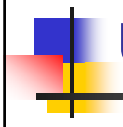

Ulteriori tecniche crittografiche

Stefano Paraboschi

20/05/2005

Outline

- In questa serie di lucidi vengono presentate una serie di tecniche crittografiche che sfruttano i principi illustrati precedentemente
- L'obiettivo è quello di mostrare risposte a particolari esigenze applicative

Interlock protocol

- Problema: Alice e Bob devono dialogare, senza CA e su un canale che ammette MITM
  1) Alice invia a Bob la propria chiave pubblica
  2) Bob invia ad Alice la propria chiave pubblica
  3) Alice cifra il messaggio con la chiave pubblica di Bob; manda metà del messaggio a Bob (solo bit dispari?)
  4) Bob cifra il proprio messaggio con la chiave pubblica di Alice; manda metà del messaggio ad Alice
  5) Alice manda la metà rimanente del messaggio a Bob
  6) Bob combina le due metà e decifra il messaggio; manda la metà rimanente ad Alice
  7) Alice combina le due metà e decifra il messaggio di Bob

Interlock protocol/2

- Mallory può sostituire la propria chiave pubblica a quelle di Alice e Bob nello scambio, ma non ha modo di estrarre l'informazione dalla metà del messaggio che ha in mano.
- Il ruolo di eavesdropper non può essere realizzato, a meno che il dialogo sia prevedibile
- Interessante, ma si è mostrato come in un contesto di rete si possano simulare malfunzionamenti che limitano la forza del metodo

Tecnica SKEY

- Problema: inviare password su un canale aperto monodirezionale (altrimenti, challenge-response: dammi l'hash della stringa che combina il random che ti do con la chiave che conosci)
- Si prende un random x_0 e si calcola x_1=H(x_0); x_2=H(x_1); ... x_N=H(x_N-1)
- Si dà la lista x_0..x_N-1 all'utente
- Il server conserva solo l'ultimo valore x_N
- Ogni volta che l'utente si collega
  - Invia l'ultimo valore della serie (x_N-1 la prima volta)
  - Server calcola H(x_i) e confronta con il valore memorizzato
  - Server salva x_i al posto del valore precedente
  - L'utente butta via il valore consumato
- Problema: capacità finita

Needham-Schroeder

- Usa una trusted third party e cifratura simmetrica per il dialogo tra Alice e Bob. Ognuno ha una chiave simmetrica di dialogo con la TTP
  1. Alice -> TTP: A,B,Ra
  2. TTP genera chiave random K;
     TTP->Alice: $E_A(Ra,B,K,E_B(K,A))$
  3. Alice->Bob: $E_B(K,A)$
  4. Bob apre il messaggio, crea Rb;
     Bob->Alice: $E_K(Rb)$
  5. Alice conferma a Bob la ricezione;
     Alice->Bob: $E_K(Rb-1)$
  6. Bob verifica che il messaggio ricevuto sia corretto

Needham-Schroeder/2

- Problema: la chiave K è critica. Se Mallory entra in possesso della traccia di una conversazione e conosce la K corrispondente, può realizzare un replay attack (dal punto 3 in avanti)
- L'uso di timestamp rende la soluzione sicura

Kerberos

- È una variante di Needham-Schroeder. Anche in questo caso ciascuno condivide una chiave simmetrica con una TTP/Authorization server
  1. Alice->TTP: A,B
  2. TTP sceglie T timestamp, L lifetime, K session key;
     TTp-> Alice: $E_A(T,L,K,B,E_B(T,L,K,A))$
  3. Alice->Bob: $E_K(A,T)$, $E_B(T,L,K,A)$
  4. Bob->Alice: $E_K(T+1)$

Kerberos/2

- Problemi:
  - assume che gli orologi siano tutti sincronizzati
  - Rispetto a soluzioni con chiavi asimmetriche e CA, è necessario il coinvolgimento della TTP in ogni scambio
- Kerberos come viene implementato distingue Authorization Server e Ticket Granting Server e server generici, così come ticket (molti usi, generato da AS (Ticket Granting Ticket TGT) e TGS) ed authenticator (uso singolo, generato dal client per usare i server),

Cifratura a Multiple public key

- Si possono costruire soluzioni in cui 3 (in generale *n*) esponenti consentono di riottenere un messaggio.
  - Alice: K_A
  - Bob: K_B
  - Carol: K_C
  - Dave: K_A e K_B
  - Ellen: K_B e K_C
  - Frank: K_A e K_C
- Messaggi di Alice possono essere letti solo da Ellen o da Bob e Carol in cooperazione; etc.

