T intercetta messaggio, scopre con chi vuol parlare A e si sostituisce a B e manda a A $K(\text{Hello } A, \text{ sono } B)$

T conosce dopo che A vuole comunicare con B

Challenge-response Symm.Key

Deve essere garantita l'integrità dei dati

- timestamps (un messaggio è valido solo all'interno di una piccola finestra temp.)
- numeri di sequenza (A e B ricordano numeri per evitare attacchi di replay)
- attento uso dei nonce

Cosa è Kerberos?

Sistema, che fornisce autenticazione
crittografica in ambienti distribuiti

- Garantisce accesso sicuro alle risorse di rete
- Riduce il numero di password che devono essere gestite da utenti e amministratori di rete

Riferimento:

- <http://web.mit.edu/kerberos/www/dialogue.html>
- free download negli USA e in Canada
- molto usato (es. windows2000)

Protocollo Kerberos

Progetto Athena del MIT

- Scenario: A vuole accedere i servizi di B
 - autenticazione di A
 - opzionale: B si autentica
 - opzionale: stabilire chiave di sessione
- C è autorità fidata e condivide chiavi con A e B
- Idea: uso biglietti di durata prefissata (da C)

Kerberos

- KDC (Key Distribution Center) sono 'fisicamente' sicuri
- Le librerie di Kerberos sono distribuite su tutti i nodi con utenti, applicazioni e altre risorse del sistema
- Tutte le comunicazioni sono sicure rispetto ad attacchi alla riservatezza e all'integrità dei messaggi
- Kerberos fornisce le seguenti applicazioni
 - telnet
 - rtools (rlogin, rcp, rsh)
 - Network file systems (NFS/AFS)

Preliminari

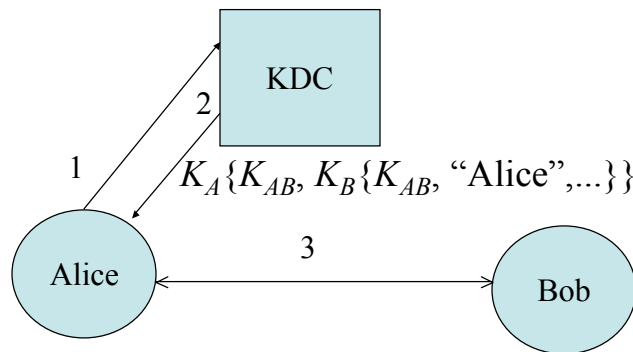
Ogni utente (principal) ha una chiave segreta master

- per gli utenti umani viene derivata dalla password
- per le risorse del sistema le chiavi sono configurate nell'applicazione

Ogni principal è registrato presso il KDC

Tutte le master keys sono codificate nel KDC database, codificate con la master key del KDC

Biglietti Alice, Bob e KDC



Biglietto (Ticket) di Bob:

$K_B\{K_{AB}, \text{"Alice"}, \dots\}$

solo Bob decodifica e verifica il contenuto)

Prot. Kerberos (vers. semplific.)

Richiesta di servizio utilizzando un biglietto TicketB

1. A invia a C A,B,N
2. C invia a A TicketB, KAC(K,N,"tempo di vita",B)
3. [A verifica N e conosce tempo di vita del ticket]
A invia a B TicketB, K(A,tA) [autenticatore]
4. [B verifica che ident.di A in TicketB e in autent. coincidano, validità timesptamp in autent. e che il suo clock preceda L]
B invia a A K(tA) [in modo tale da mostrare conoscenza di K]

TicketB = KBC(K,A,"tempo di vita"), N nonce,
K chiave sess. "tempo di vita"= validità ticket, Uso di indirizzo rete del mittente

Chiavi di Sessione e Ticket-granting Ticket (TGT)

Messaggi tra host e KDC possono essere protetti usando la master key del principal

Per ogni richiesta a KDC:

- Chiedere ogni volta la password
- Ricordare la password
- Ricordare la master key derivata dalla password

Tutte le soluzioni sono inadeguate!

Chiavi di Sessione e TGT...

