Dipartimento
Elettronica
Informazione

Mobilità e sicurezza delle reti

Lezione 2

Protocolli di sicurezza

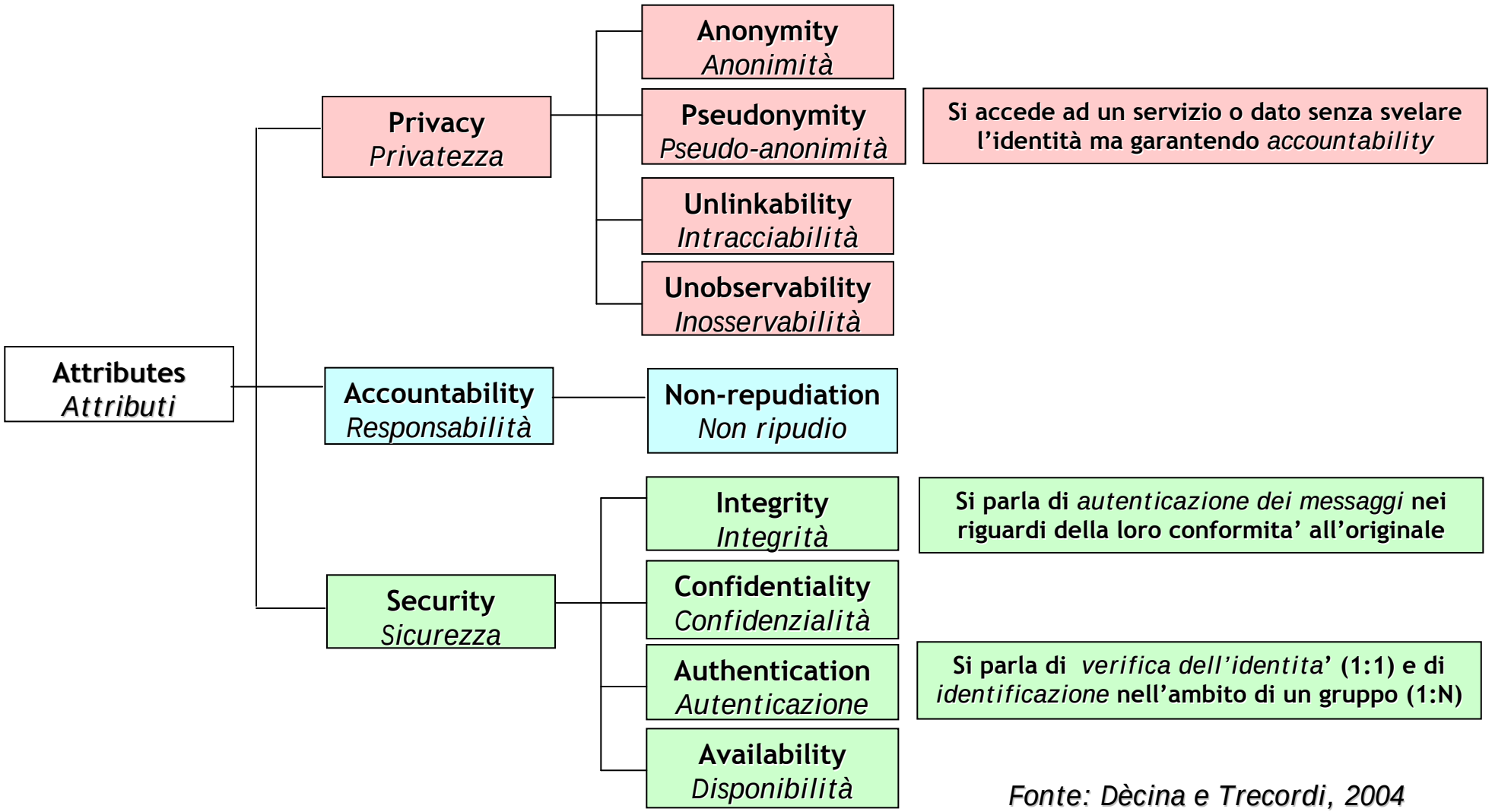
Maurizio Dècina
Politecnico di Milano

maurizio.decina@polimi.it
http://www.ictc.it/decina

Servizi di sicurezza

- Confidenzialità dei messaggi, *segretezza, riservatezza (protezione della proprietà intellettuale di un messaggio)*
  - Crittografia a chiave privata (simmetrica)
  - Crittografia a chiave pubblica (asimmetrica)
- Autenticazione di entità, *identificazione della sorgente e autenticazione del tempo (la sorgente è viva a un dato istante)*
  - Firme digitali
  - Certificati digitali
- Autenticazione dei messaggi, *integrità: identificazione della sorgente e autenticazione del contenuto*
  - MAC
  - Hash
  - Firme digitali
- Controllo degli accessi, *identificazione, autorizzazione, amministrazione*
  - Protocolli di autenticazione: User ID/Password, firma di un codice hash
  - Kerberos
- Affidabilità, *guasti e riparazioni, reliability/availability*
  - Business Continuity, sistemi ridondanti
- Disponibilità, *dependability,* 'costanza'
  - DoS
  - Virus
- Non ripudio, autenticazione delle transazioni*: identificazione sorgente e autenticazione di contenuto e tempo/unicità*
  - Firme digitali e audit log
- Privatezza, delle transazioni di un'entità, *anonimità*

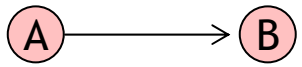
Sicurezza, privatezza e responsabilità



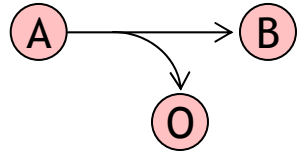
Fonte: Dècina e Trecordi, 2004

Tipi di attacchi alla sicurezza

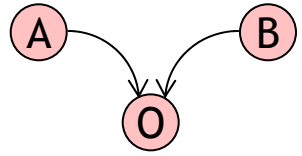
- **Cryptoanalist Attack,** attacco del crittoanalista
  - ciphertext only, known plaintext, chosen plaintext, chosen ciphertext
- **Reply Attack**, attacco della replica
- **Man-in-the-middle Attack**, attacco dell'uomo nel mezzo
- **Type-flaw Attack**, attacco dell'errore di battuta
- **Masquerading Attack**, attacco del travestimento
  - **Binding Attack**, attacco dell'aggancio
- **Denial of Service Attack**, attacco del diniego di servizio
- **Virus Attack**, attacco con virus



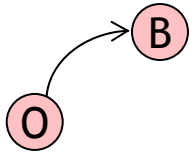
- Intercettare
  - Eavesdropping
  - Reply



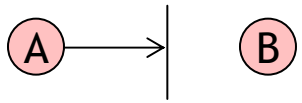
- Modificare
  - Man-in-the-middle
  - Type-flaw



- Fabbricare/Forgiare
  - Masquerading
  - Binding



- Interrompere
  - DoS
  - Virus



Meccanismi di sicurezza

- Prevenzione: politiche, procedure, risk assessment, vulnerability assesment, progetto di reti sicure con meccanismi di protezione e rivelazione, rafforzamento dei sistemi informativi (hardening), audit, ...
- Protezione
  - Encryption
  - Firewall
  - Nat/Pat
  - Virtual Private Network, Tunneling
  - Access Control
  - Honeypot
  - Antivirus, ...
- Rivelazione
  - Vulnerability Assessment, Penetration Test
  - Intrusion Detection Systems, Intrusion Prevention Systems
  - Monitoring, ...
- Reazione: emergency response, intelligence, patch, restore, audit,..

Simbolismo

- *Messaggi*: M o P, in chiaro; C, in cifrato
- *Identità di Entità/Persone*: A, Alice; B, Bob; TA, Thrusted Authority
- *Chiave segreta condivisa tra A e B* (privata): K_{AB}
- *Crittografia classica*: $C=E_{K_{AB}}(P)$; $P=D_{K_{AB}}(C)$; tali che $P=D_{K_{AB}}[E_{K_{AB}}(P)]$;
  $C=\{P\}_{K_{AB}}$; $P=\{C\}_{K_{AB}}$
- *Chiave pubblica di A* (crittografia a chiave pubblica): K_A
- *Chiave privata di A* (crittografia a chiave pubblica): K^{-1}_A
- *Crittografia a chiave pubblica:*
  da A a B $C=E_{K_B}(P)$; $P=D_{K_B^{-1}}(C)$; $C=\{P\}_{K_B}$; $P=\{C\}_{K_B^{-1}}$
  oppure, firma di A verso B
  $C=E_{K^{-1}_A}(P)$; $P=D_{K_A}(C)$; $C=\{P\}_{K^{-1}_A}$; $P=\{C\}_{K_A}$

Simbolismo

- N_A^i, N_B^i, *nonce*, numeri casuali di *A* e di B, ecc.
- TS_A^i = *Time stamp* di *A*; TTL_A^i = *Time to Live* di *A*
- M1,M2: *concatenazione di messaggi*
- $H_M = h(M)$, *codice hash* di M; tale che $h^{-1}(H_M)$ sia molto difficile da calcolare
- $MAC_M = MAC_{K_{AB}}(M)$, *codice di autenticazione*
- *Certificato Digitale di A*: $A, K_A, \{h(A, K_A)\}_{K^{-1}_{TA}}$
- *Firma Digitale di A su M*: $M, \{H_M\}_{K^{-1}_A}$

Crittografia a chiave pubblica

- Chiave pubblica K_B , Chiave privata K_B^{-1}

$$\{\{P\}_{K_B}\}_{K_B^{-1}} = P = \{\{P\}_{K_B^{-1}}\}_{K_B}$$

- 1. Alice e Bob sono d'accordo su un crittosistema (es. RSA)
- 2. Bob manda a Alice la sua chiave pubblica K_B .(Oppure la chiave K_B è immagazzinata in un DataBase, opure è in appendice a un messaggio, ...)
- 3. Alice cifra il messaggio con la chiave pubblica di Bob K_B e lo manda a Bob
- 4. Bob decifra il messaggio con la sua chiave privata K_B^{-1}

$$C=\{P\}_{K_B} \; ; \quad P=\{C\}_{K_B^{-1}}$$

- Chiunque può cifrare un messaggio con la chiave pubblica K_B che a sua volta può essere decifrato soltanto da B, possessore della chiave privata K_B^{-1}
- A, possessore della chiave privata K_A^{-1}, può anche cifrare con la chiave privata i messaggi (vedi firma digitale) che possono essere letti in chiaro da chiunque usando la chiave pubblica K_A
- La crittografia pubblica risolve il problema della gestione delle chiavi: soltanto due chiavi per entità (per agente)
- La crittografia a chiave pubblica può essere usata per segretare e distribuire chiavi di sessione che a loro volta possono servire per crittografare con algoritmi simmetrici le comunicazioni

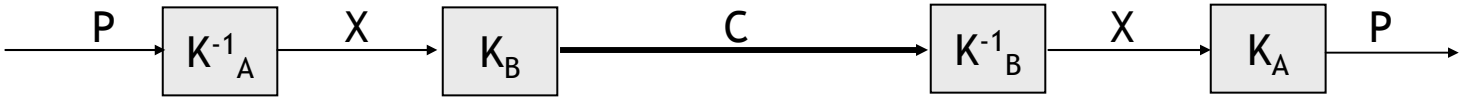
Simbolismo

- *Crittografia a chiave pubblica, da A a B*
  - Confidenzialità:

    $C=E_{K_B}(P)$; $P=D_{K^{-1}_B}(C)$; $C=\{P\}_{K_B}$; $P=\{C\}_{K^{-1}_B}$,
  - Firma digitale di *A*:

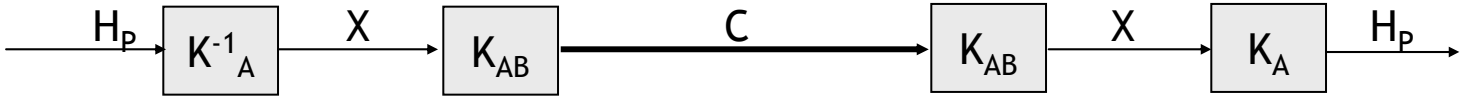
    $C=E_{K^{-1}_A}(H_P)$; $H_P=D_{K_A}(C)$; $C=\{H_P\}_{K^{-1}_A}$; $H_P=\{C\}_{K_A}$
  - Firma di *A* e segretezza:

    $C=\{\{P\}_{K^{-1}_A}\}_{K_B}$; $P=\{\{C\}_{K^{-1}_B}\}_{K_A}$



- *Confidenzialità, identificazione e integrità, da A a B*

  $C=\{\{H_P\}_{K^{-1}_A}\}_{K_{AB}}$; $H_P=\{\{C\}_{K_{AB}}\}_{K_A}$

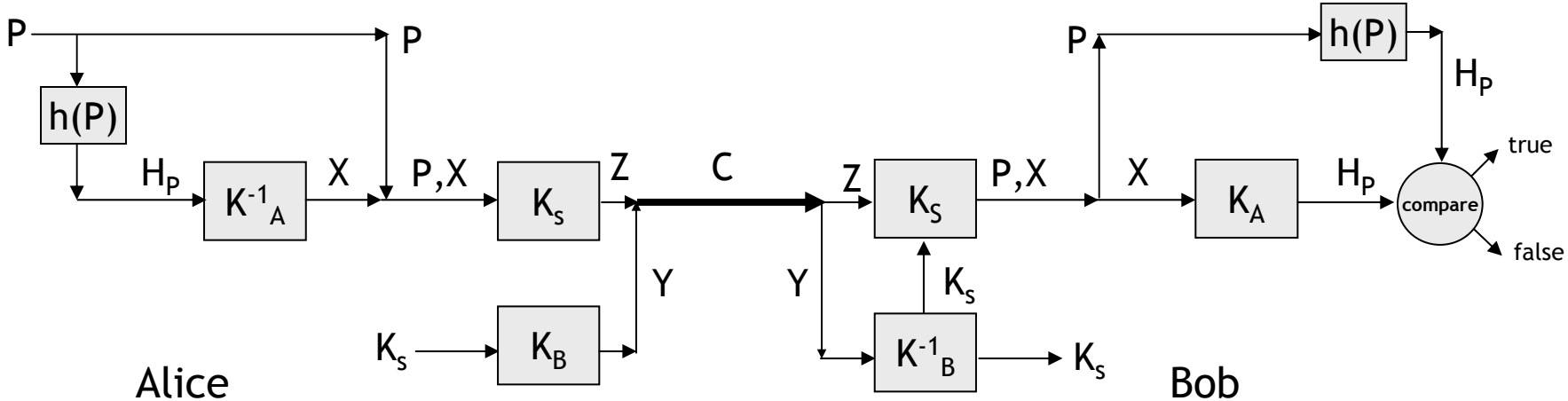


Come si fa in pratica

- *Confidenzialità, identificazione, integrità e scambio di chiave segreta , da A a B*

