Autenticazione

Gianluigi Me
Universita` di Roma “Tor Vergata”
me@disp.uniroma2.it

Argomenti

- Principi di Autenticazione
- Password
- Challenge-Response
- Protocolli di tipo *Zero Knowledge*
- Autenticazione Bluetooth
- Autenticazione GSM/GPRS/UMTS
- Autenticazione EAP

Argomenti (cont.)

- Autenticazione Client/Server (Kerberos)
- Servizi di autenticazione di utenti remoti (RADIUS)

- Sicurezza nell'accesso al Web (SSL)
- Sicurezza nelle connessioni (SSH)
- Sicurezza dell'e-mail (PGP, S/MIME)

Principali Obiettivi

- Se Alice e Bob sono entrambi onesti, allora Alice dovrebbe riuscire ad identificarsi con successo rispetto a Bob
- Bob non può riutilizzare lo scambio di informazioni avuto con Alice per identificarsi come Alice rispetto a Carla (trasferibilità)
- Carla non deve potersi identificare (con **sufficiente** probabilità) come Alice rispetto a Bob (impersonificazione)

Requisiti richiesti

- Vogliamo raggiungere tutti e tre gli obiettivi
  - anche se Carla riesce ad acquisire informazioni su un gran numero di procedure di autenticazione tra Alice e Bob
  - anche se Carla ha scambiato legittimamente informazioni sull'autenticazione con Alice e/o Bob.
  - Anche se Carla ha la possibilità di tentare un *grande* numero di procedure di autenticazioni allo stesso tempo.
- Con *grande* si intende polinomialmente grande (non esponenziale).
- I protocolli *Zero Knowledge* richiedono che l'esecuzione dei tentativi di autenticazione non forniscano informazioni al candidato.

Basi per l'Identificazione

- Qualcosa che si conosce…
  - Passwords, PINs, Segreto o chiave
- Qualcosa che si possiede….
  - Dispositivi fisici: carte magnetiche, smart card, generatori di password
- Qualcosa che si è…
  - Biometrics (impronte digitali, riconoscimento dell'iride , della voce, della calligrafia), caratteristiche del *keyboarding*.
- Un posto dove si è...
  - Locazione tramite GPS
- Idealmente, due o più meccanismi dovrebbero essere utilizzati
- Nella maggior parte dei casi l'autenticazione deve avvenire in tempo reale

Proprietà dei protocolli di autenticazione

- Reciprocità dell'autenticazione
- Complessità
  - Efficienza computazionale
  - Efficienza della comunicazione
- Costo
- Terze parti
  - Se una terza parte è necessaria
  - Se è necessaria in tempo reale
  - Natura della fiducia richiesta dalla terza parte
- Quali garanzie di sicurezza sono offerte
- Come e dove sono mantenute le chiavi e/o i

Passwords

(Autenticazione Debole)

Passwords

- Stringhe di 6-8 caratteri che permettono l'identificazione
  - Password fisse/PIN, password *one-time*
- Forma di autenticazione su "qualcosa che sai".
- Proprietà
  - Non reciprocità – normalmente le password permettono identificazione unilaterale
  - Bassa complessità – molto efficienti sia dal punto di vista computazionale che dal punto di vista della comunicazione
  - Normalmente non è richiesta una terza parte, (il *Single Sign On* rappresenta l'eccezione)
  - La chiave è normalmente tenuta a memoria dall'utente ed in un file dal sistema

Attacchi alla Password fisse

- Attacchi *Replay*
  - Osservazione durante l'inserimento, *social engineering*, cavalli di Troia
  - *Eavesdropping* su un canale di comunicazione in chiaro
- Ricerca esaustiva
  - Random o sistematicamente tentare le password online
  - Ricerca Offline su un file di password
- Attacchi tipo dizionario
  - Si assume che non tutte le password abbiano la stessa probabilità
  - E' sufficiente che un utente scelga una password debole
- Attacchi su password note
  - Molti sistemi sono provvisti di password "standard" (di servizio o ripristino).

Sicurezza delle Password Fisse

- File di password criptati
  - Scopo: evitare un troppo facile accesso dallo staff interno
  - Normalmente, la password non è criptata usando una chiave simmetrica, ma piuttosto usando una funzione di hash *one-way*
    - e.g., la password di Alice è immagazzinata come h(Alice,pwd)
- Regole sulle Password
  - Scopo: avere password con "alta entropia"
  - Normalmente, regole sintattiche e procedurali
    - la password deve avere almeno 8 caratteri, o usare *passphrase*
    - la password deve includere cifre e caratteri speciali
    - la password non dovrebbe avere un significato
    - è necessario modificare la password con regolarità (tipicamente 30 giorni)
    - Non si deve utilizzare la stessa password su sistemi diversi

Sicurezza delle Password Fisse

- Rallentamento nella verifica della password
  - Lo scopo è limitare l'uso di programmi per la ricerca esaustiva
  - Normalmente si ottiene applicando in maniera ricorsiva una funzione più semplice in cui l'iterazione $t+1$ usa il risultato dell'iterazione t
  - Deve essere accettabile per gli utenti (*e.g.,* un secondo)