Per ridurre uso della password e/o della master key

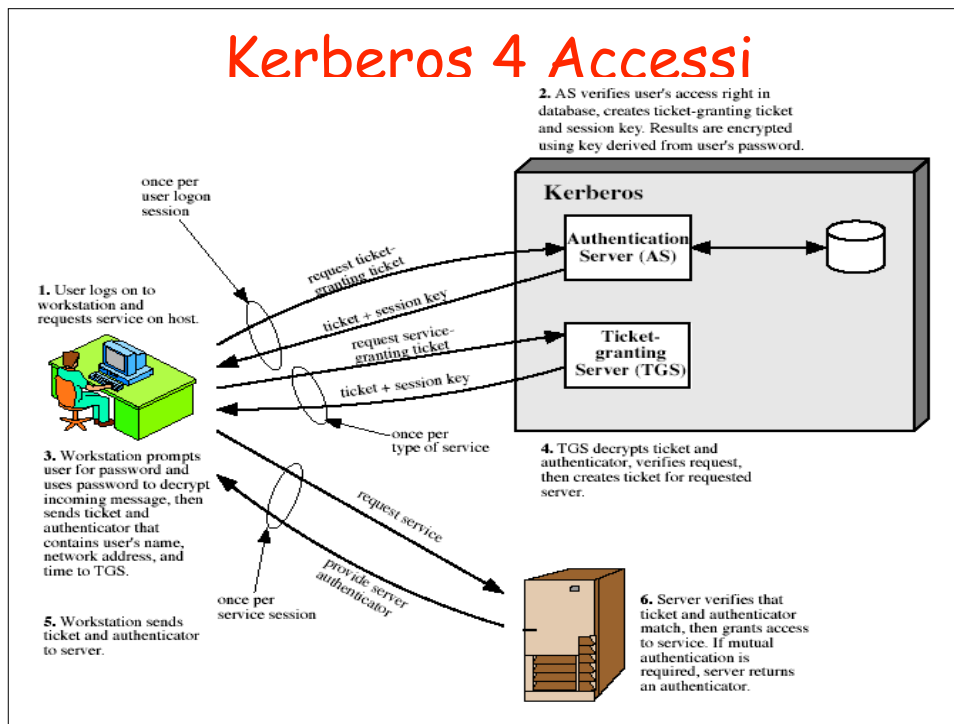
- Al tempo del login iniziale una chiave di sessione per il principal S_A (per Alice) viene richiesta al KDC
- S_A ha un periodo di validità ridotto
- Un TGT per Alice viene anche fornito dal KDC, che include la chiave di sessione S_A e le informazioni che identificano Alice (codificando con la chiave master)

Chiavi di Sessione e TGT...

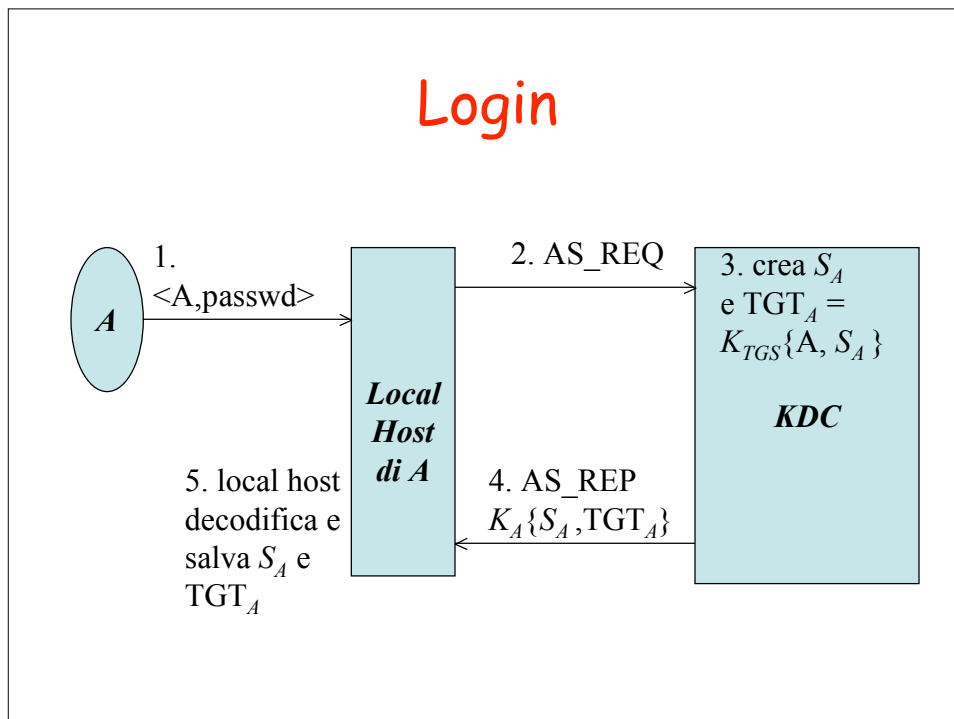
- Il client Kerberos di Bob decodifica e memorizza:
 - S_B , per messaggi successivi con KDC
 - TGT, per notificare KDC utilizzo S_B
- Nuove richieste al KDC devono includere TGT nel messaggio di richiesta
- Nuovi ticket dal KDC sono decodificati con S_B