$C = Z,Y$ *ove:* $Y = \{K_s\}K_B$; $Z = \{P,X\}K_s$; $X = \{H_P\}K^{-1}_A$ e $H_P = h(P)$

$C = \{P,\{H_P\}K^{-1}_A\}K_s, \{K_s\}K_B$



Alice

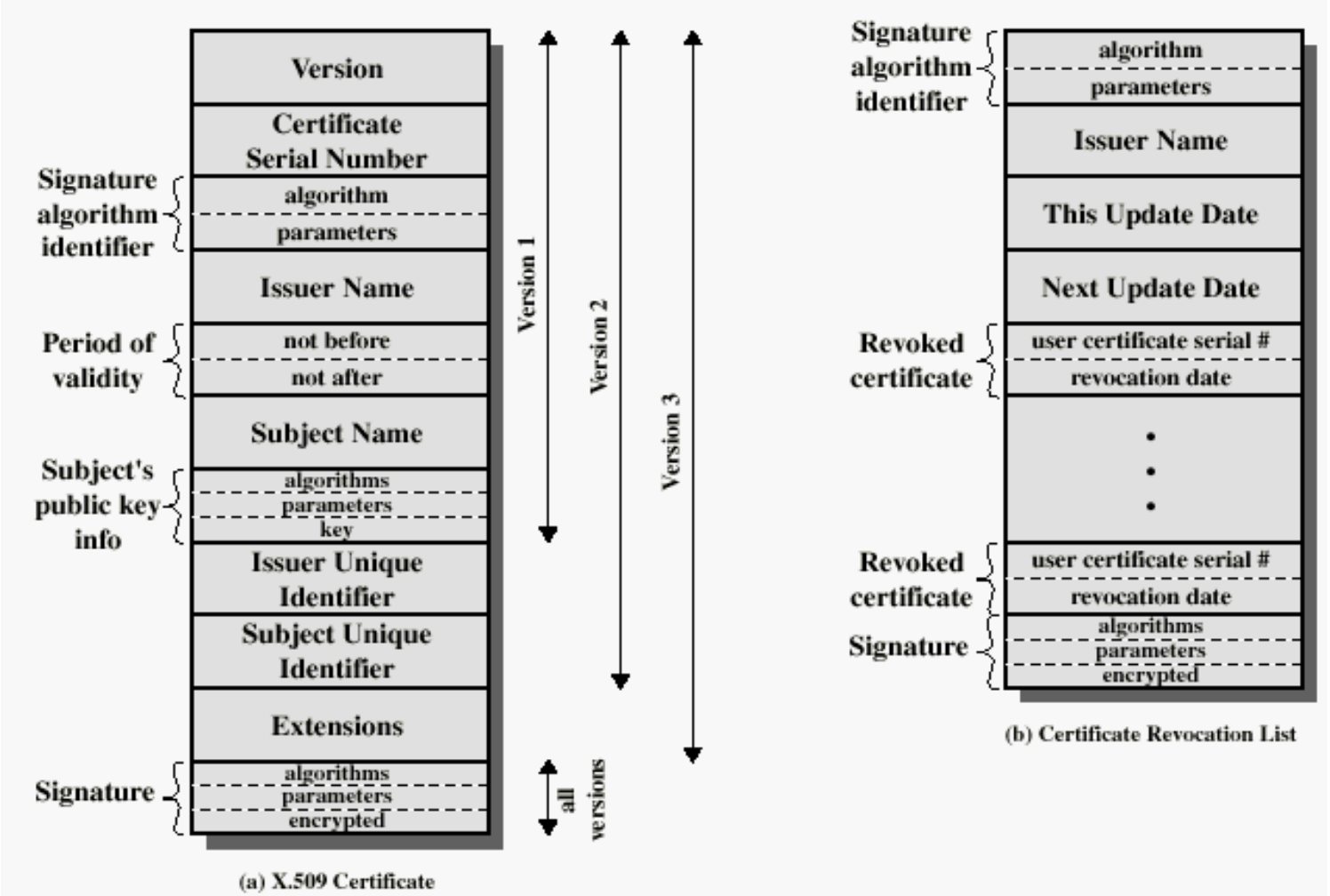
K^{-1}_A, K_A,

K_s= chiave di sessione,
generata casualmente da Alice
per questa sessione con Bob

Bob

K^{-1}_B, K_B

Certificati X.509



Autenticazione

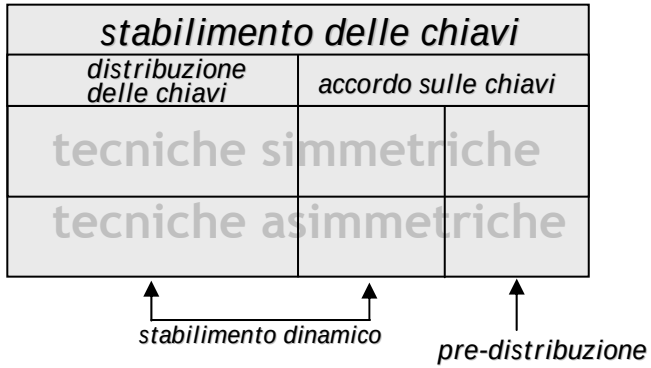
Autenticazione mutua o unilaterale

Proprietà	Identificazione della sorgente	Integrità dei dati	temporalità, unicità
Tipo di autenticazione			
autenticazione di messaggio	si	si	_
autenticazione di entità	si	_	si
autenticazione di transazione	si	si	si
autenticazione di chiave	si	si	desiderabile

Termine di autenticazione	significato
autenticazione	**dipende dal contesto d'uso**
autenticazione di entità	**identità di un'entità e sua esistenza a un certo istante**
autenticazione origine dati	**identità dell'origine dei dati**
autenticazione implicita chiave	**identità di un'entità che potrebbe condividere la chiave**
autenticazione conferma chiave	**evidenza che una chiave è posseduta da una certa entità**
autenticazione esplicita chiave	**evidenza che un'entità identificata possiede una certa chiave**

Stabilimento delle Chiavi

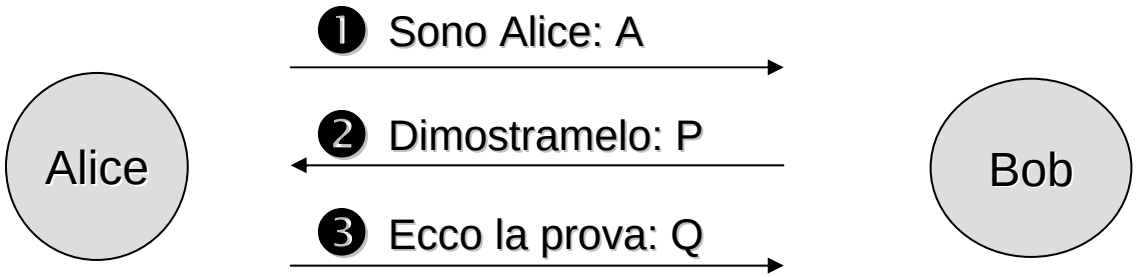
- **Key establishment**, *stabilimento delle chiavi*: un processo o un protocollo per cui un segreto condiviso diventa disponibile a due o piu' entità, al fine di successivo impiego crittografico
  - **Key distribution protocol**, *protocollo o meccanismo di distribuzione delle chiavi*, è una tecnica di stabilimento delle chiavi per cui un'entità crea o ottiene un valore segreto e lo trasferisce in modo sicuro ad altri
  - **Key agreement protocol**, *protocollo o meccanismo di accordo sulle chiavi*, è una tecnica di stabilimento delle chiavi per cui un segreto condiviso è derivato da due o piu' entità come una funzione delle informazioni scambiate da ciascuno di loro, o associate a ciascuno di loro, in modo tale che nessuno dei partecipanti possa predeterminare il valore segreto che ne risulta
  - I protocolli di accordo sulle chiavi comportano funzioni di autenticazione e richiedono tipicamente una **fase di set-up** in cui viene distribuito materiale segreto per le chiavi iniziali
- **Key pre-distribution**, *pre-distribuzione delle chiavi*, in questa modalità le chiavi si stabiliscono in modo predeterminato in base al materiale iniziale segreto fornito sulle chiavi
- **Dynamic key establishment** o *session key establishment, stabilimento dinamico delle chiavi o stabilimento delle chiavi di sessione*, le chiavi stabilite da un gruppo fisso di persone, due o piu', variano durante le interazioni successive



Protocolli Autenticati di Stabilimento delle Chiavi

- Key establishment protocol: stabilire un segreto condiviso
- Authentication protocol: fornire a un partecipante (agent, partner) garanzia sull'identità dell'interlocutore
  - Authenticated key establishment protocol: stabilire un segreto condiviso avendo avuto prova dell'identità dell'interlocutore
- Key establishment protocols: il segreto condiviso è la chiave di sessione
  - La chiave di sessione è un segreto 'effimero', nel senso che dura un breve periodo di tempo (ad esempio, pochi minuti), dopo il quale ogni traccia della chiave stessa è eliminata
- Motivationi per l'uso di chiavi effimere
  - limitare il tempo a disposizione del crittoanalista per decifrare la chiave di sessione
  - Limitare la durata dell'esposizione in chiaro, nel caso di attacco con successo del crittoanalista
  - Evitare la necessità di immagazzinare svariate chiavi di sessione: basta chiedere la chiave effimera solo quando serve per una determinata sessione
  - Creare indipendenza tra le varie sessioni di comunicazione cifrata

Semplice sfida e risposta



M1. A →B : A

M2. B →A : P

M3. A →B : Q

P, numero casuale

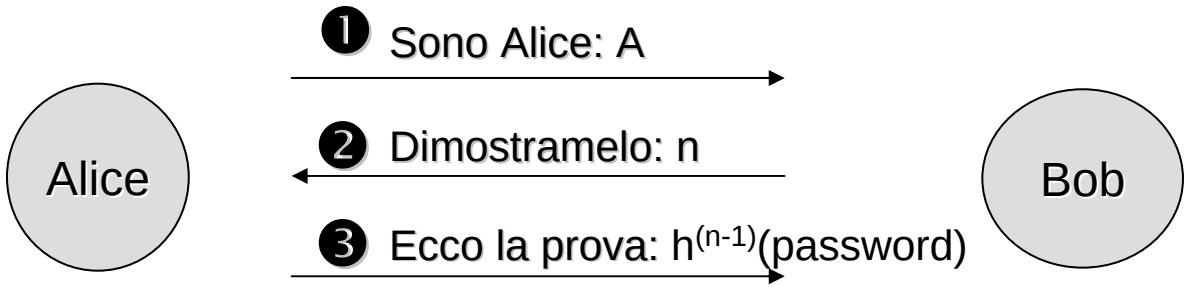
$Q = E_K(P)$ = Segreto Condiviso (P)

ad es.: $Q = P^b \bmod m$

K=(b, m)
$[E_K(P), K]$ = segreto condiviso da A e B

Bob autentica Alice, non viceversa

One Time Password hash di Lamport



M1. A →B : A

M2. B →A : n

M3. A →B : $h^{(n-1)}$(password)

Bob effettua h($h^{(n-1)}$(password)) e lo confronta col valore memorizzato

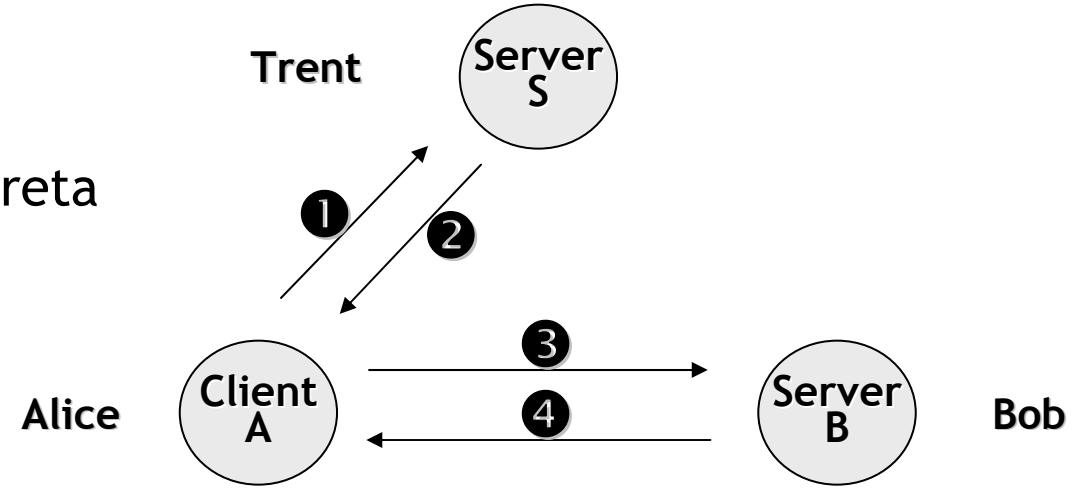
Bob (il server di autenticazione) memorizza per ogni utente:

1. Il nome utente;
2. Un intero *n*;
3. La quantità $h^{(n)}$(password) = h(h(…h(password) …)), n volte

la funzione hash **non è invertibile** e la catena di password **è percorribile in un solo senso** $h^{(1)}$-->$h^{(2)}$- >…-->$h^{(n)}$

Simplified Kerberos
Secret Key, Authenticated Key Establishment

a chiave segreta



M1. $A \rightarrow S$: $\{A, B\}K_{AS}$

M2. $S \rightarrow A$: $\{B, K_{AB}, T_S, L, \{A, K_{AB}, T_S, L\}K_{BS}\}K_{AS}$

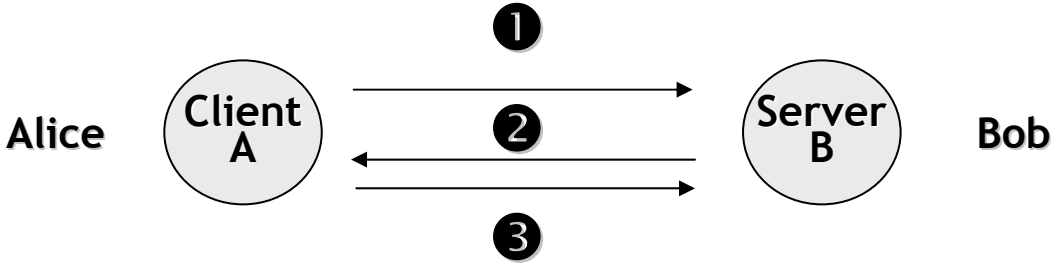
M3. $A \rightarrow B$: $\{A, K_{AB}, T_S, L\}K_{BS}, \{A, T_A\}K_{AB}$

M4. $B \rightarrow A$: $\{T_A+1\}K_{AB}$

Alternativo

M2. $S \rightarrow A$: $\{B, K_{AB}, T_S, L\}K_{AS}, \{A, K_{AB}, T_S, L\}K_{BS}$

Needham-Schroeder, Public-Key Authentication (Identification) Protocol



M1. $A \rightarrow B : \{N_A, A\}K_B$

M2. $B \rightarrow A : \{N_A, N_B\}K_A$

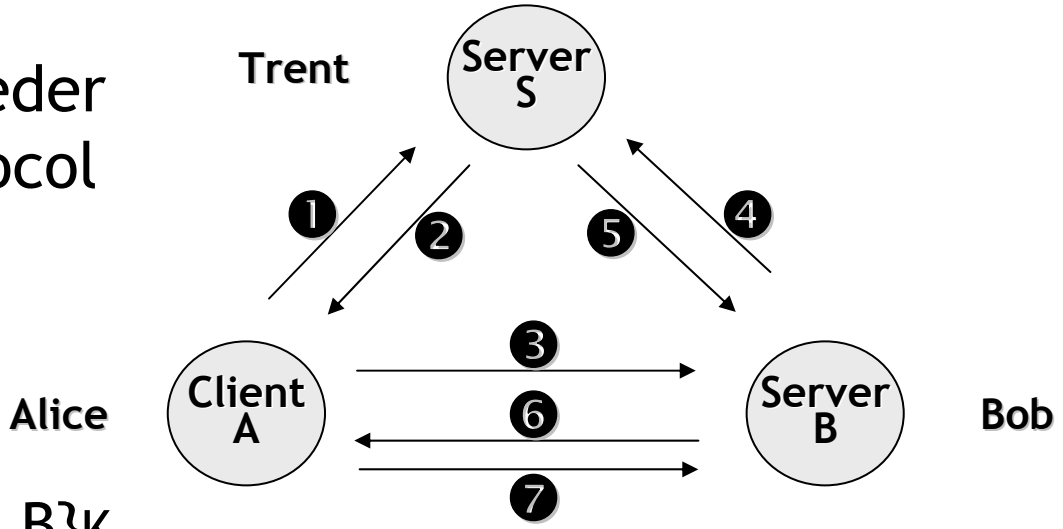
M3. $A \rightarrow B : \{N_B\}K_B$

Nonce(*): N_A and N_B
Only B can recover N_A
Only A can recover N_B

(*) number used only **once**

Needham-Schroeder Public-Key Protocol NSPK, 1978

Needham-Schroeder Public-Key Protocol NSPK, 1978



M1. $A \rightarrow S : \{A,B\}K_S$

M2. $S \rightarrow A : \{K_B,B\}K^{-1}_S$

M3. $A \rightarrow B : \{N_A,A\}K_B$

M4. $B \rightarrow S : \{B,A\}K_S$

M5. $S \rightarrow B : \{K_A,A\}K^{-1}_S$

M6. $B \rightarrow A : \{N_A,N_B\}K_A$

M7. $A \rightarrow B : \{N_B\}K_B$

- (M 1, 2, 4, 5): A and B obtain each other's public keys.
- (M 3, 6, 7): A and B use keys to communicate secret nonces N_A and N_B.
- Nonce: shared secret.
- Only B can recover N_A; only A can recover N_B.

Reply Attack to Full NSPK

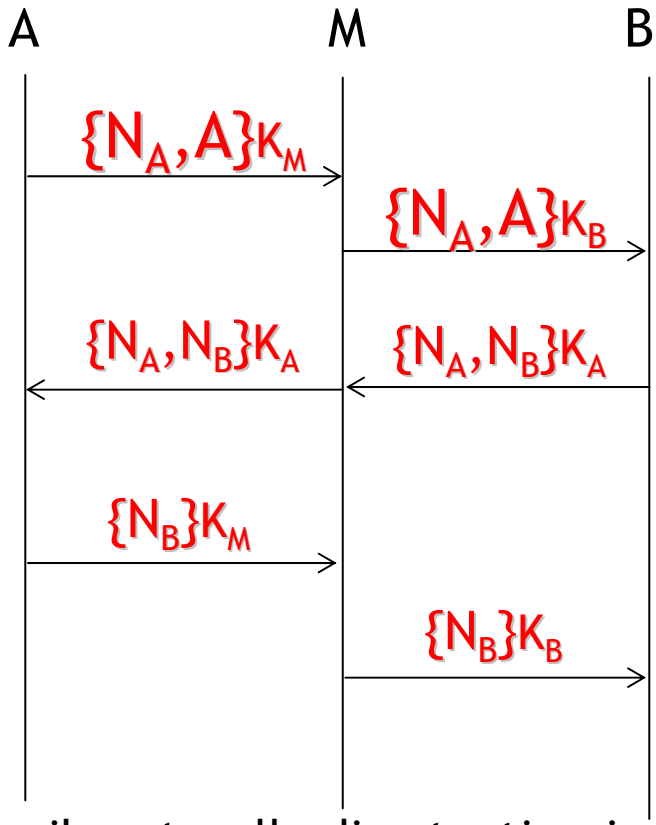
- Replay (o freshness) attack: attacco della replica, intercettare, registrare, e reintrodurre più tardi un messaggio o una sua parte
- NSPK: non c'è garanzia che le chiavi pubbliche K_B e K_A siano fresche, e cioè che siano state create da S appositamente per questo giro di protocollo

  M2. $S \rightarrow A$: $\{K_B, B\}K^{-1}{}_S$
  M5. $S \rightarrow B$: $\{K_A, A\}K^{-1}{}_S$

  se vecchie chiavi K_A and K_B sono state intercettate e decifrate (cioè sono state individuate le chiavi private $K^{-1}{}_A$ e $K^{-1}{}_B$), allora Oscar può intercettare nuovi messaggi indirizzati a S e replicare i vecchi
  - A e B credono che le chiavi sono nuove e le usano per la sessione seguente, che anche Oscar può leggere in chiaro
- Fix: usare i timestamp

  M2. $S \rightarrow A$: $\{K_B, B, T_{S1}\}K^{-1}{}_S$
  M5. $S \rightarrow B$: $\{K_A, A, T_{s2}\}K^{-1}{}_S$
- Purtroppo questo approccio richiede che tutti I server e gli host ricevano una temporizzazione sicura e accurata, in altre parole che gli orologi di ciascun sistema siano sincroni

Lowe's Man-In-The-Middle Attack to NSPK, 1996

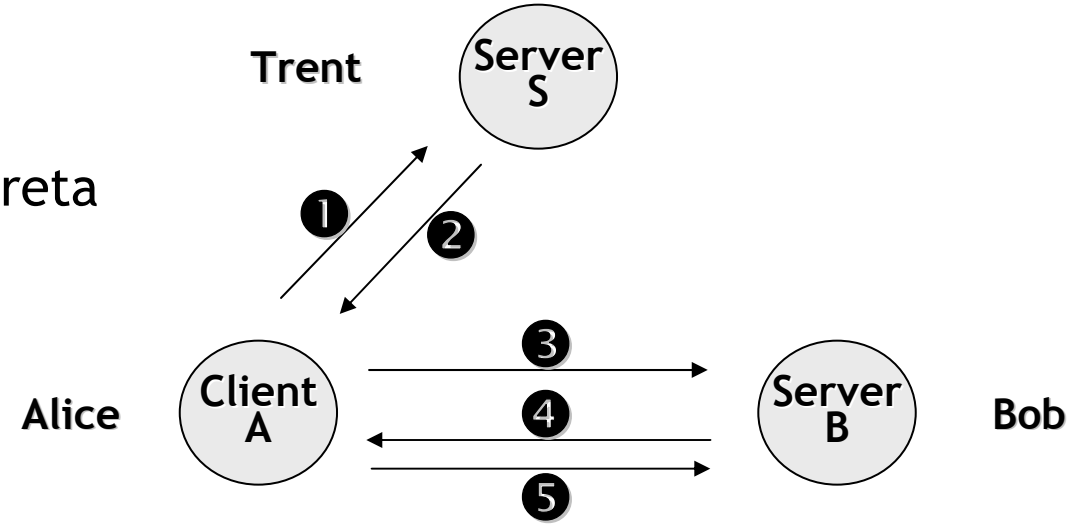


A esegue il protocollo di autenticazione con M.
M si spaccia per A nei confronti di B.
FIX: B deve includere la propria identità in M2.

M2. $B \rightarrow A$: $\{N_A, N_B, B\}K_A$

Secret Key, Authenticated Session Key Establishment

a chiave segreta



M1. $A \rightarrow S : \quad \{A,B,N_A\}K_{AS}$

M2. $S \rightarrow A : \quad \{N_A,B,K_{AB},\{K_{AB},A\}K_{BS}\}K_{AS}$

M3. $A \rightarrow B : \quad \{K_{AB},A\}K_{BS}$

M4. $B \rightarrow A : \quad \{N_B\}K_{AB}$

M5. $A \rightarrow B : \quad \{N_B+1\}K_{AB}$

Needham-Schroeder Shared Key Protocol, NSSK

Analizzare questo protocollo, studiare l'attacco di reply e fornire il 'fix'

M1. $A \Rightarrow S$: $\{A,B,N_1\}K_{AS}$

M2. $S \Rightarrow A$: $\{N_1,B,K_{AB}, \{K_{AB},A\}K_{BS}\}K_{AS}$

M3. $A \Rightarrow B$: $\{K_{AB},A\}K_{BS}$

M4. $B \Rightarrow A$: $\{N_2\}K_{AB}$

M5. $A \Rightarrow B$: $\{N_2\text{-}1\}K_{AB}$

Replay attack: non c'è garanzia che M3. sia stato creato da S proprio per questa istanza del protocollo! Una chiave precedentemente distribuita K_{0AB} è compromessa e nota all'attaccante M, che ha spiato il collegamento, ha intercettato precedenti messaggi e ha individuato la chiave

M3'. $A \Rightarrow B$: $\{K_{0AB},A\}K_{BS}$

M puo' imbrogliare B e fargli accettare la chiave K_{0AB} come se fosse 'nuova'

M3. $M(A) \Rightarrow B$: $\{K_{0AB},A\}K_{BS}$

M4. $B \Rightarrow M(A)$: $\{N_2\}K_{0AB}$

M5. $M(A) \Rightarrow B$: $\{N_2\text{-}1\}K_{0AB}$

FIX: aggiungere i timestamp nei messaggi rilevanti (o aggiungere un 'handshake' supplementare all'inizio del prtocollo)

The Otway-Rees protocol

M1. $A \Rightarrow B$: $I,A,B, \{N_A,I,A,B\}_{KAS}$

M2. $B \Rightarrow S$: $I,A,B, \{N_A,I,A,B\}_{KAS}, \{N_B,I,A,B\}_{KBS}$

M3. $S \Rightarrow B$: $I, \{N_A,K_{AB}\}_{KAS}, \{N_B,K_{AB}\}_{KBS}$

M4. $B \Rightarrow A$: $I, \{N_A,K_{AB}\}_{KAS}$

- Le chiavi simmetriche K_{AS} e K_{BS} sono predistribuite
- I è un numero casuale che identifica il protocollo di mutua autenticazione e di distribuzione della chiave di sessione simmetrica
- A e B si convincono mutuamente della loro identità e ricevono dal server S la chiave di sessione simmetrica K_{AB}

Type Flaw Attack

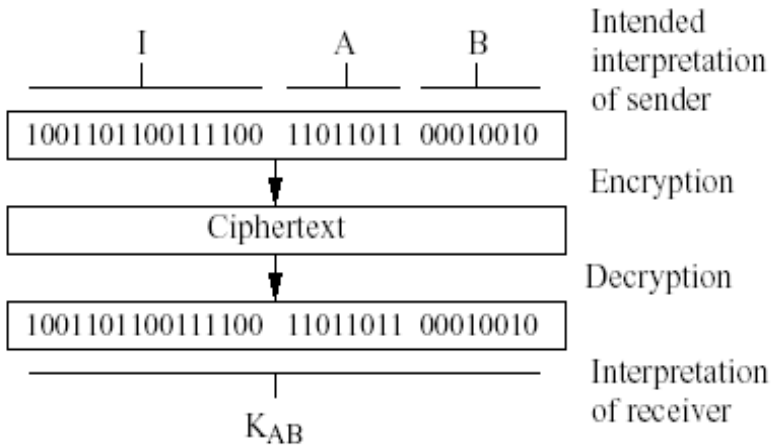
M1. $A \Rightarrow B$: $I,A,B, \{N_A,I,A,B\}_{KAS}$