Da qui

- **Salting**
  - Scopo: limitare l'uso di attacchi simultanei tipo dizionario
  - Aggiunge alcuni bit alla password prima dell'hashing
  - Normalmente, qualcosa legato al tempo o allo

Spazio delle Password

- Numero pw ($\log_2 c^n$)

c n	26 minus.	36 minus. alfanum.	62 mix	95 tastiera
5	23.5	25.9	29.8	32.9
6	28.2	31.0	35.7	39.4
7	32.9	36.2	41.7	46.0
8	37.6	41.4	47.6	52.6

- Tempo di ricerca

(5000/sec)

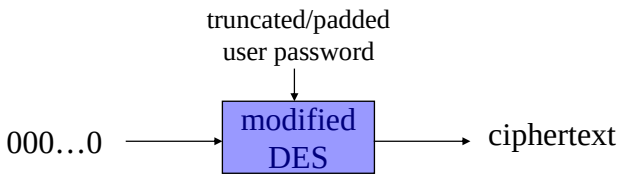
c n	26 minus.	36 minus. alfanum	62 mix	95 tastiera
5	0.67hr	3.4hr	51hr	430hr
6	17hr	120hr	130dy	4.7yr
7	19dy	180dy	22yr	440yr
8	1.3yr	18yr	1400yr	42000yr

Conclusione sullo Spazio delle Password

- Le password brevi, di sole lettere, sono facilmente identificabili
  - è cruciale aggiungere altri caratteri
  - è cruciale aumentare la lunghezza della password
- Riduzione dello spazio delle password
  - Una password appartenente ad uno spazio a bassa entropia (“dizionario”) riduce la dimensione dello spazio di ricerca
  - Funzioni di verifica troppo semplici aumentano il numero di tentativi per secondo
  - In un attacco ad un file di password è sufficiente che ci sia *una* password debole
- Password “random” di almeno 8 caratteri

Esempio: Password Unix

- Unix tiene tutte le password in un file (*e.g.,* /etc/passwd)
- La password serve come chiave per criptare 64 bit posti a zero, viene mantenuto questo testo criptato



- I primi 8 caratteri sono usati, completati con degli zeri se necessario e solo i primi 7 bit di ognuno sono presi per creare una chiave DES a 56-bit

Esempio: Password Unix

- Dal punto di vista criptografico, si può vedere che l'algoritmo è noto così come il *plaintext*.
- Il DES è ripetuto 25 volte
- La Password è "salted"
  - Sono usati 12 bits scelti a caso dal clock di sistema. Vengono utilizzati nella funzione di espansione del DES
  - In questo modo, 2^{12}=4096 variazioni sono richieste in qualsiasi attacco simultaneo di tipo "dizionario".
  - Non può essere utilizzato hardware DES *off-the-shelf.*

Esempio: Password Cracking (Wu)

- Tentativo di indovinare le passwords di 25.000 utenti Kerberos
- In due settimane, utilizzando 8 workstation, indovina 2.045 passwords

Length	2	3	4	5	6	7	8	9	10	>10
Percent	0.1	0.6	3.8	7	11	8	54	8	4.5	3

- Solo il 4% usava almeno un carattere alfanumerico
- 86% non richiedeva l'uso dello shift
- Alcuni account usavano date, o numeri di telefono
- Alcune password erano comuni a più di un account!
- 24% erano combinazioni di due parole
- 25% risultava da semplici trasformazioni di singole parole, *e.g.,* maiuscole, inversione, o raddoppiamento di una parola
  - Mettere in minuscolo un nome era la trasformazione più comune
  - "1" era il più comune prefisso/suffisso

Personal Identification Numbers

- Normalmente usati come "qualcosa che si conosce" in combinazione con "qualcosa che si possiede"
  - Molto spesso una carta di credito o di bancomat
  - Tipicamente brevi (4 cifre), per essere ricordati a memoria.
- Per impedire ricerche esaustive, carta viene ritirata (o disabilitata) dopo 3-4 tentativi.
- Per permettere l'uso con dispositivi *offline*, il PIN può essere immagazzinato sulla carta, alcune volte criptato con una "*master key*"
- Questa è una forma di autenticazione a due livelli, dove la seconda chiave ad alta entropia è immagazzinata sulla carta

Passkeys (passwd-derived keys)

- Password utente fatta corrispondere (mediante 1WHF) a una chiave crittografica (i.e., DES a 56-bit): chiamata *passkey*
- *passkey* usata per criptare la comunicazione
  - Più resistente della password contro un attacco di tipo replay (se ricerca esaustiva su passwd non piu` facile di passkey)
  - Per ottenere *passkey* che cambia nel tempo, aggiungere contatore alla password
  - A volte aggiunto *salt* basato su userid

One-time Password

- Verso autenticazione forte: soluzione parziale contro attacchi tipo *eavesdropping* e *replay*.
- Variazioni:
  - Liste condivise di one-time password
  - Aggiornamento sequenziale delle one-time password
    - Password *i+1* concordata quando avviene autenticazione con password *i*
  - Uso di funzioni *one-way* per creare una sequenza
    - Lamport: $P_i = H(P_{i+1})$, dove H( ) è una 1WF
      - Nota 1: autenticazione richiede un contatore
      - Note 2: un attacco tipo replay sarebbe possibile se la sequenza fosse in avanti