Non si memorizzano password

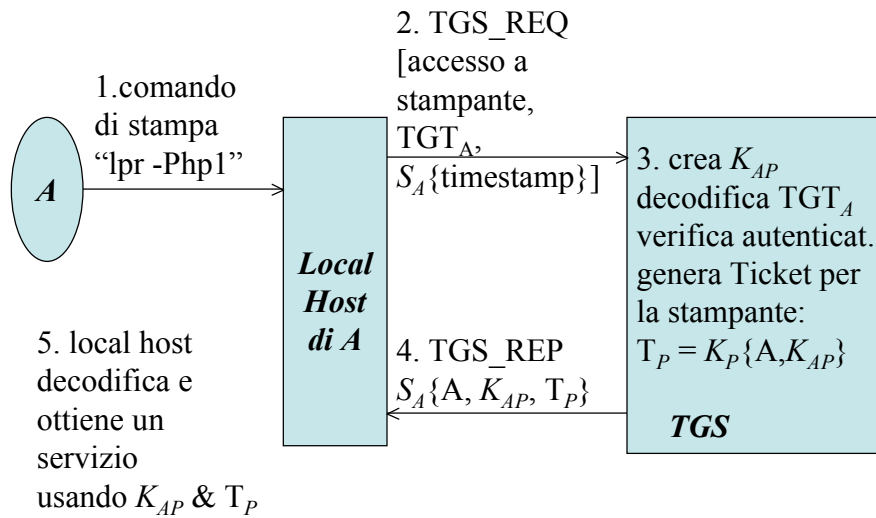
Kerberos 4 Accessi



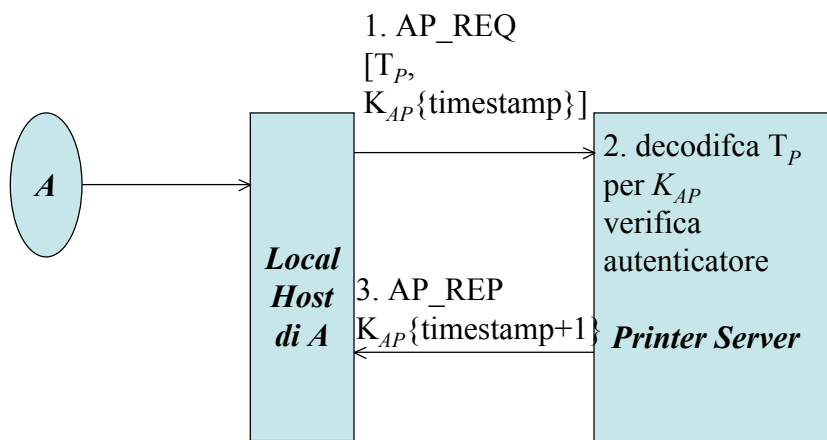
Login



Richiesta di Ticket...



Uso del Ticket per la stampa



Autenticazione e Sincronizzazione

Autenticatore == $K_x\{\text{timestamp}\}$

- Clock sincrono globale è implicito
- Lo scopo principale dell'autenticatore è eliminare
 - replay di vecchi messaggi allo stesso server (messaggi con vecchio timestamp sono scartati)
 - replay di richieste ad un server su altri server (server farm)
 - autenticatore NON garantisce integrità dei dati (serve un MAC)

Come si ottiene un TGT

1. A a KDC : A, TGS, Tempo1
2. KDC a A : (solo A conosce KA)
 $KA[KATGS, TGS, Tempo2, Tempo\ vita, TGT]$

TGT= Ticket Granting Ticket
= $KTGS[KATGS, A, IndirizzoA, Tempo\ autor., Tempo\ di\ vita]$

Come si ottiene un Ticket T per un Server (es. stampante)

1. A a TGS : Server, TGT, AutenticatoreA
2. TGS a A : KBTGS[KAServer, Server, Tempo, T]

T= Ticket

=

KServer[KAServer, A, IndirizzoA, Server, Tempo autor., Tempo di vita]

AutenticatoreA = KATGS[A, IndirizzoA, Tempo in via]

Come si ottiene un servizio

1. A a Server : T, AutenticatoreA
2. Server a A : KAServer[N+1]

T= Ticket

=

KServer[KAServer, A, IndirizzoA, Server, Tempo autor., Tempo di vita]

AutenticatoreA = KBTGS[A, IndirizzoA, N]

NOTA: il messaggio 2 fornisce autentica Server (perché?)

KDC e TGS

- KDC e TGS sono simili (o identici) perchè due entità?
 - motivi storici
 - un KDC può servire diversi sistemi applicativi (1 KDC e diversi TGS)
- Copie multiple di KDC - disponibilità e prestazioni
- Problemi di consistenza dei database dei KDC
 - un singolo KDC è l'unico punto di aggiornamento delle informazioni sui principal
 - periodicamente si esegue un download per gli altri KDC