M2. $B \Rightarrow M(S)$: $I,A,B, \{N_A,I,A,B\}_{KAS}, \{N_B,I, A,B\}_{KBS}$

M3. $M(S) \Rightarrow B$: $I, \{N_A,I,A,B\}_{KAS}, \{N_B,I,A,B\}_{KBS}$

M4. $B \Rightarrow A$: $I, \{N_A,I,A,B\}_{KAS}$

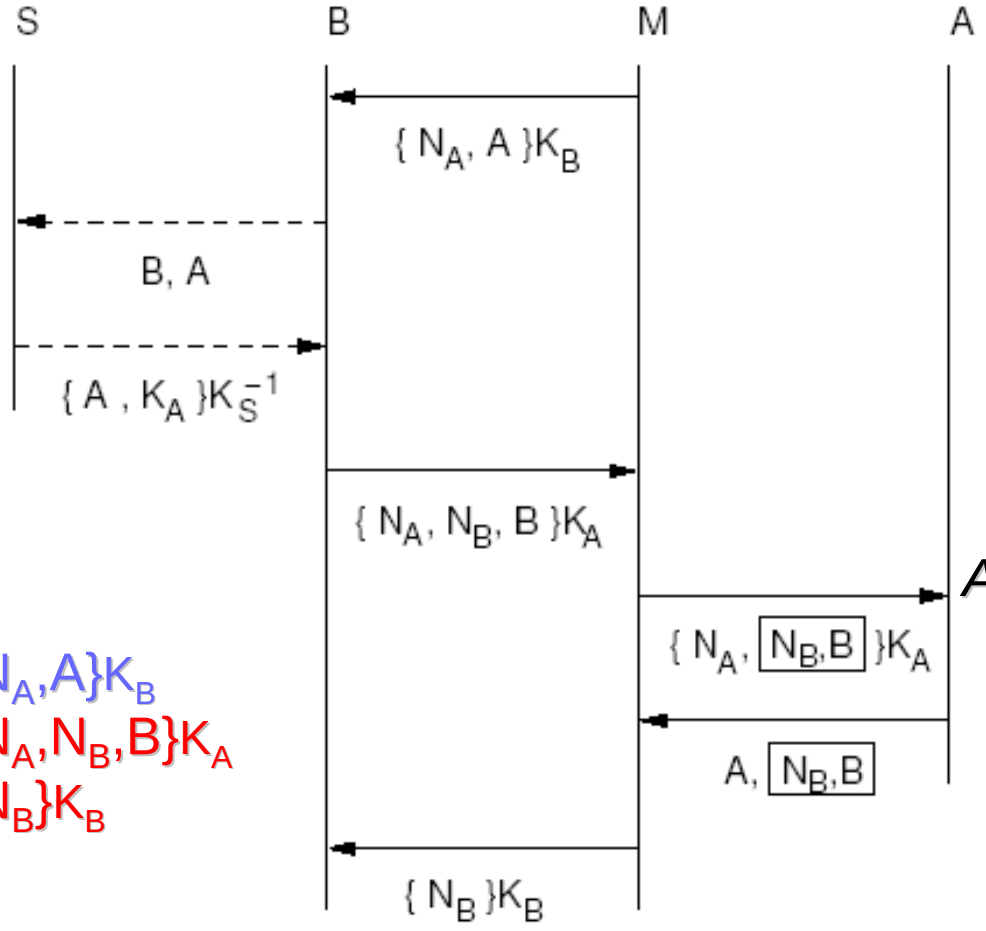
M ascolta la conversazione tra A e B usando la chiave I,A,B



- FIX: la 'battitura' deve essere 'chiara' e 'inequivocabile'

Type-Flaw Attack to full NSPK

B chiede a S la chiave pubblica di A



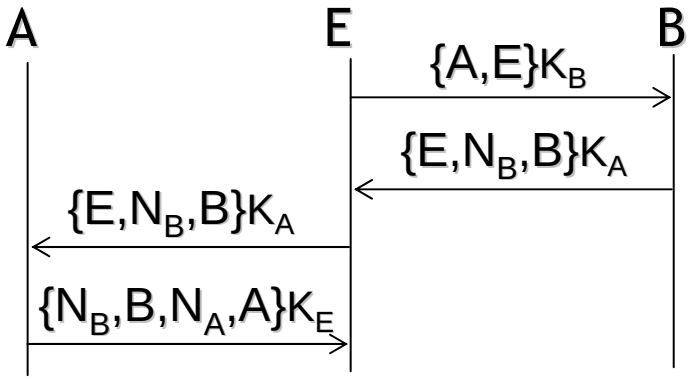
A takes N_B,B for an ID

A chiede a S la chiave pubblica di N_B,B = A,[N_B,B]

M1. A →B : $\{N_A,A\}K_B$
M2. B →A : $\{N_A,N_B,B\}K_A$
M3. A →B : $\{N_B\}K_B$

Another Type Flaw Attack to NSPK

M1. A →B : $\{A,N_A\}K_B$
M2. B →A : $\{N_A,N_B,B\}K_A$
M3. A →B : $\{N_B\}K_B$



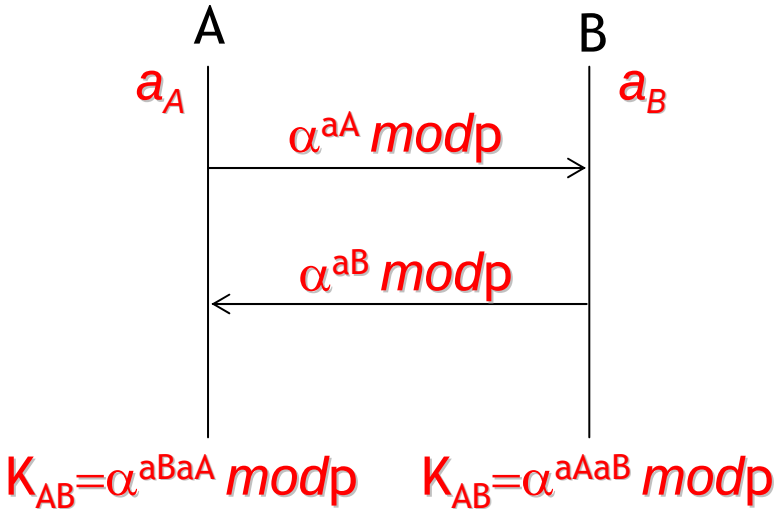
M1. E(A) ⇒ B: $\{A,E\}K_B$
(E is the intruder name, should be a nonce!)

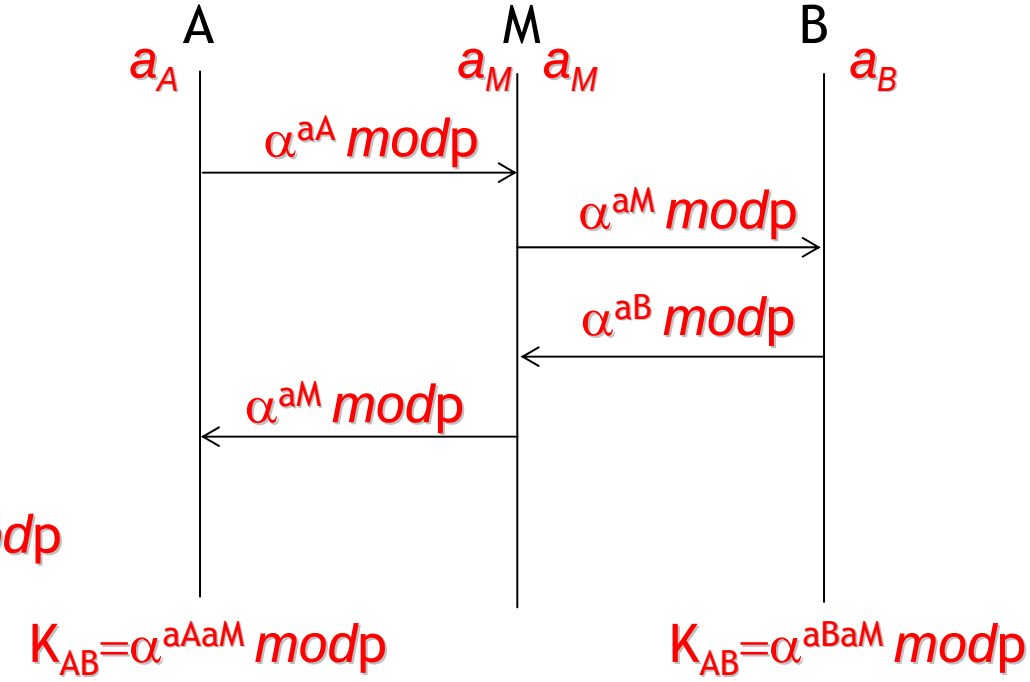
M2. B ⇒ E(A): $\{E,N_B,B\}K_A$

M1'. E(B) ⇒ A: $\{E,N_B,B\}K_A$
(here is the field confusion, A takes 'N_B,B' for a nonce!)

M2'. A ⇒ E(B): $\{N_B,B,N_A,A\}K_E$

Diffie-Hellman



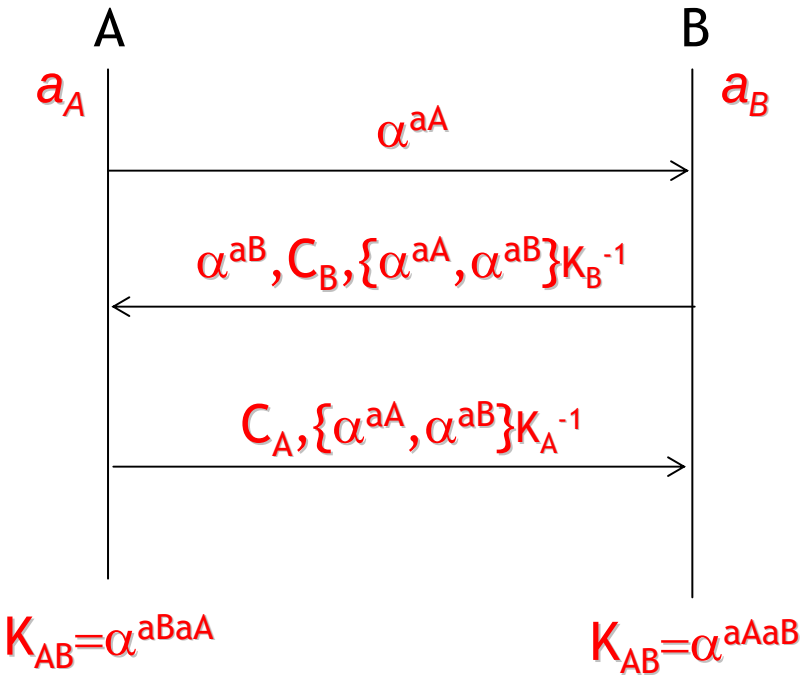


Dati: *p* primo grande,
(p-1)/2, ancora primo,
α elemento primitivo di Z_p^*, e
$0 < a_A, a_B \leq p-2$

MITM Attack

A e B parlano con Oscar, e credono di parlare tra loro!
FIX: certificati di identità

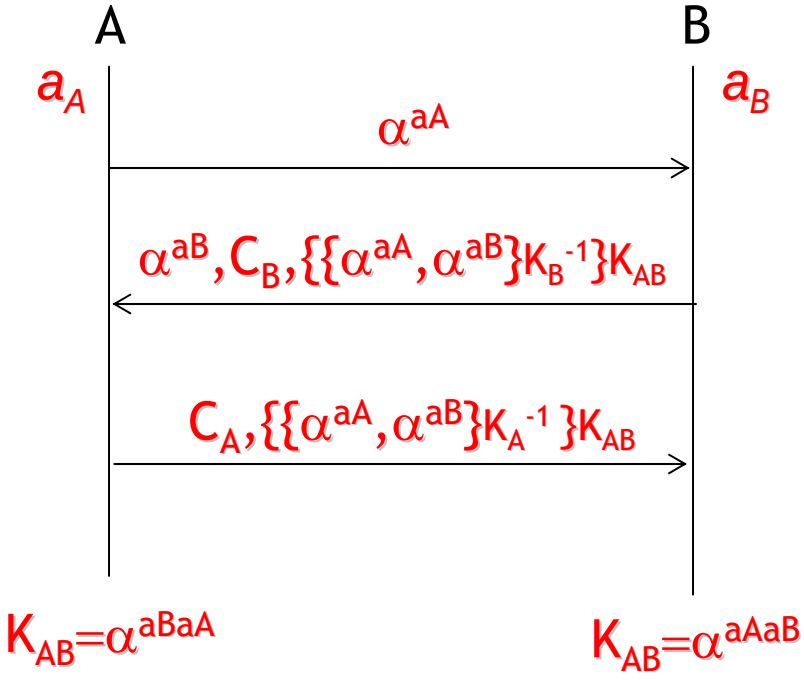
Station-to-Station Protocol



$C_B = B,\ K_B,\ \{h(B,K_B)\}K_{TA}^{-1}$

$C_A = A,\ K_A,\ \{h(A,K_A)\}K_{TA}^{-1}$

A e B mandano i loro certificati digitali
e le firme delle porzioni della chiave



Con 'conferma' di possesso mutuo
della chiave di sessione
(doppia cifratura: da evitare!)