Identificazione Challenge-Response

(Autenticazione Forte)

Parametri variabili nel tempo

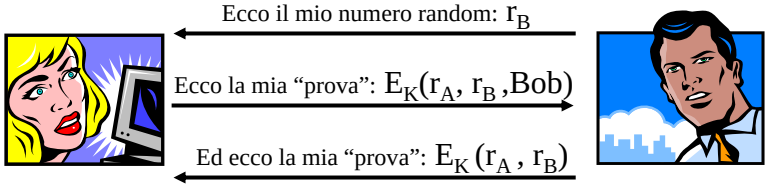
- Utili contro attacchi *replay* e su testi specifici
  - *e.g*, sequenze di numeri, numeri random one-time (*nonces*), timestamps, numeri random
- Il parametro dipendente dal tempo è generato da un partner mentre l'altro partner lega un'informazione criptata a questo numero per garantire la "freschezza"

Challenge-Response con chiavi simmetriche

- Le parti possono accordarsi a priori su una chiave, oppure la chiave può essere fornita da un server "affidabile" (trusted)
  - *e.g.,* protocolli come Kerberos o Needhan-Schroeder
- Esempio 1: autenticazione *one way* usando un time-stamp
  - Alice si identifica rispetto a Bob inviando in forma criptata il proprio time-stamp, usando la chiave segreta, $E_K(t_A)$
  - Per impedire a Carla di identificarsi con Alice, se la chiave è bidirezionale, Alice può inviare $E_K(Bob,t_A)$
- Esempio 2: usando numeri random
  - Prima, Bob invia ad Alice un numero random r_B
  - Quindi, Alice invia a Bob $E_K(Bob,r_B)$

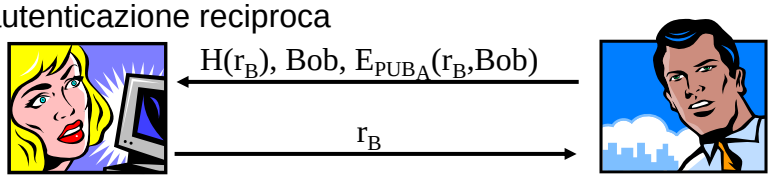
Challenge-Response con chiavi simmetriche (cont.)

- Autenticazione reciproca richiede un passaggio in più (può essere fatta sia utilizzando timestamps che numeri random)



Challenge Response con chiavi pubbliche

- Per autenticarsi, Alice deve dimostrare di conoscere la propria chiave privata
  - Decifra una “sfida” criptata usando la propria chiave pubblica
  - Oppure, firma digitalmente la sfida
- Anche in questo caso, un numero random può essere usato per impedire attacchi su testi specifici in un’autenticazione reciproca



Difesa dagli attacchi a sistemi Challenge-Response

- Attacco replay
  - Usare *nonces*, includere l’identità nella risposta
- Attacco di tipo *interleaving*
  - Concatenare i messaggi del protocollo
- Attacco “man-in-the-middle”
  - Autenticazione reciproca per impedire all’avversario di impersonare il sistema partner
- Attacco “*reflection*”
  - Include l’identità del partner, usare chiavi uni-direzionali
- Attacco su testi scelti
  - Usare un confounder in ogni messaggio
  - Usare protocolli *zero-knowledge*

Protocolli di identificazione *Zero-Knowledge*

Motivazione

- La password può rivelare il segreto di Alice a Bob, che può quindi impersonare Alice rispetto ad altri.
- Con i protocolli *challenge-response*, Alice rivela soltanto la conoscenza del segreto (non il segreto stesso)
- Ma un avversario abile potrebbe scegliere le “sfide” in modo che alcuni aspetti di questo segreto siano identificabili
- I protocolli ZK permettono ad Alice di rivelare la conoscenza del segreto senza fornire a Bob nessuna informazione sul segreto stesso
- Nota: l’RSA è ZK; la maggior parte dei protocolli ZK è più efficiente dell’RSA ma non può essere usata per la cifratura o la firma digitale

Zero Knowledge Protocols

- Let (n,e) be RSA public key. Alice wants to prove to Bob she knows her private key (without giving it away!).
- Namely, she wishes to prove she knows some m where $c = m^e \pmod{n}$.

Protocol 1

1. Alice picks random number r mod n and sends Bob $y = r^e$
2. Alice sends Bob $z = r \cdot m$
3. Bob checks that $z^e = y \cdot c$

- If Alice passes the test, she must know m (thus d)

Protocol 1

1. Alice picks random number r mod n and sends Bob $y = r^e$
2. Alice sends Bob $z = r \cdot m$
3. Bob checks that $z^e = y \cdot c$

If Alice passes the test, she must know m (thus d)

Problem: If bad Eve tries to impersonate Alice, she can cheat. I.e., she sends $y = r^e/c$ and $z = r$: this always satisfies 3.