Kerberos - Prestazioni

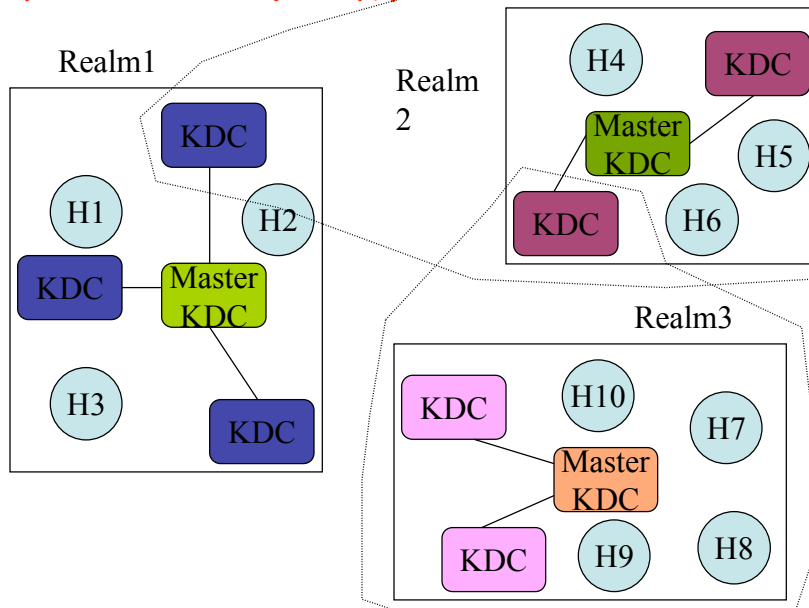
- KDC devono mantenere solo TGT e tickets.
- Il carico maggiore è sui client.
- KDC è coinvolto solo all'inizio nella fornitura del ticket
- KDC usa solo informazioni statiche (possibilità di repliche dei KDC).

Kerberos: Regni (Realms)

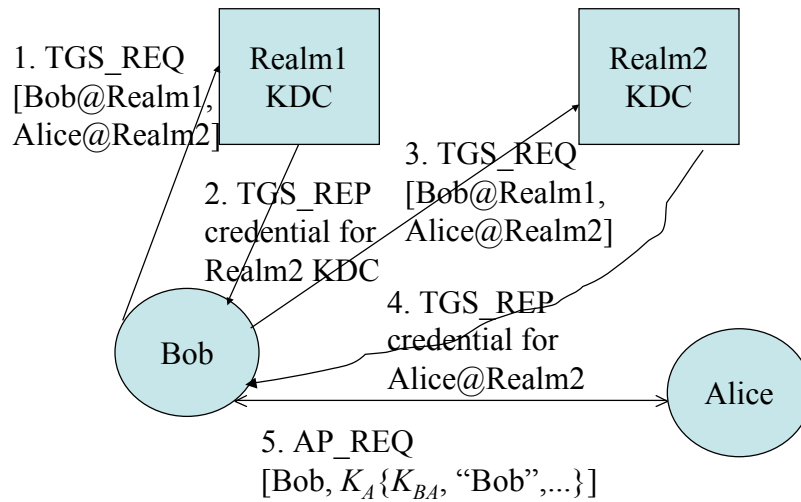
Problemi di dimensionamento e affidabilità suggeriscono di non usare un solo dominio ma più domini (Reami)

- Ogni reame ha una diversa master KDC
- KDC, ma tutte condividono la stessa master key del KDC
- Due KDC in differenti reami hanno diversi database delle master key degli utenti

Kerberos V. 4: Modello



Autenticazione fra Reami



Scenari di autenticazione

Autenticazione basata su conoscenza di una chiave (pubblica o segreta- possibilmente memorizzata su smart card)

1. utenti che condividono una password fra loro
2. utenti che condividono una password con autorità fidata (authentication servers)
3. utenti che hanno una chiave pubblica
 - protocolli si basano sulla firma di messaggi che rispondono alle sfide e/o contengono timestamp
 - autorità di certificazione garantiscono autenticità chiavi pubbliche

Autenticazione con chiave pubblica

Idea utilizzare messaggi firmati contenenti timestamp o risposta alle sfide

Problema: come essere sicuri della chiave pubblica dell'altro? Esempio

1. A vuole autenticarsi con B e KPA ma non conosce la chiave pubblica di B KPB
2. Sia KPT la chiave pubblica di Trudy
3. Se Trudy riesce a far credere a A che la chiave pubblica di B sia KPT allora inganna A

Soluzione: autorità che certifica le chiavi pubbliche degli utenti

Autenticazione con chiave pubblica Protocollo Needham-Schroeder

KPA: chiave pubblica di A, Sig_C firma di C (autorità che certifica la chiave pubblica degli utenti)

1. A invia a C $\langle A, B \rangle$
2. C a A : B, KPB, Sig_C(KPB, B)
3. A verifica firma C, crea nonce N e manda a B KPB(N, A)
4. B decodifica e verifica A id. e manda a C $\langle A, B \rangle$
5. C invia a B: A, KPA, Sig_C(KPA, A)
6. B verifica firma C e chiave KPA, crea nonce N' e manda a A KPA(N, N')
7. A decodifica verifica e manda a B KPB(N')

Prot. N-S public key - Attacco

- Trudy è utente del sistema che può effettuare autenticazione con A,B e C (C autorità certificazione)
- Due esecuzioni del protocollo interleaved: R1 A e T si autenticano; in R2 T si autentica come A con B, Man in the middle attack
- T è in grado di indurre A a iniziare sessione con T
- I passi 1,2,4,5 permettono di ottenere chiavi pubbliche
- I passi 3,6,7 eseguono l'autenticazione