Cifratura per gruppi con Multiple public key

- Se voglio inviare messaggi a un gruppo di *n* utenti, creo *n* chiavi e l'utente *i* riceve tutte le chiavi meno la *i*-esima
- Il messaggio viene poi inviato cifrandolo con tutte le chiavi dei destinatari autorizzati; gli altri non potranno decifrarlo
  - Bisogna applicare tutte le chiavi mancanti per ritornare al messaggio in chiaro; se la mia non viene applicata, dato che io non la posseggo, non sono in grado di decifrare
- Vantaggi: Un solo messaggio in broadcast va bene per tutti
- Problema: la gestione delle chiavi è complicata e ciascuno sa quanti altri vedono il messaggio

Condivisione di segreti

- Vi sono tecniche crittografiche che consentono di ottenere un segreto solo grazie alla collaborazione di un certo numero di utenti
- Le tecniche più semplici si secret splitting usano sequenze random in XOR con il segreto
  - Ad es.: R, S, e T sequenze di bit random;
    - M ⊗ R ⊗ S ⊗ T = U
    - R ad Alice; S a Bob; T a Carol; U a Dave
  - La ricostruzione richiede tutti e 4
- Con il secret sharing si realizzano meccanismi a soglia
  - Schema a soglia (m,n): servono *m* di *n* utenti per aprire il messaggio

Tecniche a soglia

- La tecnica classica per (*m*,*n*) prevede di costruire un polinomio con grado *m*-1, in cui il messaggio da proteggere rappresenta l'ultimo parametro (gli altri sono scelti a caso)
- Ciascun utente riceve un punto del polinomio
- Con *m* punti si è in grado di ricostruire il polinomio, e quindi il messaggio
- Nei dettagli, si usa l'aritmetica modulare e un primo *p* come modulo di valore superiore al messaggio
- Si può dimostrare facilmente che, se i parametri del polinomio sono scelti a caso, la tecnica è arbitrariamente sicura
- Vi sono altre tecniche simili con base geometrica o algebrica

Firme digitali Fail-stop

- Si tratta di firme digitali che possono essere verificate con una chiave pubblica ben definita, ma prodotte da diverse (2^n) chiavi private
- Mallory può trovare una chiave per firmare, ma Alice può dimostrare che non è stata lei (perché la sua chiave produce una firma diversa)

Bit commitment

- Si tratta di tecniche che permettono di prendere un impegno su un certo bit (o stringa di bit) non revocabile, senza rivelare la scelta in anticipo
  - Ad es.: voglio mostrare che so predire il futuro; voglio giocare a pari-dispari (o bim-bum-bam) in rete in modo corretto
- Prima soluzione
  - Prima fase
    - Bob invia un nonce *R* ad Alice
    - Alice sceglie K ed invia a Bob Ek(*R*,*b*)
  - Seconda fase
    - Alice invia a Bob la chiave K
    - Bob verifica il bit *b*
  - L'assunzione è che Alice non sia in grado di trovare una chiave K' tale che Ek(R,b)=Ek'(R,b')

Bit commitment/2

- Altra soluzione
  - Prima fase
    - Alice sceglie dei nonce *R1* e *R2*
    - Alice costruisce un messaggio concatenando *R1, R2, b*
    - Alice invia a Bob *H(R1,R2,b)* ed *R1*
  - Seconda fase
    - Alice invia a Bob il messaggio completo *R1, R2, b*
    - Bob verifica che lo hash coincida e ha quindi la garanzia che sia stato precedentemente scelto *b*
  - Assunzione: non è facile trovare 2 collisioni nella funzione hash, altrimenti se *H(R1,R2,b)=H(R1,R2',b')* Alice può ingannare Bob
  - Il vantaggio rispetto alla soluzione precedente è che nella prima fase Bob ha solo un ruolo passivo