Protocollo di Hughes – Variante DH

•Il protocollo di Hughes è una variante di Diffie-Hellman in cui Alice genera una chiave e la manda a Bob

1. Alice sceglie un intero grande a caso x e genera la chiave

$$K = \alpha^x \bmod p$$

2. Bob sceglie un intero grande a caso y e manda a Alice

$$Y = \alpha^y \bmod p$$

3. Alice manda a Bob

$$X = Y^x \bmod p$$

4. Bob calcola

$$z = y^{-1}$$

$$K' = X^z \bmod p = Y^{zx} \bmod p = \alpha^{xyz} \bmod p$$

• Se tutto è corretto: K' = K

• Vantaggio su Diffie-Hellman: K può essere calcolata prima di ogni interazione e Alice può cifrare un messaggio usando K prima di contattare Bob

Group DH

Alice, Bob e Carol
$0 < a_A, a_B, a_C \leq p-2$

M1. A⇒B: $\alpha^{aA} \; mod p = X$

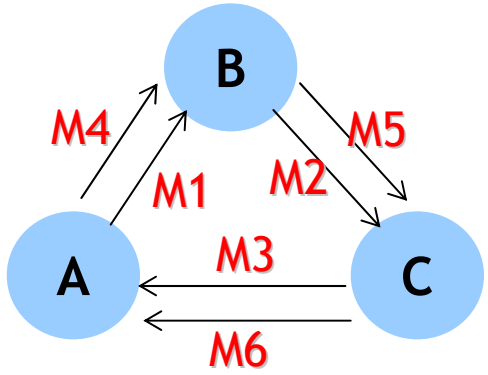
M2. B⇒C: $\alpha^{aB} \; mod p = Y$

M3. C⇒A: $\alpha^{aC} \; mod p = Z$

M4. A⇒B: $Z' = Z^{aA} \; mod p$

M5. B⇒C: $X' = X^{aB} \; mod p$

M6. C⇒A: $Y' = Y^{aC} \; mod p$

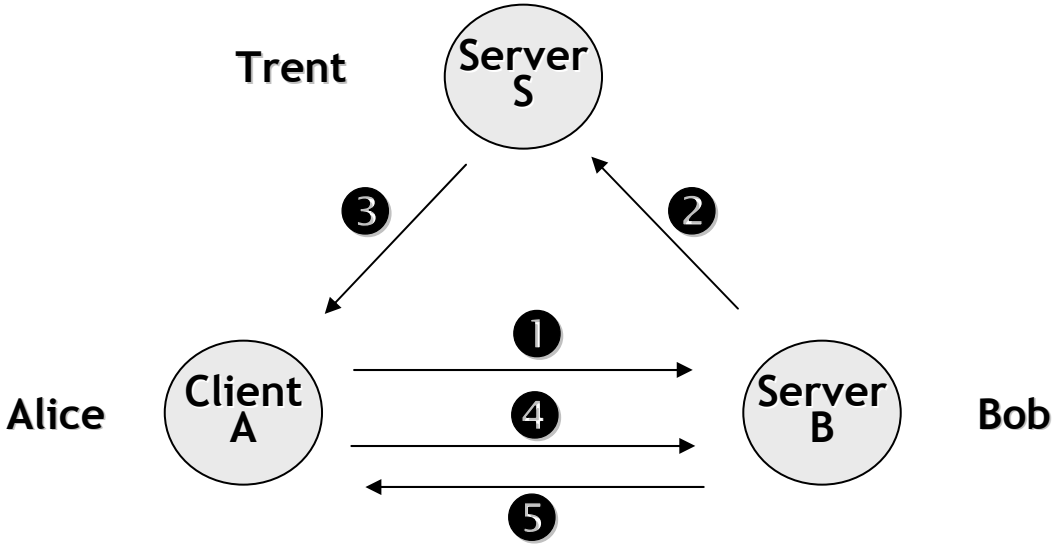


Alice calcola: $K_{ABC} = (Y')^{aA} \; mod p$
Bob calcola: $K_{ABC} = (Z')^{aB} \; mod p$
Carol calcola: $K_{ABC} = (X')^{aC} \; mod p$

Attacchi: Reply, MITM, DOS
Conferma chiave di sessione?

$K_{ABC} = \alpha^{aBaAaC} \; mod p$

Yahalom Protocol



M1. $A \rightarrow B$: A, N_A
M2. $B \rightarrow S$: $\{A, N_A, N_B\}K_{BS}$
M3. $S \rightarrow A$: $\{B, K_{AB}, N_A, N_B\}K_{AS}, \{A, K_{AB}\}K_{BS}$
M4. $A \rightarrow B$: $\{A, K_{AB}\}K_{BS}, \{N_B\}K_{AB}$
M5. $B \rightarrow A$: $\{N_B+1\}K_{AB}$

Esercizio

Che problema ha questo protocollo?

$$M1.\ A \Rightarrow B: \quad \{N_A, K_{AB}\}K_A^{-1}$$
$$M2.\ B \Rightarrow A: \quad \{N_A\}K_{AB}$$

Tutti possono decifrare M1. usando la chiave pubblica di A: K_A!

Esercizio

Analizzare questo protocollo, studiare un attacco e fornire il 'fix'

M1. $A \Rightarrow S$: A, B, N_A
M2. $S \Rightarrow A$: $S, \{S, A, N_A, K_B\}K_S^{-1}$

Il protocollo ammette un 'binding attack':

M1.1. $A \Rightarrow M(S)$: A, B, N_A
M2.1. $M(A) \Rightarrow S$: A, M, N_A
M2.2. $S \Rightarrow M(A)$: $S, \{S, A, N_A, K_M\}K_S^{-1}$
M1.2. $M(S) \Rightarrow A$: $S, \{S, A, N_A, K_M\}K_S^{-1}$

Fix: includere il nome di B in M2.

M2. $S \Rightarrow A$: $S, \{S, A, N_A, B, K_B\}K_S^{-1}$

Esercizio

Analizzare questo protocollo, studiare un attacco e fornire il 'fix'

M1. A⇒B : $\{N_A\}K_{AB}$
M2. B⇒A : $\{N_A+1\}K_{AB}$

Il protocollo ammette un 'parallel sessions attack' (con 'oracolo'):

M1.1. A⇒M(B) : $\{N_A\}K_{AB}$
M2.1. M(B)⇒A : $\{N_A\}K_{AB}$
M2.2. A⇒M(B) : $\{N_A+1\}K_{AB}$
M1.2. M(B)⇒A : $\{N_A+1\}K_{AB}$

- A agisce come un oracolo: risponde correttamente a se stessa
- A crede che B è operativo, mentre B potrebbe non esistere piu'
- FIX: aggiungere il nome di A nel messaggio M1.

M1. A⇒B : $\{N_A,A\}K_{AB}$

Three-Pass Protocol, SRA
Shamir, Rivest e Adleman

Assume l'esistenza di cifrari simmetrici 'commutativi'

$$E_{KA}(E_{KB}(P)) = E_{KB}(E_{KA}(P))$$

La chiave segreta di A è K_A e quella di B è K_B

M1. $A \Rightarrow B$: $E_{KA}(M)$
M2. $B \Rightarrow A$: $E_{KB}(E_{KA}(M))$
M3. $A \Rightarrow B$: $E_{KB}(M)$

soggetto a molti tipi di attacchi

- M puo' usare Alice come un oracolo

M1. $A \Rightarrow M(B)$: $E_{KA}(M)$
M2. $M(B) \Rightarrow A$: $E_{KA}(M)$
M3. $A \Rightarrow M(B)$: M

- Oppure

M1. $A \Rightarrow M(B)$: $E_{KA}(M)$
M1'. $M(B) \Rightarrow A$: $E_{KA}(M)$
M2'. $A \Rightarrow M(B)$: M
M2. $M(B) \Rightarrow A$: bogus
M3. $A \Rightarrow M(B)$: $E_{KA}(bogus)$

Encrypted Key Exchange

EKE: A e B condividono una password P e usano EKE per autenticarsi mutuamente e per generare una chiave di sessione

M1. $A \Rightarrow B$: $E_P(K_A)$
M2. $B \Rightarrow A$: $E_P(E_{KA}(K))$
M3. $A \Rightarrow B$: $E_K(N_A)$
M4. $B \Rightarrow A$: $E_K(N_A, N_B)$
M5. $A \Rightarrow B$: $E_K(N_B)$

ove P è usata come chiave simmetrica, K_A è una chiave pubblica generata casualmente da A (con corrispondente chiave privata K_A^{-1}), and K è una chiave di sessione generata a caso da B

EKE con DH

EKE può essere realizzato con algoritmi a chiave pubblica: RSA, Diffie Hellman, ElGamal, ... Per esempio, con Diffie-Hellman, dati (α, p), K è generata automaticamente e il protocollo è semplificato

1. Alice sceglie un numero casuale grande a_A e manda a Bob

   M1. $A \Rightarrow B$: $A, \alpha^{aA} \bmod p$

2. Bob sceglie un numero casuale grande a_B e calcola $K = \alpha^{aAaB} \bmod p$. Sceglie la stringa casuale R_B e manda a Alice

   M2. $B \Rightarrow A$: $E_P(\alpha^{aB} \bmod p), E_K(R_B)$

3. Alice decifra per ottenere $\alpha^{aB} \bmod p$, e calcola K per decifrare R_B. Genera un'altra stringa casuale R_A e manda a Bob

   M3. $A \Rightarrow B$: $E_K(R_A, R_B)$

4. Bob controlla che R_B sia quello da lui trasmesso e 'risponde' alla 'sfida' di Alice

   M4. $B \Rightarrow A$: $E_K(R_A)$

Simple Kerberos

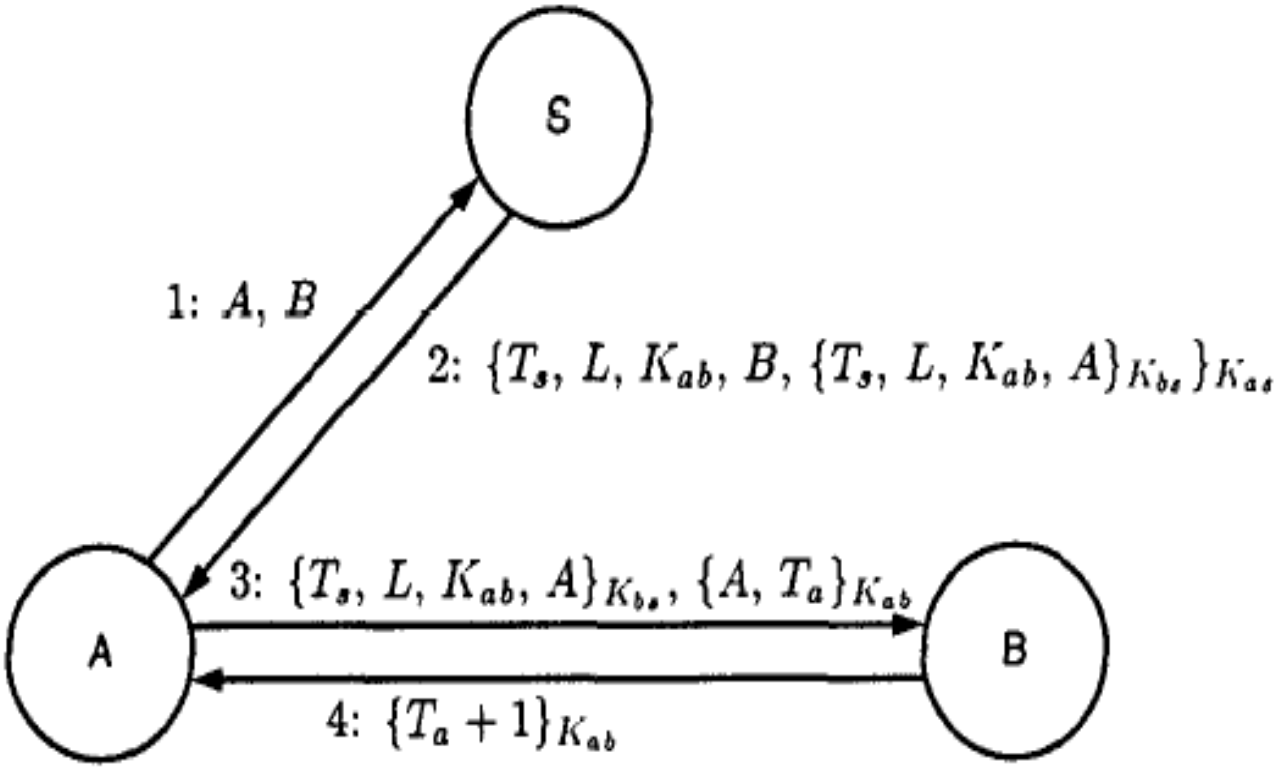


Fig. 1. The Kerberos Protocol.

Andrew RPC Handshake

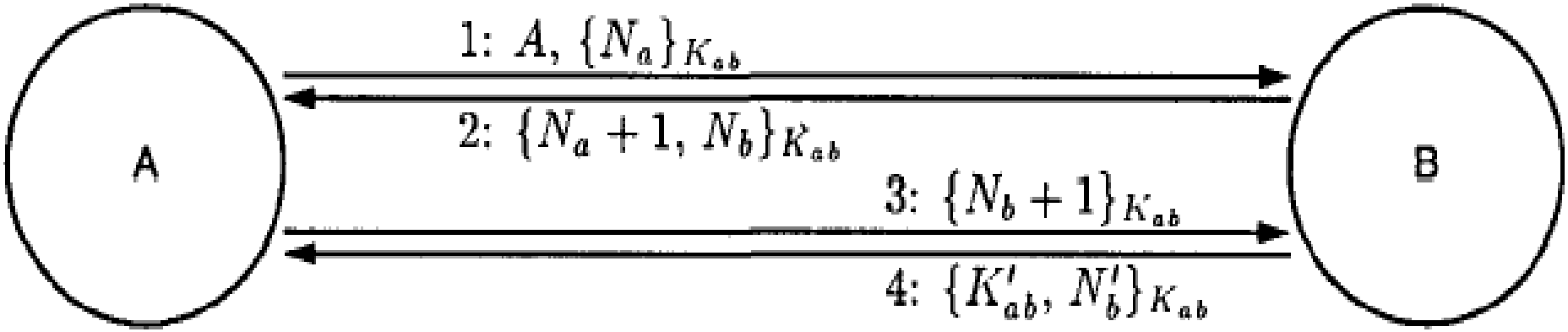


Fig. 2. The Andrew Square RPC Handshake.