Prot. N-S public key - Attacco

Nei passi 1, 2, 4 e 5 si conoscono chiavi pubbliche

Si considerano solo passi 3,6,7 di R1 e R2:

- a. A-->T : passo 3 di R1 invio di $KPT(N,A)$
- b. T-->B: passo 3 di R2 invio di $KPB(N,A)$
- c. B-->T: passo 6 di R2 invio di $KPA(N',N)$
- d. T-->A: passo 6 di R1 invio di $KPA(N',N)$
- e. A-->T: passo 7 di R1 invio di $KPT(N')$
- f. T-->B: passo 7 di R2 invio di $KPB(N')$

B crede di parlare con A e condividere con A i nonce segreti

Protocollo Needham-Schroeder con public key - correz.

1. A invia a C $\langle A, B \rangle$
2. C a A : $\text{Sig}_C(\text{KPB}, B)$
3. A verifica firma C crea nonce N e manda a B $\text{KPB}(N, A)$
4. B decodifica e verifica A id. e manda a C $\langle A, B \rangle$
5. C invia a B $\text{Sig}_C(\text{KPA}, A)$
6. B verifica firma C crea nonce N' e manda a A $\text{KPA}(\text{Bob}, N, N')$
7. A decodifica verifica e manda a B $\text{KPB}(N')$

Prot. N-S public key - correz

Si considerano solo passi 3,6,7 di R1 e R2:

- a. A \rightarrow T : passo 3 di R1 invio di $\text{KPT}(N, A)$
- b. T \rightarrow B: passo 3 di R2 invio di $\text{KPB}(N, A)$
- c. B \rightarrow T: passo 6 di R2 invio di $\text{KPA}(N', N)$
- d. T \rightarrow A: **passo 6 di R1 PRIMA invio di $\text{KPA}(N', N)$ - ORA T non può mandare $\text{KPA}(\text{Bob}, N', N)$ mentre sta parlando con A!**
- e. A \rightarrow T: passo 7 di R1 invio di $\text{KPT}(N')$
- f. T \rightarrow B: passo 7 di R2 invio di $\text{KPB}(N')$

X.509 Servizio Autenticazione

Parte degli standard del direttorio CCITT X.500

- Definisce un riferimento per servizi di autenticazione
 - direttorio può memorizzare certificati di chiave pubblica degli utenti firmata dall'autorità di certificazione
- Definisce protocolli di autenticazione
- Usa crittografia chiave pubblica e firma digitale
 - algoritmi non sono nello standard; si raccomanda RSA

Procedure di Autenticazione

- X.509 include tre procedure di autenticazione:
 - One-Way Authentication
 - Two-Way Authentication
 - Three-Way Authentication
- Tutte usano crittografia chiave pubblica

One-Way Authentication

1 messaggio (A→B) fornisce

- identità di A e che A è originatore del messaggio
- messaggio è per B
- integrità del messaggio
- proposta chiave di sessione

One-Way Authentication

Timestamp: 1 messaggio (A→B) fornisce

- identità di A e che A è originatore del messaggio
- messaggio è per B
- integrità del messaggio
- proposta per chiave di sessione k
- timestamp t_A

Sia $[DA=t_A, B, PB(k)]$:

A invia a B : $cert_A, DA, Sig_A(DA)$

Timestamp: X.509 Two-Way Authentication

MUTUA AUTENTICAZIONE

- A invia a B
 $certA, DA, Sig_A(DA) \quad [DA=tA, N, B, PB(k)]$
 - B invia a A
 $certB, DB, Sig_B(DB) \quad [DB=tB, N', A, N, PA(k')]$
- $tA, tB = \text{timest.}; k, k'$ chiavi sessione prop. da A(B) uso di N' esclude attacchi chosen-text
- critica: in DA non c'è identità di A

Challenge-response: X.509 Three-Way Authentication

Mutua autenticazione senza orologi sincronizzati
(0 indica tempo, che si può aggiungere)

Tre messaggi (A→B, B→A, A→B):

1. A a B: $certA, DA, Sig_A(DA) \quad [DA=0, N, B, PB(k)]$
2. B a A: $certB, DB, Sig_B(DB) \quad [DB=0, N', A, N, PA(k')]$
3. A a B: $B, Sig_A(N, N', B)$

Nota: la firma esplicita della sfida proposta (passo 2 e 3) serve a evitare attacchi (del tipo di quello visto con Needham-Schroder)

Challenge-response:
ISO/IEC 9798-3 Mutua autenticazione

Perchè B deve firmare N che è noto ed è stato proposto da B stesso?