Cut-and-Choose Protocol

1. Alice picks a random number r mod n and sends Bob $y = r^e$ - Commitment
2. Bob sends Alice either b = 0 or b = 1 - Challenge
3. Alice sends Bob $z = r \cdot m^b$ (either r or $r \cdot m$) - Response
4. If b = 0 Bob checks that $z^e = y$;
   if b = 1 Bob checks that $z^e = y \cdot c$

Protocol repeated many times (say 20):

- If any test fails, Bob calls the police

Intuition behind Cut-and-Choose

- Challenge requires Alice be capable of answering two questions:
  - One demonstrates her knowledge of m
  - The other is easy question (for honest provers) to prevent cheating
- A prover knowing m can answer both questions
- A cheating prover (bad Eve) can at best answer one of the two questions: has probability (1/2) of escaping detection

Correctness of the Protocol

1. (Completeness): Honest Alice who knows m always pass the protocol
2. (Soundness): If bad Eve tries to impersonate Alice without knowing m, her only chance to fool Bob is to guess b right:

- If guess is b=0, she sticks to the protocol and sends $y = r^e$
- If guess is b=1, she sends $y = r^e/c$.
- Her only chance to pass protocol is to be always lucky, i.e., to always guess right b, which she does w.p. (1/2) per round.
- Probability to always guess right b in 20 rounds is $2^{-20} \sim 10^{-6}$
- If more paranoid, increase number of rounds

Zero Knowledge

- Bob learns nothing from his interactions with Alice. How to prove this?

- What does Bob get from the protocol?
- Bit (Alice knowns m or not) plus transcript of conversation with Alice

- Is the transcript revealing something?
- No, Bob can construct such transcript on his own without interacting with Alice: i.e., (behaving like Eve) could generate a transcript with same identical probability distribution of true conversation withAlice

Feige-Fiat-Shamir Identification

- P proves knowledge of secret s to V (difficulty of extracting square roots modulo large integers n of unknown factorization)
- Arbitrator chooses random modulus $n = pq$ (1024 bits), p, q primes
- P computes $v = s^2 \bmod n$, and register v with arbitrator

1. P picks random $r < n$, computes $x = r^2 \bmod n$ and sends it to V
2. V sends b in {0,1} to P
3. If b = 0 P sends y = r to V; if b=1 P sends $Y = r \cdot s \pmod n$ to V;
4. If b = 0 V checks that $y^2 = x \pmod n$ (P knows sqrt(x)). If b = 1 V checks that $y^2 = x \cdot v \pmod n$ (P knows sqrt(v)).

Proprietà dei protocolli ZK

- Resistono agli attachi su testi scelti
- Nessun degrado del protocollo dovuta all'utilizzo
  - Nessuna informazione è rivelata in tempo polinomiale
- Confrontati con protocolli a chiave pubblica
  - Costo computazionale più basso
  - Costi di comunicazione più alti
  - Basati su assunzione matematiche non completamente dimostrate

Attacchi a protocolli crittografici

Notation

- $E(K : P)$
  - denote the result of encrypting message plaintext P with key K
- A protocol run consists of a sequence of messages between principals and will be described using the standard notation. Principals are generally denoted by A, B and S (for a server). The sequence of messages

  (1) $A \rightarrow B : M_1$

  (2) $B \rightarrow S : M_2$

  (3) $S \rightarrow B : M_3$

  denotes a protocol in which A sends M_1 to B, B then sends M_2 to S who then sends M_3 to B.

Notation

- *We denote a mischievous principal by Z. The notation Z(A) denotes the principal Z acting in the role of A.*

Notations

- A message may have several components; some will be plaintext and some will be encrypted.
- $A \rightarrow B : A, E(K_{ab} : N_a)$

Notations

- A number generated by a principal *A* is denoted by *Na*. Such numbers are intended to be used *only once* for the purposes of the current run of the protocol and are generally termed **nonces**.

Freshness Attacks

- A freshness attack occurs when a message (or message component) from a previous run of a protocol is recorded by an intruder and replayed as a message (component) in the current run of the protocol.

Freshness Attacks

- The classic example of such an attack occurs in the Needham Schroeder conventional (symmetric) key protocol.

$$
\begin{array}{lll}
(1)\ A \rightarrow S & : & A, B, Na \\
(2)\ S \rightarrow A & : & E(Kas : Na, B, Kab, E(Kbs : Kab, A)) \\
(3)\ A \rightarrow B & : & E(Kbs : Kab, A) \\
(4)\ B \rightarrow A & : & E(Kab : Nb) \\
(5)\ A \rightarrow B & : & E(Kab : Nb - 1)
\end{array}
$$

Freshness Attacks

- Although B decrypts this message and (if it is indeed well-formed) assumes legitimately that it was created by the server S, there is nothing in the message to indicate that it was actually created by S as part of the current protocol run. Thus, suppose a previously distributed key K_{ab} has been compromised (for example, by cryptanalysis) and is known to an intruder Z
- He can now fool B into accepting the key as new by the following protocol.