Needham-Schroeder Public Key

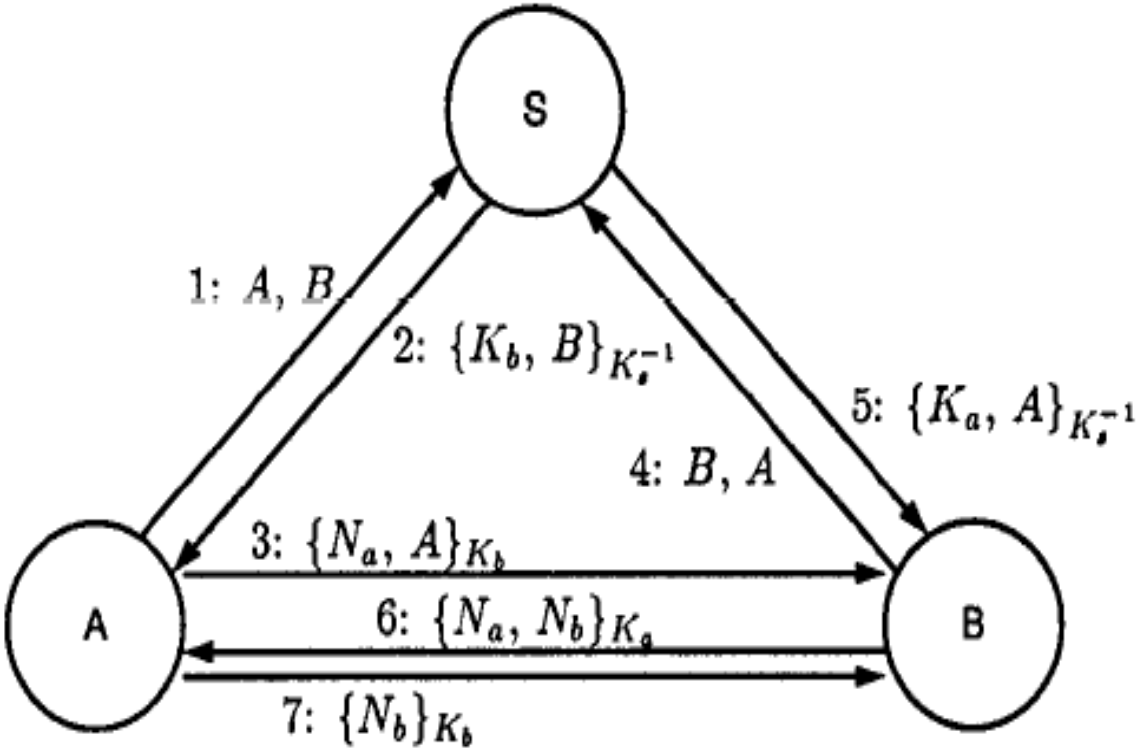


Fig. 3. The Needham–Schroeder Public-Key Protocol.

CCITT X.509 Three-Way Authentication

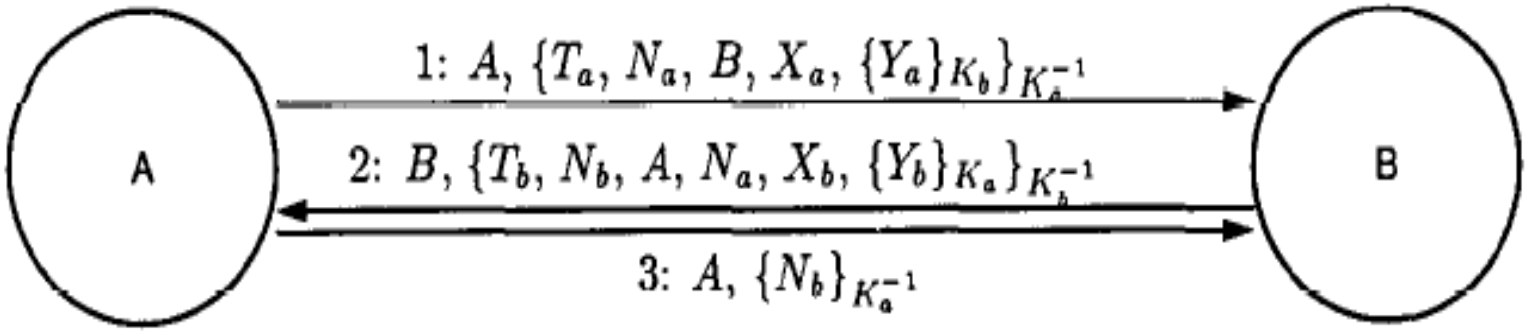


Fig. 4. The CCITT X. 509 Protocol.

Message 1. $A \rightarrow B: A, \{T_a, N_a, B, X_a, \{Y_a\}_{K_b}\}_{K_a^{-1}}$.

Message 2. $B \rightarrow A: B, \{T_b, N_b, A, N_a, X_b, \{Y_b\}_{K_a}\}_{K_b^{-1}}$.

Message 3. $A \rightarrow B: A, \{N_b\}_{K_a^{-1}}$.

Xa, Xb: dati di A , B da firmare

Ya, Yb: chiavi di sessione di A, B

DH - Cookies against DoS

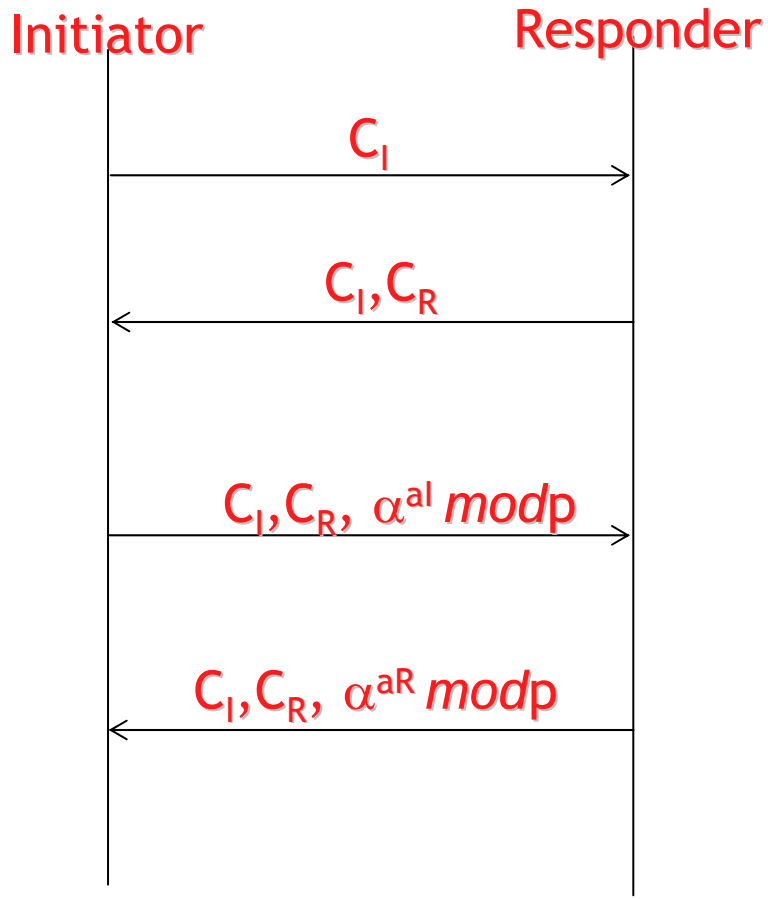
COOKIE di X

$C_X = h(IP_X, PA_X, Nonce_X, Secret_X)$

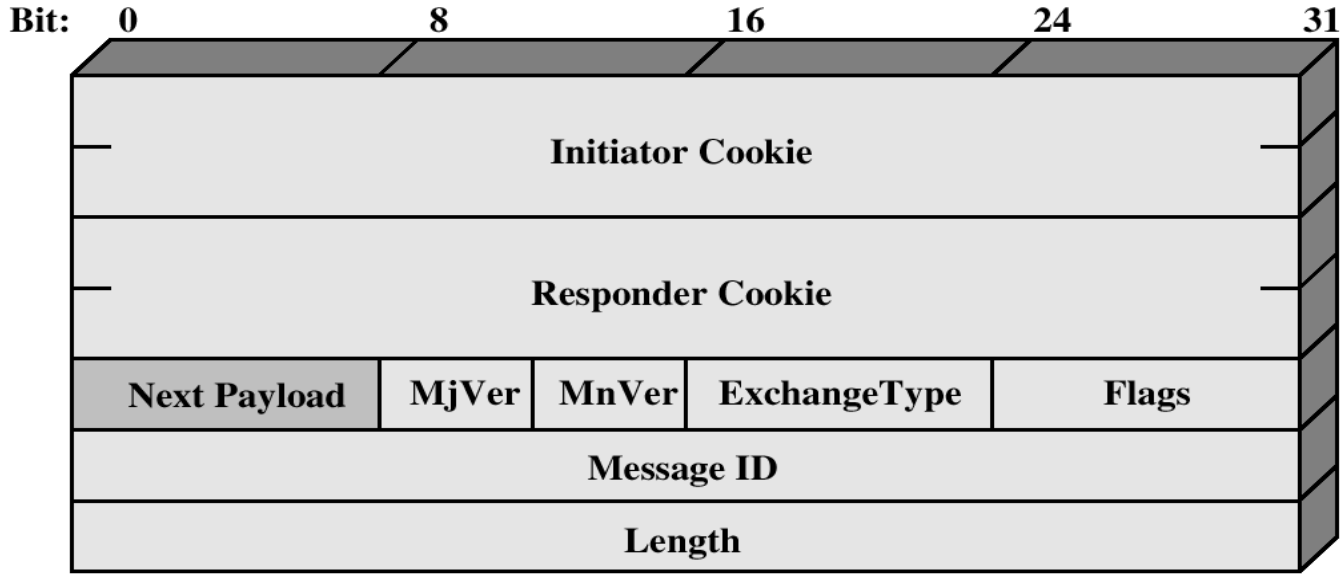
I e R si scambiano i cookie che contengono in codice hash gli indirizzi IP e di porta mescolati con un nonce, e spesso anche con un altro numero segreto

I messaggi che contengono coppie di cookie con indirizzi IP differenti vengono scartati

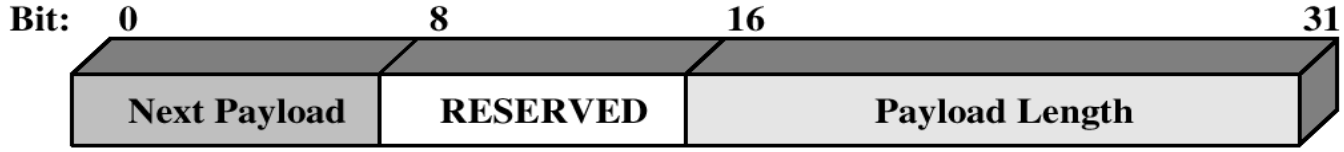
L'attaccante DoS deve associare ai messaggi di flooding che manda anche la coppia di coockie