Considera

1. B invia a A N
2. A invia a B $\text{cert}_A(N', B, \text{Sig}_A(N, N', B))$
3. B invia a A $\text{cert}_B(N'', A, \text{Sig}_B(N'', N', A))$

Challenge-response:
ISO/IEC 9798-3 Mutua autenticazione

Perchè B deve firmare N che è noto? Considera

1. B invia a A N
2. A invia a B $\text{cert}_A(N', B, \text{Sig}_A(N, N', B))$
3. B invia a A $\text{cert}_B(N'', A, \text{Sig}_B(N'', N', A))$

Attacco

1. T (come B) invia a A : N
2. A invia a T (come B) $\text{cert}_A(N', B, \text{Sig}_A(N, N', B))$
 1. T (come A) invia a B: N'
 2. B invia a T (come A) $\text{cert}_B(N'', A, \text{Sig}_B(N'', N', A))$
3. T (come B) invia a A $\text{cert}_B(N'', A, \text{Sig}_B(N'', N', A))$

PKI: Public Key Infrastructure

- Certificati utilizzati sono emessi da una Autorità di Certificazione (CA) **FIDATA**
- CA fornisce certificati dei diversi utenti
- Quando un Utente vuole conoscere la chiave pubblica di un altro utente si rivolge a CA che la fornisce (firmandola con la propria chiave a garanzia)
- In questo modo è sufficiente conoscere una sola chiave pubblica con certezza

Nota:

- Se non c'è fiducia nella CA allora i certificati sono inutili
- Le chiavi non sono usate per sempre e sono cambiate
- La lunghezza della chiave è correlata al livello di sicurezza

Certificati X.509

Emessi da una Autorità di Certificazione (CA) contengono:

- versione: 1, 2, o 3 (ogni versione con maggiori informazioni)
- numero seriale (unico nella CA) che identifica il certificato
- identificatore algoritmo di firma
- nome emittente (CA) il certificato X.509
- periodo di validità (da - a)
- soggetto del certificato (nome possessore)
- protocollo chiave pubblica (algoritmo, chiavi, parametri)
- unico identificatore emittente (v2+)
- unico identificatore soggetto (v2+)
- campi estensione (v3)
- firma (hash di tutti i campi del certificato)

CA<<A>> denota il certificato di A firmato da CA

Certificati X.509

Signed fields	Version		
	Certificate serial number		
	Signature Algorithm Object Identifier (OID)		
	Issuer Distinguished Name (DN)		
	Validity period		
	Subject (user) Distinguished Name (DN)		
	Subject public key information	Public key Value	Algorithm Obj. ID (OID)
	Issuer unique identifier (from version 2)		
	Subject unique identifier (from version 2)		
	Extensions (from version 3)		
	Signature on the above fields		

Certificati X.509

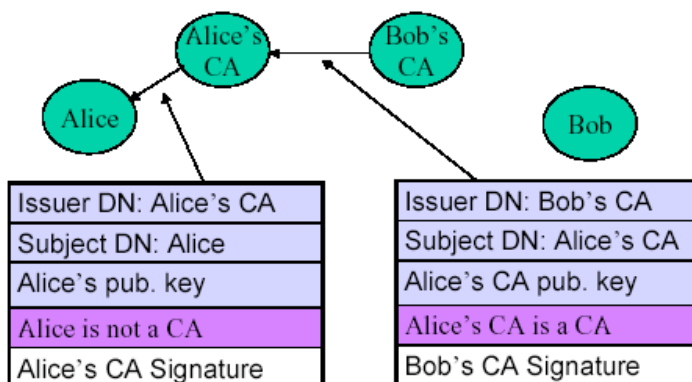
- chiunque ha accesso a CA può chiedere certificati
- solo la CA può modificare un certificato
- i certificati non si possono falsificare (RSA 2000 o più bit) e possono essere posti in direttorio pubblico
- per ricevere certificati:
 - centrale: CA fornisce chiave privata e la invia all'utente
 - distribuita: l'utente genera la sua chiave e invia richiesta di pubblicazione della chiave pubblica

Gerarchia CA

- Se due utenti condividono una CA allora possono conoscere la reciproca chiave pubblica
- Altrimenti le CA formano una gerarchia
- Ogni CA valida altre CA
- Ogni CA ha certificati per i suoi clienti e per il genitore nella gerarchia

Gerarchia CA

Come fa Bob ad avere il certificato di Alice
(se non fanno parte della stessa CA?)