Freshness Attacks

$$
\begin{array}{lll}
(3)\ Z(A) \rightarrow B & : & E(Kbs : K'ab, A) \\
(4)\ B \rightarrow Z(A) & : & E(K'ab : Nb) \\
(5)\ Z(A) \rightarrow B & : & E(K'ab : Nb - 1)
\end{array}
$$

B believes he is following the correct protocol. Z is able to form the correct response in (5) because he knows the compromised key K_{ab}. He can now engage in communication with B using the compromised key and masquerade as A. Denning and Sacco suggested that the problem could be fixed by the inclusion of timestamps in relevant messages.

Type flaws

- A message consists of a sequence of components each with some value (for example, the name of a principal, the value of a nonce, or the value of a key). The message is represented at the concrete level as a sequence of bits.
- A **type flaw** arises when the recipient of a message accepts that message as valid but imposes a different interpretation on the bit sequence than the principal who created it

Type flaws

- consider the Andrew Secure RPC Protocol

$$
\begin{array}{lll}
(1)\ A \rightarrow B & : & A, E(Kab : Na) \\
(2)\ B \rightarrow A & : & E(Kab : Na+1, Nb) \\
(3)\ A \rightarrow B & : & E(Kab : Nb+1) \\
(4)\ B \rightarrow A & : & E(Kab : K'ab, N'b)
\end{array}
$$

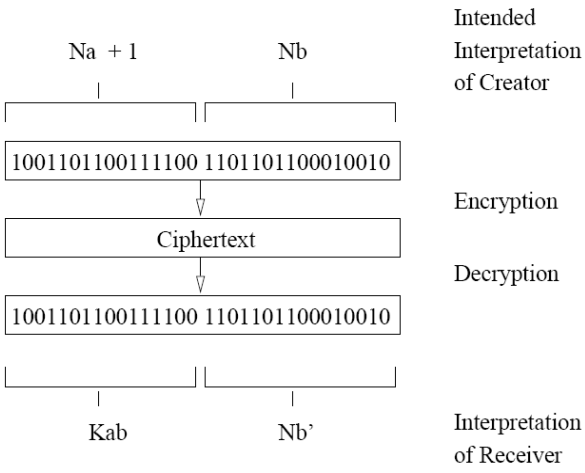
- However, if the nonces and keys are both represented as bit sequences of the same length, say 64 bits, then an intruder could record message (2), intercept message (3) and replay message (2) as message (4). Thus the attack looks like:

Type flaws

$$
\begin{array}{lll}
(1) & A \rightarrow B & : \; A, E(Kab : Na) \\
(2) & B \rightarrow A & : \; E(Kab : Na+1, Nb) \\
(3)\; A \rightarrow Z(B) & & : \; E(Kab : Nb+1) \\
(4)\; Z(B) \rightarrow A & & : \; E(Kab : Na+1, Nb)
\end{array}
$$

- Thus principal *A* may be fooled into accepting the nonce value N_a + 1 as the new session key.

Type Flaws



Type Flaws

- If the nonces are random then the use of the nonce value as a key may not lead to a security compromise but it should be noted that nonces cannot be assumed to be good keys.
- Furthermore, nonces do not necessarily have to be random, just unique to the protocol run. Thus a predictable nonce might be used. In such cases A will have been fooled into accepting a key whose value is known to the intruder.
- The above protocol is awed in other ways too. For example, it is equally possible to record message (4) of a previous run and replay it in the current run, i.e. there is a freshness attack

Type flaws-Question

$$
\begin{array}{lll}
(1)\ A \rightarrow B & : & M, A, B, E(Kas : Na, M, A, B) \\
(2)\ B \rightarrow S & : & M, A, B, E(Kas : Na, M, A, B), E(Kbs : Nb, M, A, B) \\
(3)\ S \rightarrow B & : & M, E(Kas : Na, Kab), E(Kbs : Nb, Kab) \\
(4)\ B \rightarrow A & : & M, E(Kas : Na, Kab)
\end{array}
$$

- Is this protocol subject to a type attack?How?

Type flaws-Answer

$$
\begin{array}{lll}
(1)\ A \rightarrow B & : & M, A, B, E(Kas : Na, M, A, B) \\
(2)\ B \rightarrow S & : & M, A, B, E(Kas : Na, M, A, B), E(Kbs : Nb, M, A, B) \\
(3)\ S \rightarrow B & : & M, E(Kas : Na, Kab), E(Kbs : Nb, Kab) \\
(4)\ B \rightarrow A & : & M, E(Kas : Na, Kab)
\end{array}
$$

- The above protocol causes a key K_{ab} created by the trusted server *S* to be distributed to principals *A* and *B*. *M* is a protocol run identifier.
- After initiating the protocol *A* expects to receive a message back in (4) that contains the nonce N_a he used in message (1) together with a new session key K_{ab} created by *S*. If *M* is (say) 32 bits long, *A* and *B* each 16 bits long and K_{ab} is 64 bits then an intruder *Z* can simply replay the encrypted component of message (1) as the encrypted component of message (4).