ISAKMP Header



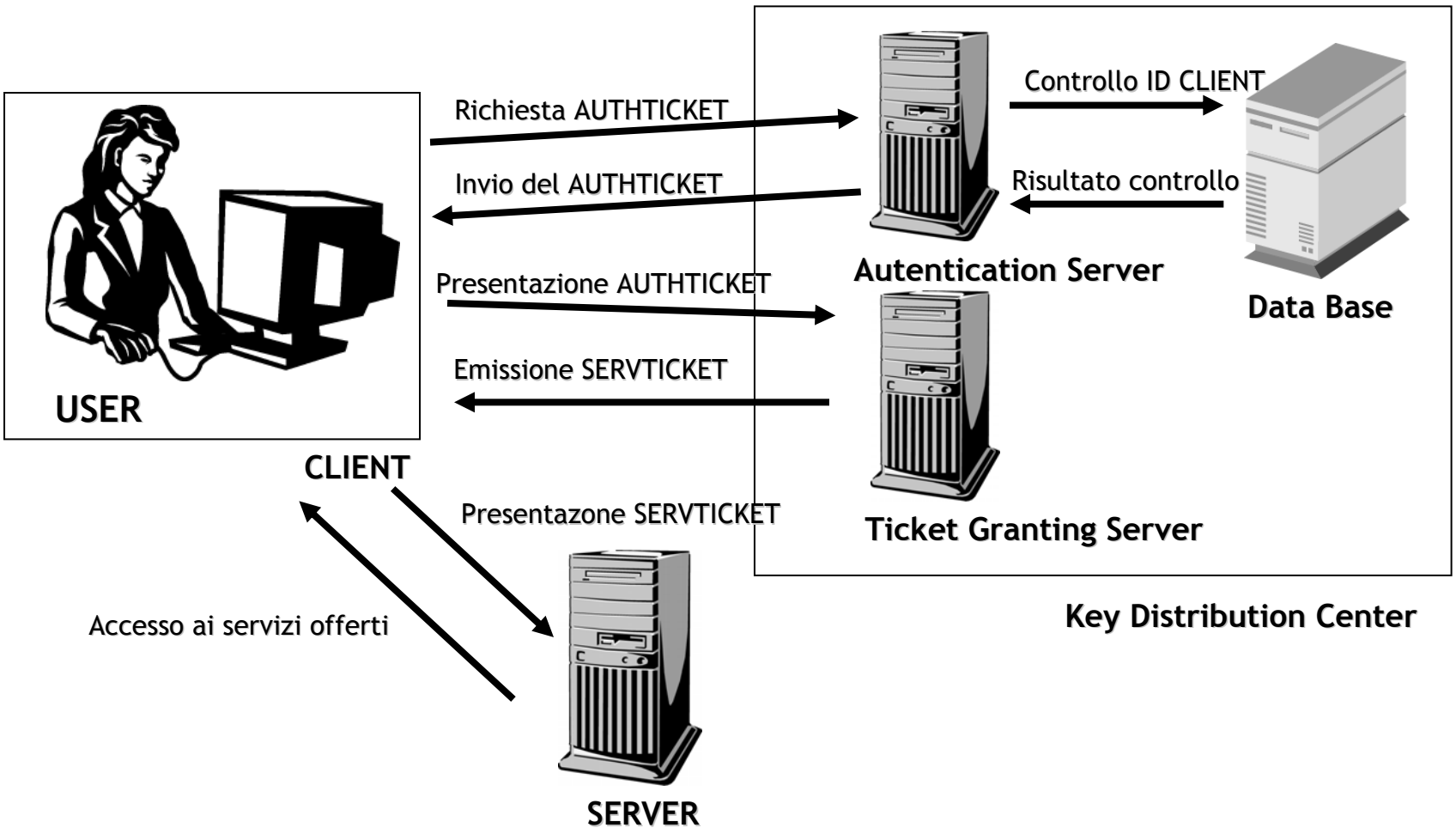
(a) ISAKMP Header



(b) Generic Payload Header

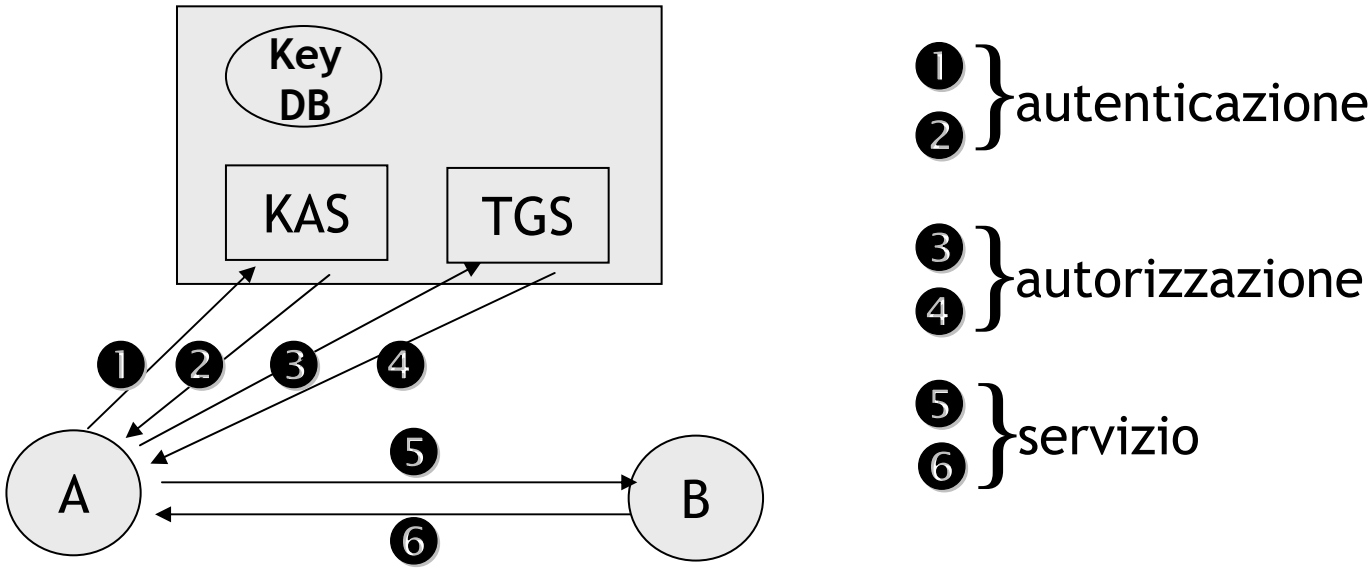
Cookie = h(IP_address, Port_address, Nonce, Secret) [64 bit?]

Kerberos - schema di principio



Kerberos

- Sviluppato dal MIT negli anni ottanta
  - Version IV (1989) ancora usata
  - Version V (1993) oggi è lo standard Kerberos
- Basato su NS, usa i Timestamp e richiede orologi sincronizzati
- Il servizio di 'Ticket Granting' 'Erogazione Biglietti' permette nuove richieste di sessione senza dover ridigitare UID e Password



Kerberos IV, semplificato

- M1. A si collega (login) con la workstation per entrare in rete
- M2. KAS accede alla base di dati e manda a A una chiave di sessione AuthKey e un ticket cifrato AuthTicket. AuthKey ha un tempo di vita di alcune ore: quando AuthKey scade A è automaticamente scollegato (logout)
- M3. tutte le volte che A vuole accedere alla risorsa B, A presenta AuthTicket a TGS insieme ad un autenticatore per dimostrare che AuthTicket era stato rilasciato proprio a lei
- M4. TGS manda a A una nuova chiave di sessione ServKey con un tempo di vita di alcuni minuti e un nuovo ticket ServTicket
- M5. tutte le volte che A vuole accedere alla risorsa B, presenta ServTicket insieme ad un autenticatore
- M6. B replies

Kerberos IV, semplificato

- Autenticazione – ticket granting
- M1. A→KAS : A, TGS, τ_{A1}
- M2. KAS→A : $\{AuthKey, TGS, \tau_{KAS}, \underbrace{\{A, TGS, AuthKey, \tau_{KAS}\}K_{TGS}}_{AuthTicket}\}K_{AS}$
- Autorizzazione – service granting
- M3. A→TGS : $\underbrace{\{A, TGS, AuthKey, \tau_{KAS}\}K_{TGS}}_{AuthTicket}, \underbrace{\{A, \tau_{A2}\}AuthKey}_{Authenticator}, B$
- M4. TGS→A : $\{ServKey, B, \tau_{TGS}, \underbrace{\{A, B, ServKey, \tau_{TGS}\}K_{BS}}_{ServTicket}\}AuthKey$
- Servizio - servicing
- M5. A→B : $\underbrace{\{A, B, ServKey, \tau_{TGS}\}K_{BS}}_{ServTicket}, \underbrace{\{A, \tau_{A3}\}ServKey}_{Authenticator}$
- M6. B→A : $\{\tau_{A3}+1\}ServKey$

Kerberos IV, Secret Key

- Autenticazione
- M1. A→KAS : ID_A, ID_{TGS}, T_{A1}
- M2. KAS→A : $\{AKey_{ATGS}, ID_{TGS}, T_{KAS1}, L_{KAS1}, \{ID_A, IP_A, ID_{TGS}, AKey_{ATGS}, T_{KAS1}, L_{KAS1}\}K_{TGS}\}K_{AS}$

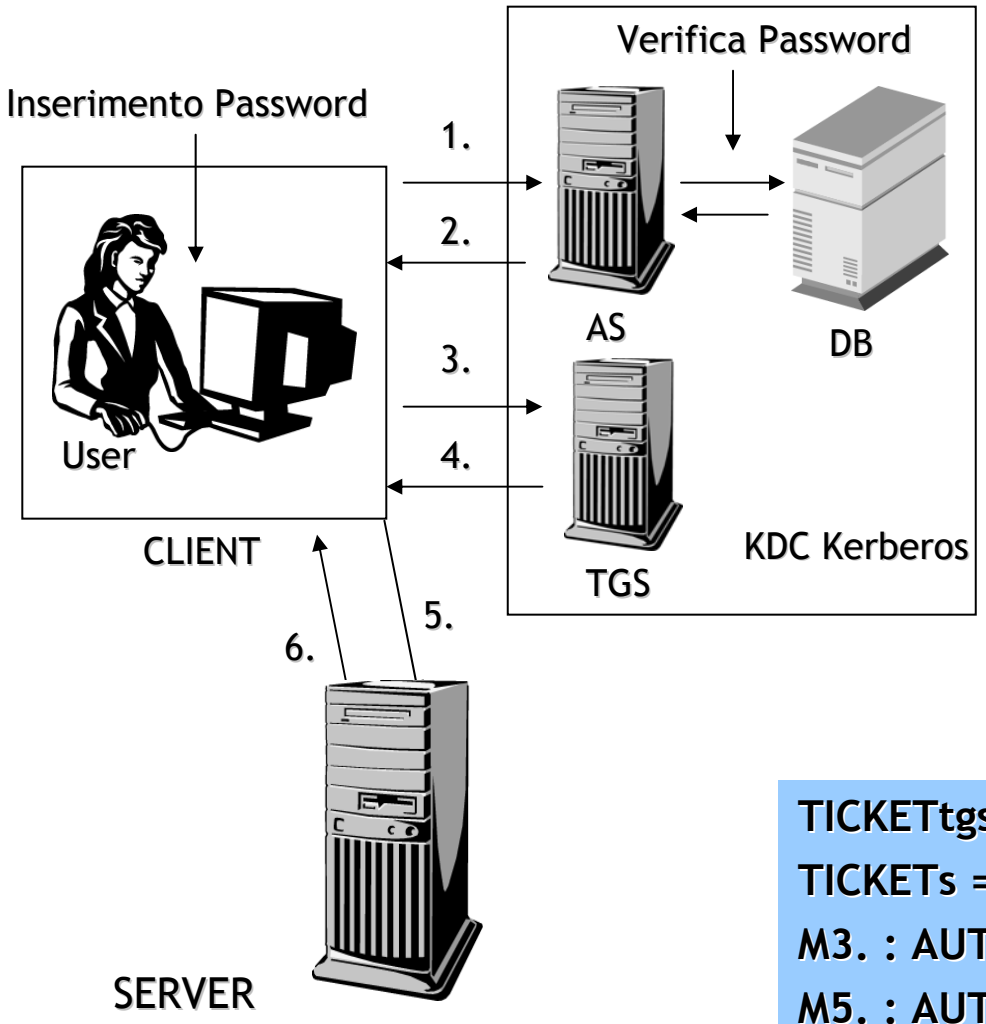
  AuthTicket
- Autorizzazione
- M3. A→TGS : $\{ID_A, IP_A, ID_{TGS}, AKey_{ATGS}, T_{KAS1}, L_{KAS1}\}K_{TGS}, \{ID_A, IP_A, T_{A2}\}AKeyATGS, ID_B$

  AuthTicket — Authenticator
- M4. TGS→A : $\{SKey, ID_B, T_{TGS1}, \{ID_A, IP_A, ID_B, SKey, T_{TGS1}, L_{TGS1}\}K_{BS}\}AKeyATGS$

  ServTicket
- Servizio
- M5. A→B : $\{ID_A, IP_A, ID_B, SKey, T_{TGS1}, L_{TGS1}\}K_{BS}, \{ID_A, IP_A, T_{A3}\}SKey$

  ServTicket — Authenticator
- M6. B→A : $\{T_{A3}+1\}SKey$

Kerberos v4



- Una volta per ogni connessione
  - M1. C⇒AS:
    IDc, IDtgs, TS1
  - M2. AS ⇒ C:
    {Kctgs, IDtgs, TS2, TTL2, TICKETtgs}Kcas
- Una volta per tipo di servizio
  - M3. C⇒ TGS:
    IDs, TICKETtgs, AUTHENTICATORc1
  - M4. TGS⇒ C:
    {Kcs, IDs, TS4, TICKETs}Kctgs
- Una volta per sessione di servizio
  - M5. C ⇒ S:
    TICKETs, AUTHENTICATORc2
  - M6. S ⇒ C:
    {TS5 +1}Kcs

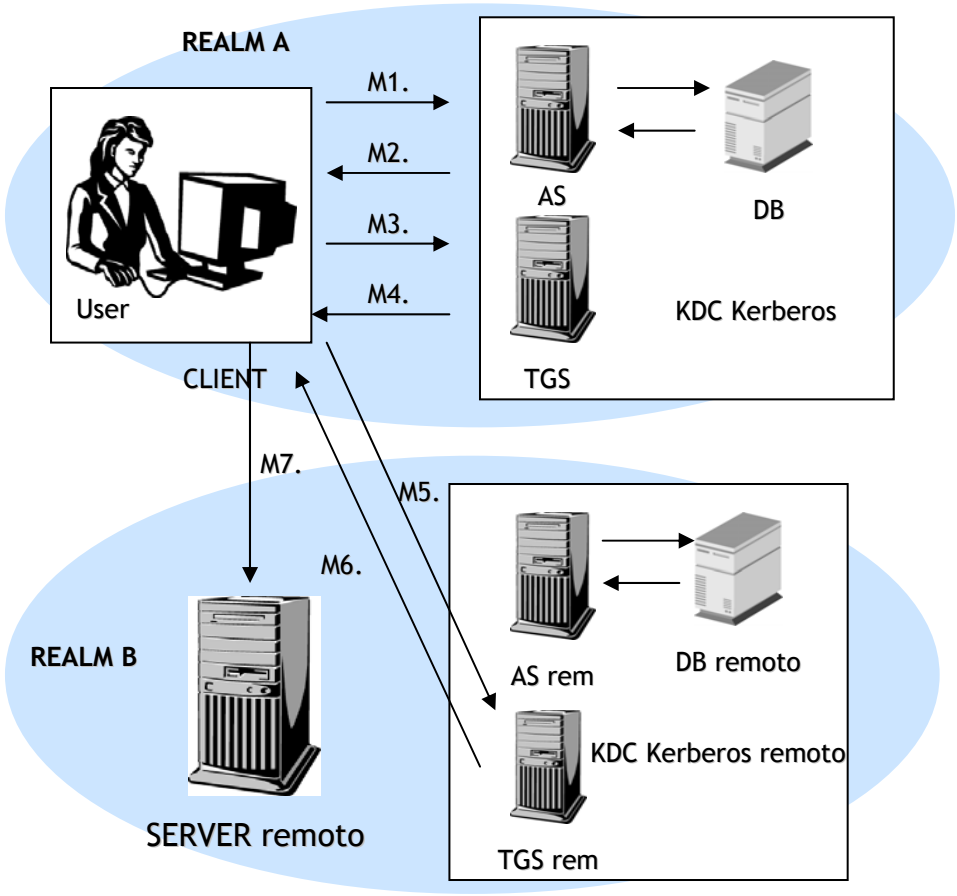
TICKETtgs = { Kctgs, IDc, IPc, IDtgs, TS2, TTL2}Ktgs
TICKETs = {Kcs, IDc, IPc, IDs, TS4, TTL4}Ks
M3. : AUTHENTICATORc1 = {IDc, IPc, TS3}Kctgs
M5. : AUTHENTICATORc2 = {IDc, IPc, TS5}Kcs