Revoca dei Certificati

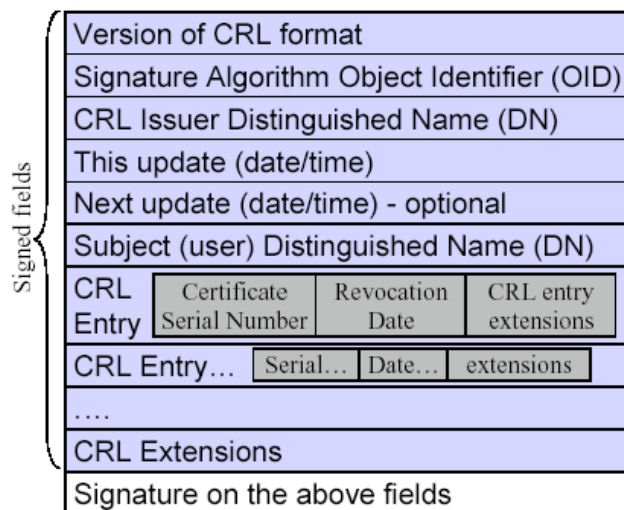
I Certificati hanno un periodo di validità e possono essere revocati prima della scadenza:

1. la chiave segreta dell'utente è compromessa
2. utente non è più certificato da CA
3. il certificato della CA è compromesso

CA mantiene CRL (lista certificati revocati)

- CRL che deve essere esaminata dagli utenti (nota: la fase di verifica se un certificato è revocato rallenta molto le prestazioni)

Revoca dei certificati



X.509 Versione 3

- I certificati devono contenere nuove informazioni:
 - email/URL, limiti utilizzo del certificato stesso
- Invece di definire nuovi campi per ciascuna informazione si definisce un metodo per definire estensioni
- Estensioni consistono di:
 - identificatore di estensione
 - informazioni relative all'estensione

PGP: Pretty Good Privacy

- Posta elettronica (Zimmerman)
- Modello di fiducia (trust model)
 - non ci sono CA fidate e ogni utente si comporta come CA e decide di chi si fida
 - certificati contengono email e Chiavi pubbliche
 - certificati sono firmati da diversi utenti
 - se ti fidi di un numero sufficiente delle firme apposte allora credi nel certificato
 - ogni utente mantiene informazioni su chiavi pubbliche di altri utenti e firme di queste chiavi e una valutazione del grado di fiducia nelle stesse

Prtocolli: Zero-knowledge

Molti protocolli rilasciano informazione

Esempio: firmi messaggio con Nonce N ,
attaccante conosce la firma di N -
informazione non nota prima

- Se il nonce viene cambiato ogni volta l'informazione non è molto utile. Però da un punto di vista teorico delle informazioni sono rilasciate

Protocolli Zero-knowledge: protocolli che non rilasciano nessuna informazione

Autentica zero-knowledge

Trovare radici mod n è difficile se n è prodotto di due primi grandi (analogo a RSA)

- Chiave pubblica: (n,v) , n prodotto di due primi grandi, v numero scelto a caso
- Chiave segreta: s radice quadrata di v mod n (non serve ricordare fattori di n)

Nota: è facile trovare chiave pubblica ma conoscendo la chiave pubblica è difficile conoscere la chiave segreta

Autentica zero-knowledge

Alice sceglie k numeri a caso: $r_1, r_2, r_3 \dots r_k$ e invia $r_i^2 \bmod n$ a Bob

Bob sceglie un sottoinsieme $S1$ dei r_i^2 e lo comunica a A; altro insieme è $S2$

Alice invia $sr_i \bmod n$ per ognuno dei r_i^2 di $S1$ e invia r_i per ognuno dei r_i^2 di $S2$

Bob

- per ogni numero in $S1$: esegue quadrato e verifica che il quadrato sia $vr_i^2 \bmod n$;
- per ogni numero in $S2$: esegue il quadrato di r_i e verifica

Zero knowledge: verifica

Protocollo più lento di RSA ma comunque efficace

Perché funziona:

- Alice mostra di conoscere la chiave segreta s per i numeri in $S1$
- Se T si autentica al posto A può scegliere i numeri ma conosce $\langle r_i, r_i^2 \rangle$ solo per quei numeri in $S2$ ma non sa come rispondere ai numeri in $S1$ - scelti da Bob;
- Per ciascuno di questi ha prob 50% di sbagliare (se k è grande allora prob è vicina a 1)

Zero knowledge: verifica

- Se T si autentica per A conosce $\langle r_i, r_i^2 \rangle$ solo per quei numeri in S2 ma non sa come rispondere ai numeri in S1 - scelti da Bob
- Se T utilizza info ottenute da altre autentiche allora conosce per ciascuno r_i^2 o r_i oppure br_i^2 ma non entrambi.
- Quindi se T si autentica con Bob per insiemi in S2 conosce $\langle r_i^2, sr_i \rangle$ ma non r_i
- Zero knowledge: alla fine del protocollo Bob conosce solo coppie $\langle x, x^2 \rangle$ che può calcolare da solo

Esercizio: Protocollo NS

Il seguente protocollo Needham-Schroeder è più semplice di quello NS esteso:

- A genera N' e invia a C $\langle N', \text{voglio parlare con B} \rangle$
- C genera K e manda a A $KAC(N', K, B, KBC(K, A))$
- A decodifica, verifica N' e identità B e manda a B $KBC(K, A)$ e $K(N')$
- B decodifica, verifica identità A crea nonce N'' e invia a A $K(N''-1, N''')$
- A invia a B $K(N''-1)$

ESERC: Assumi che A cambi KAC (derivata dalla password) e un attaccante conosca la vecchia chiave KAC. Se B non sa del cambio un attaccante potrebbe autenticarsi come A con B. 1. COME SI PUO' FARE? 2. RIMEDI?