Type flaws-Answer

$$
\begin{array}{lll}
(1)\ A \rightarrow Z(B) & : & M, A, B, E(Kas : Na, M, A, B) \\
(4)\ Z(B) \rightarrow A & : & M, E(Kas : Na, M, A, B)
\end{array}
$$

- Here *A* decrypts $E(K_{as} : N_a, M, A, B)$ checks for the presence of the nonce *Na* and accepts (M,A,B) as the new key. M, A and B are all publicly known (since they were broadcast in the clear). It is clear that an intruder can play the role of *S* in messages (3) and (4) simply replaying the encrypted components of message (2) back to *B*. The attack is:

$$
\begin{array}{lll}
(1)\ \ A \rightarrow B & : & M, A, B, E(Kas : Na, M, A, B) \\
(2)\ B \rightarrow Z(S) & : & M, A, B, E(Kas : Na, M, A, B), E(Kbs : Nb, M, A, B) \\
(3)\ Z(S) \rightarrow B & : & M, E(Kas : Na, M, A, B), E(Kbs : Nb, M, A, B) \\
(4)\ \ B \rightarrow A & : & M, E(Kas : Na, M, A, B)
\end{array}
$$

- He can now listen in to conversation between A and B using the now publicly available key (M, A, B)

Parallel Session Attacks

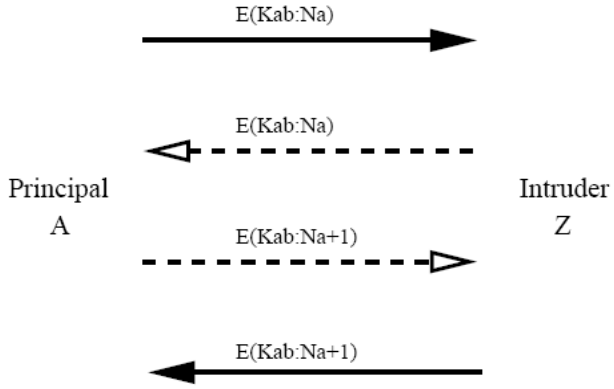
- A parallel session attack occurs when two or more protocol runs are executed concurrently and messages from one are used to form messages in another.

Parallel Session Attacks

- Consider
$$(1)\ A \rightarrow B \quad : \quad E(Kab : Na)$$
$$(2)\ B \rightarrow A \quad : \quad E(Kab : Na + 1)$$
- Successful execution should convince *A* that *B* is operational since only *B* could have formed the appropriate response to the challenge issued in message (1). In addition, the nonce N_a may be used as a shared secret for the purposes of further communication between the two principals.
- In fact, an intruder can play the role of *B* both as responder and initiator.
- The attack works by starting another protocol run in response to the initial challenge.

$$(1.1)\ A \rightarrow Z(B) \quad : \quad E(Kab : Na)$$
$$(2.1)\ Z(B) \rightarrow A \quad : \quad E(Kab : Na)$$
$$(2.2)\ A \rightarrow Z(B) \quad : \quad E(Kab : Na + 1)$$
$$(1.2)\ Z(B) \rightarrow A \quad : \quad E(Kab : Na + 1)$$

Parallel Session Attacks



- Solid arrows indicate messages of the first protocol run, broken arrows indicate messages of the second protocol run.

Parallel Session Attacks

- In the above attack *Z* used principal *A* to do some work on his behalf.
- He needed to form an appropriate response to the encrypted challenge but could not do so himself and so he "posed the question" to *A* who provided the answer. *A* is said to act as an **oracle** (because he always provides the correct answer) and attacks of this form are often called **oracle attacks**.
- An interesting example of an oracle attack occurs in the Wide-Mouthed Frog Protocol

$$(1)\ A \rightarrow S \quad : \quad A, E(Kas : Ta, B, Kab)$$
$$(2)\ S \rightarrow B \quad : \quad E(Kbs : Ts, A, Kab)$$

Parallel Session Attacks

- The first way it can be attacked is by simply replaying the first message within an appropriate time window - this will succeed since *S* will produce a new second message with an updated timestamp. If *S's* notion of timeliness is the same as *B's* (i.e. it accepts messages only if the timestamp is later than that of any other message it has received from the sender) then this attack will not work.
- The second method of attack allows one protocol run to be recorded and then the attacker continuously uses *S* as an oracle until he wants to bring about reauthentication between *A* and *B*.

Parallel Session Attacks

$$
\begin{array}{lrcl}
(1) & A \rightarrow S & : & A, E(Kas : Ta, B, Kab) \\
(2) & S \rightarrow B & : & E(Kbs : Ts, A, Kab) \\
(1') & Z(B) \rightarrow S & : & B, E(Kbs : Ts, A, Kab) \\
(2') & S \rightarrow Z(A) & : & E(Kas : T's, B, Kab) \\
(1'') & Z(A) \rightarrow S & : & A, E(Kas : T's, B, Kab) \\
(2'') & S \rightarrow Z(B) & : & E(Kbs : T''s, A, Kab)
\end{array}
$$

- *Z* now continues in the above fashion until he wishes to get *A* and *B* to accept the key again. He does this by allowing *A* and *B* to receive messages intended for them by *S*.