Kerberos Realm, e Kerberos multipli

- Un 'reame' (realm) Kerberos, ambiente di servizio Kerberos, consiste di
  - Un server Kerberos
  - Un certo numero di clienti (utenti)
  - Un certo numero di server di applicazioni
- E richiede che
  - Tutti gli utenti del 'reame' siano registrati con il server Kerberos, il server possiede gli 'identificativi' (UID: User IDentification) in chiaro e le 'parole d'ordine' (password) in codice hash di tutti gli utenti
  - Tutti i server di applicazioni del reame sono registrati con il server Kerberos, ciascun server condivide una chiave segreta con il server Kerberos
- Reti di clienti e di server sotto differenti organizzazioni amministrative costituiscono 'realm' differenti
- Kerberos fornisce autenticazioni inter-regno (inter-realm)
  - Gli utenti di un certo regno vogliono accedere a server di altri regni
  - I server di un certo regno vogliono concedere accesso a utenti di altri regni, a patto che tali utenti 'stranieri' vengano autenticati

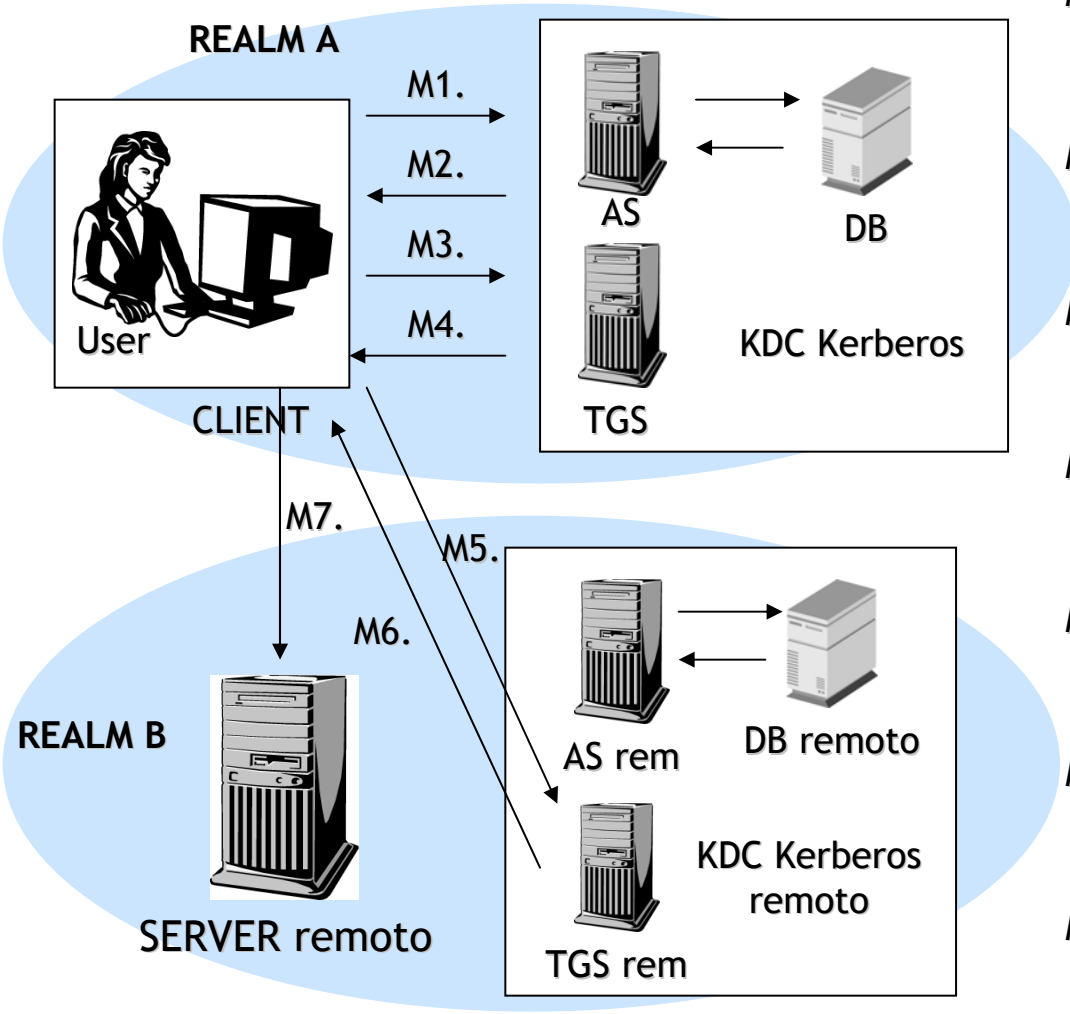
Regni Multipli

- Il server Kerberos in ciascun 'realm' condivide una chiave segreta con ciascuno dei server Kerberos degli altri 'realm'
  - Il server Kerberos in un 'realm' deve fidarsi del server Kerberos del 'realm' remoto per l'autenticazione degli utenti
  - Inoltre, anche Il server applicativo del 'realm' remoto deve fidarsi dell'autenticazione fatta dal server Kerberos dell'altro 'realm'



Se ci sono n 'realm' interoperanti, allora ci sono n(n - 1)/2 chiavi sicure condivise dalle coppie di server Kerberos

Scambio tra Realm



M1. C ⇒ AS:

IDc, IDtgs, TS1

M2. AS ⇒ C:

{Kctgs, IDtgs, TS2, TTL2, TICKETtgs}Kc

M3. C ⇒ TGS:

IDtgsrem, TICKETtgs, AUTHENTICATORc

M4. TGS ⇒ C:

{Kctgsrem, IDtgsrem, TS4, TICKETtgsrem}Kctgs

M5. C ⇒ TGSrem:

IDsrem, TICKETtgsrem, AUTHENTICATORc

M6. TGSrem ⇒ C:

{Kcs, IDs, TS4, TICKETs}Kctgsrem

M7. C ⇒ S:

TICKETsrem, AUTHENTICATORc

Kerberos IV e Kerberos V

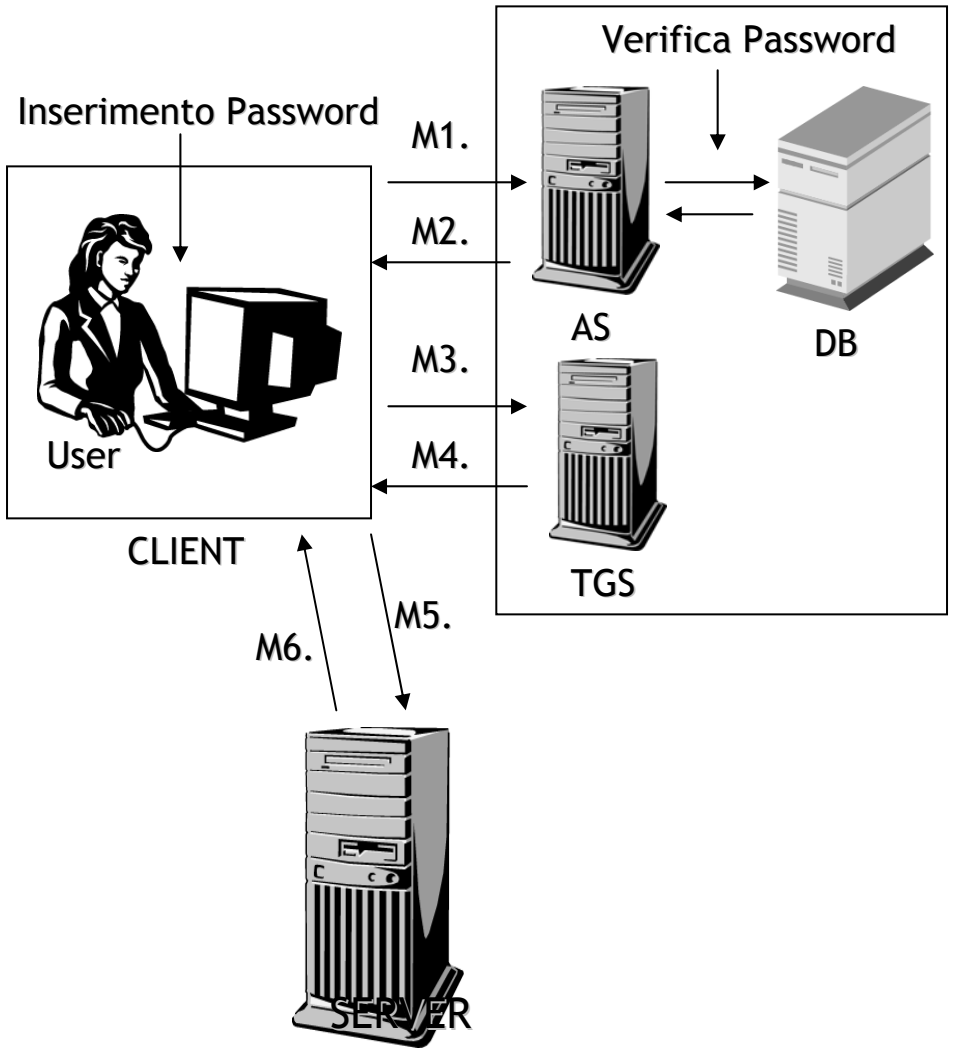
- Limitazioni di Kerberos IV
  - Vecchi autenticatori possono essere riutilizzati per repliche future, perlomeno durante il lifetime dei ticket: i server dovrebbero tenere in memoria tutti i ticket per contrastare questi attacchi, ma spesso non è praticabile
  - Gli autenticatori si basano sull'uso di orologi sincroni e non compromessi. Se un client è compromesso, allora il suo orologio può essere compromesso e si può attuare un attacco di reply
  - È basato su client e server fidati, nonchè sulla sicurezza e fidatezza dei server Kerberos e Ticket Granting
- Differenze tra Versione IV e V
  - M2. e M4. doppia cifratura (ridondante) eliminata
  - Encryption system dependence
  - Internet protocol dependence
  - Message byte ordering
  - Ticket lifetime
  - Authentication forwarding
  - Inter-realm authentication

Kerberos V, Any Key

- Autenticazione
- M1. A→KAS : $Options, ID_A, IDR_A, ID_{TGS}, T_{A1}, N_{A1}$
- M2. KAS→A : $IDR_A, ID_A, \{AKey_{ATGS}, ID_{TGS}, IDR_{TGS}, T_{KAS1}, L_{KAS1}, N_{A1}\}K_{AS}, \{Flag, IDR_A, ID_A, IP_A, AKey_{ATGS}, T_{KAS1}, L_{KAS1}\}K_{TGS}$ (AuthTicket)
- Autorizzazione
- M3. A→TGS : $Options, ID_B, T_{A2}, N_{A2}, \{Flag, IDR_A, ID_A, IP_A, AKey_{ATGS}, T_{KAS1}, L_{KAS1}\}K_{TGS}, \{ID_A, IDR_A, T_{A2}\}AKeyATGS,$ (AuthTicket, Authenticator)
- M4. TGS→A : $IDR_A, ID_A, \{SKey, ID_B, IDR_B, T_{TGS1}, L_{TGS1}, N_{A2}\}AKeyATGS, \{Flag, ID_A, IP_A, IDR_A, SKey, T_{TGS1}, L_{TGS1}\}K_{BS}$ (ServTicket)
- Servizio
- M5. A→B : $Opzioni, \{Flag, ID_A, IP_A, IDR_A, SKey, T_{TGS1}, L_{TGS1}\}K_{BS}, \{ID_A, IDR_A, T_{A3}, SubKey, SN\}SKey$ (ServTicket, Authenticator)
- M6. B→A : $\{T_{A3}, SubKey, SN\}SKey$

IDR= ID Realm; SN= Sequence Number; SubKey = Sottochiave di sessione

Keberos v5



- Una volta per ogni connessione

  M1. C ⟹ AS:
  Opzioni, IDc, Realmc, IDtgs, Tempi, Nonce1

  M2. AS ⟹ C:
  Realmc, IDc, TICKETtgs, {Kctgs, Tempi, Nonce1, Realmtgs, IDtgs}Kc

- Una volta per tipo di servizio

  M3. C ⟹ TGS:
  Opzioni, IDs, Tempi, Nonce2, TICKETtgs, AUThENTICATORc

  M4. TGS ⟹ C:
  Realmc, IDc, TICKETs, {Kcs, Tempi, Nonce2, Realms, IDs}Kctgs

- Una volta per sessione di servizio

  M5. C ⟹ S:
  Opzioni, TICKETs, AUTHENTICATORc

  M6. S ⟹ C:
  {TS2, SOTTOCHIAVE, SEQNo}Kcs

TICKETtgs = {Flag, Kctgs, Realmc, IDc, IPc, Tempi}Ktgs

Service Ticket Flags

- Initial -indica se il ticket è stato emesso dall'AS, invece che dal TGS
- Pre-Authent - specifica il tipo di preautenticazione del client
- Hw-Authent - specifica il tipo di supporto hw usato per l'autenticazione del client (es. smart card)
- Renewable - abilita la rinnovabilità del ticket introducendo due campi ulteriori, uno per la validità stessa e l'altro per indicare il periodo massimo consentito per i rinnovi
- May-Postdate - richiede un ticket postdatato
- Postdated - specifica che il ticket è postdatato
- Invalid - indica che il ticket al momento è invalido, in attesa di avere i requisiti di validità
- Proxiable - indica al TGS che potrà generare un ticket con indirizzo di rete diverso, per permettere ad un server proxy di operare in nome e per conto del client
- Proxy - indica che che il ticket è di tipo proxy
- Forwardable - indica al TGS che potrà generare un ticket con indirizzo di rete diverso per permettere ad un client di accedere ad un server di un altro realm.
- Forwarded - indica che il ticket è di tipo forwarded