Esercizio 1: Protocollo NS, esteso,

Soluzione per 2: nel protocollo NS - esteso B definisce un nonce N e chiede a A di fornirgli la codifica di N con la sua chiave; N viene poi fornito a B da C attraverso A

- A dice a B : voglio parlare con te
- B genera N (nonce) e invia a A $KBC(N)$
- A genera N' e invia a C $\langle N', KBC(N) \rangle$
- C genera K e manda a A $KAC(N', K, B, KBC(K, A, N))$
- A decodifica, verifica N' e identità B e manda a B $KBC(K, A, N)$ e $K(N'')$
- B decodifica, verifica identità A crea nonce N''' e invia a A $K(N''-1, N''')$
- A invia a B $K(N'''-1)$

Esercizio 2: attacco prot. Woo-Lam

Autenticazione con autorità centrale

Mostrare che il seguente protocollo (proposto da Woo e Lam) non funziona (sugg. T inizia due sessioni con B - alla fine si autentica con B come A)

1. A invia a B A
2. B invia a A N
3. A invia a B $KAC(N)$
4. B invia a C $KBC(A, KAC(N))$
5. C invia a B : $KBC(N)$
6. Bob verifica se il nonce è quello generato da lui

Soluzione: Attacco prot. Woo-Lam

T inizia due sessioni con B: una come A e una come T

1. T(come A) invia a B A
2. T invia a B T
3. B invia a T (come A) N
4. B invia a T N'

T deve indurre C a comunicare a B KBC(N)

Soluzione: Attacco prot. Woo-Lam

T inizia due sessioni con B: una come A e una come T

1. T(come A) invia a B A
2. T invia a B T
3. B invia a T (come A) N
4. B invia a T N'
5. T (come A) invia a B KTC(N) !!
6. T come A invia B KTC(N) !!!
7. B invia a C KBC(A, KTC(N))
8. B invia a C KBC(T, KTC(N))
9. C invia a B: KBC(immondizia) [messaggio senza senso]
10. C invia a B: KBC(N)
11. Bob verifica se il nonce è quello generato da lui SI!
12. Bob verifica se il nonce è quello generato da lui NO!!

Soluzione: Attacco prot. Woo-Lam

T inizia due sessioni con B: una come A e una come T

1. T (come A) invia a B A
 2. T invia a B T
 3. B invia a T (come A) N
 4. B invia a T N'
 5. T (come A) invia a B $KTC(N)$!!
 6. T come A invia a B $KTC(N)$!!!
 7. B invia a C $KBC(A, KTC(N))$
 8. B invia a C $KBC(T, KTC(N))$
 9. C invia a B: $KBC(\text{immondizia})$ [messaggio senza senso]
 10. C invia a B: $KBC(A, N)$
 11. Bob verifica se il nonce è quello generato da lui **PER A NO!**
- N.B.: questa modifica è comunque attaccabile in altro modo (sugg. 1 sessione in cui T si finge A e si sostituisce a C)

Soluzione: includere identità nel mess. 5 e verifica a 6

5. C invia a B $KBC(A, N)$

6. B verifica nonce e ident. quando decodifica

Attacco prot. Woo-Lam modificato

T inizia una sessione con B in cui sostituisce A e C

1. T (come A) invia a B A
2. B invia a T (come A) N
3. T (come A) invia a B N !! (assume che $KTC(N) = N$)
4. B invia a T (come C) $KBC(A, B, N)$
5. T (come C) invia a B: $KBC(A, B, N)$
6. Bob verifica se il nonce è quello generato da lui **SI!**

Attacco prot. Woo-Lam modificato

T inizia una sessione con B in cui sostituisce A e C

1. T invia a B C
2. A invia a T N
3. T invia a A $KTC(N)$
4. A invia a T (come C) $KAC(C, KTC(N))$

5. A invia a T (come C) $KAC(C, KTC(N))$
6. Bob verifica se il nonce è quello generato da lui [SI!](#)

Esercizio 3: Autenticazione & Firma

Dato uno schema di identificazione basato su sfida (challenge-response)

si può ottenere un metodo di firma digitale nel seguente modo:

Soluzione: rimpiazza il nonce N con $\text{hash}(x||M)$

[x è testimone, M messaggio e hash è funzione hash one-way]

Esercizio 4: Autenticazione & Firma

Dato il seguente protocollo di autenticazione

- A → B: sono A
- B → A: R1 (nonce)
- A → B: $K_{S_a}(R1)$ (invia la codifica di R1 con la chiave segreta di A)

Se R1 è scelto opportunamente il protocollo è sicuro

MA

Se si utilizza la stessa chiave sia per autenticare che per firmare messaggi allora il protocollo permette di falsificare la firma. Come?