Parallel Session Attacks

- There is some ambiguity in the available descriptions as to how timestamps are checked. It would seem sensible for a recipient *A* or *B* to impose some type of time window on the timestamps of messages received from *S* (as well as checking the message it has received from *S* is timestamped later than any other it has received from *S*).
- The efficacy of the attack is not compromised. *Z* simply plays ping-pong with *S* until it wants to rearrange authentication between *A* and *B*. Continuous use of *S* as a timestamp oracle ensures that all messages are sufficiently up to date.

Implementation Dependent Attacks

- Some protocol definitions allow both secure and insecure implementations. Typing attacks could be prevented if the concrete representations of component values contained redundancy to identify a sequence of bits as representing a specific value of a specific type (and the principals made appropriate checks). Few protocol descriptions require such enforcement of types explicitly.
- Thus, the implementation approach adopted may severely affect the actual security of a protocol that actually conforms to the description. thus, we have the possibility of implementation-dependent attacks.

Implementation Dependent Attacks

- the naïve use of certain algorithms (that are generally considered *strong*) in the context of specific protocols may produce insecure results.
- RC4-WEP…..

Implementation Dependent Attacks (stream cipher)

- Consider the last two messages of the Needham Schroeder protocol

$$
\begin{aligned}
&(1)\ A \rightarrow S \ : \ A, B, Na \\
&(2)\ S \rightarrow A \ : \ E(Kas : Na, B, Kab, E(Kbs : Kab, A)) \\
&(3)\ A \rightarrow B \ : \ E(Kbs : Kab, A) \\
&(4)\ B \rightarrow A \ : \ E(Kab : Nb) \\
&(5)\ A \rightarrow B \ : \ E(Kab : Nb - 1)
\end{aligned}
$$

- Suppose that the cipherstream for message (4) is $b_1 b_2 \ldots b_{n-1} b_n$.
- If N_b is odd then the final plaintext bit (assumed to be the least significant bit) will be 1 and $N_b + 1$ will differ only in that final bit. On a bit by bit encryption basis, the cipherstream for message (5) can be formed simply by flipping the value of the final bit b_n. On average the nonce will be odd half of the time and so this form of attack has a half chance of succeeding

Binding attacks

- In public key cryptography the integrity of the public keys is paramount.
- Suppose your public key is *Ky* and an intruder's public key is *Ki*. The intruder is able to decrypt any messages encrypted with *Ki*. Principals wishing to convey information to you secretly will encrypt using what they believe is your public key. Thus, if the intruder can convince others that your public key is *Ki* then they will encrypt secret information using *Ki* and this will be readable by the intruder.

.

Binding attacks

- Thus, the principals in charge of distributing public keys must ensure that the above cannot occur; there must be a variable binding between a public key and the corresponding agent. In some authentication protocols, this has not been achieved.
- Consider the following protocol:

.

$$(1)\ C \rightarrow AS \quad : \quad C, S, Nc$$
$$(2)\ AS \rightarrow C \quad : \quad AS, E(Kas^{-1} : AS, C, Nc, Ks)$$

Binding attacks

- The components of the encrypted part signify that the message was created by *AS*, that this message has been created in response to a request from a client *C* with nonce *Nc* and that the public key requested is *Ks*.
- However, note that there is nothing in the encrypted part of message (2) that ensures the recipient that the key is really the public key of *S*. This leads to the following parallel session attack:

$$
\begin{array}{lll}
(1.1)\ C \rightarrow Z(AS) & : & C, S, Nc \\
(2.1)\ Z(C) \rightarrow AS & : & C, Z, Nc \\
(2.2)\ AS \rightarrow Z(C) & : & AS, E(Kas^{-1} : AS, C, Nc, Kz) \\
(1.2)\ Z(AS) \rightarrow C & : & AS, E(Kas^{-1} : AS, C, Nc, Kz)
\end{array}
$$

Binding attacks

- this problem can be solved by explicitly including the identifier of the requested server *S* in message

$$
\begin{array}{lll}
(1)\ C \rightarrow AS & : & C, S, Nc \\
(2)\ AS \rightarrow C & : & E(Kas^{-1} : AS, C, Nc, S, Ks)
\end{array}
$$

Secret Sharing Algorithms

What is secret sharing?

- In cryptography, **secret sharing** refers to a method for distributing a *secret* amongst a group of participants, each of which is allocated a *share* of the secret.
- The secret can only be reconstructed when the shares are combined together; individual shares are of no use on their own.

Why do we need secret sharing?

- Gives tight control and removes single point vulnerability.
- Individual key share holder cannot change/access the data.

Mathematical Definition

- Goal is to divide some data *D* (e.g., the safe combination) into n pieces D1,D2….Dn in such a way that:
  - Knowledge of any k or more D pieces makes D easily computable.
  - Knowledge of any k -1 or fewer pieces leaves D completely undetermined (in the sense that all its possible values are equally likely).
- This scheme is called (k,n) threshold scheme. If k=n then all participants are required together to reconstruct the secret.

Shamir's Secret Sharing

- Suppose we want to use (k,n) threshold scheme to share our secret S where k < n.
- Choose at random (k-1) coefficients a1,a2,a3…ak-1 , and let S be the a0

$$f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{k-1}^{k-1}$$

Shamir's Secret Sharing

- Construct n points (i,f(i)) where i=1,2…..n
- Given any subset of k of these pairs, we can find the coefficients of the polynomial by interpolation, and then evaluate a0=S , which is the secret.

Example

- Let S=1234
- n=6 and k=3 and obtain random integers
  a1=166 and a2=94

$$f(x) = 1234 + 166\,x + 94\,x^2$$

- Secret share points
  (1,1494),(2,1942)(3,2598)(4,3402)(5,4414)(6,5614)

- We give each participant a different single point (both x and f(x)).

Reconstruction

- In order to reconstruct the secret any 3 points will be enough
- Let us consider

$(x_0, y_0) = (2,1924), (x_1, y_1) = (4,3402), (x_2, y_2) = (5,4414)$

Using Lagrange polynomials

$l_0 = x - x_1 / x_0 - x_1 * x - x_2 / x_0 - x_2 = x - 4/2 - 4 * x - 5/2 - 5 = 1/6x^2 - 11/2x + 31/3$

$l_1 = x - x_0 / x_1 - x_0 * x - x_2 / x_1 - x_2 = x - 2/4 - 2 * x - 5/4 - 5 = -1/2x^2 - 31/2x - 5$

$l_2 = x - x_0 / x_2 - x_0 * x - x_1 / x_2 - x_1 = x - 2/5 - 2 * x - 4/5 - 4 = 1/3x^2 - 2x + 22/3$

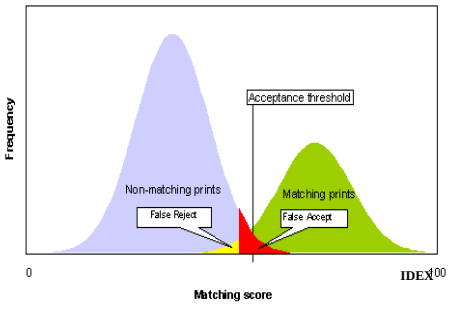
$$f(x) = \sum_{j=0}^{2} y_j l_j(x) = 1942(1/6x^2 - 11/2x + 31/3) + 3402(-1/2x^2 - 31/2x - 5) + 4414(1/3x^2 - 2x + 22/3)$$

$f(x) = 1234 + 166x + 94x^2$

Biometrics

Biometrics

- Biometrics – measuring physical characteristics that provide certain uniqueness
  - “something you are”, hence hard to impersonate
- Examples:
  - Fingerprints
  - Retinal / Iris scanning
  - Voice recognition
  - Signature/typing dynami
  - Face / hand recognition
- Applications
  - Identification: many-to-1
  - Verification: 1-to-1
- Pattern recognition errors
  - Misidentification (FN, Type 1)
  - False alarms (FP, Type 2)



Biometrics

- Biometrics can be used for less constrained problem of automatic identification of individuals.
- In this mode, biometric system carries out a “one-to-many” search of its stored models of individuals' identities.
- As a result, biometrics can be used in security applications, such as fraud and/or intrusion detection

Fingerprints Analysis

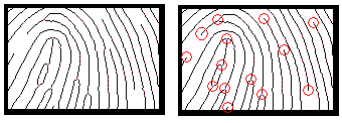
Shapes:

ARCH WHORL LOOP

Minutiae:

END BIFURCATION ISLAND LAKE DOT

FBI found that no 2 people have > 8 common minutiae points.



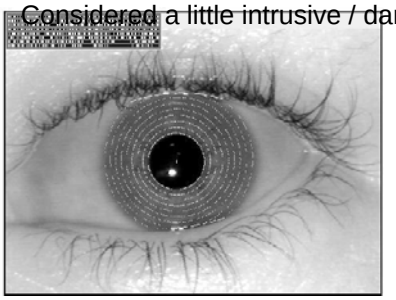
- Non-intrusive, Reliable, Inexpensive
- Semiconductor or Optical
- Useful mostly for verification and less for identificati

Hand Geometry

- One of the first practically implemented techniques
  - physical access control: airports, secured corporate areas, etc.
  - time and attendance monitoring
- Reader uses CCD camera and a number of mirrors to measure the shape of the hand perimeter, in <1 sec
  - Length, width, thickness, surface areas
- Used for verification, in conjunction with another identifier
- Non-intrusive

Iris Scanning

- Human eye encodes 3.4bits/sqmm
- Extremely accurate: chance of duplication (including twins) < 10^{-78}
- Fast comparison: Identification takes 2sec per 100,000 people in DB
- Sub-$1000 systems are available
- Considered a little intrusive / dangerous by som



Retinal Scanning

- Works by identifying patterns in retinal blood vessels
- Uses light source to take 400 measurements, which are then reduced to a signature of 96 bytes
- Preceded Iris scanning, but is less prevalent
  - considered more intrusive
  - requires precise positioning of the eye
  - requires removal of glasses

Facial Recognition

- Controlled scene – access control
  - Frontal view, similar distance, reasonable lighting
  - Compare live image to an original, captured in similar environment
  - Usually for verification purposes, with another ID
- Algorithms extract features, and compare relative positions of eyes, nose, and mouth
- Relatively user-friendly
- Not very accurate, and requires frequent updat

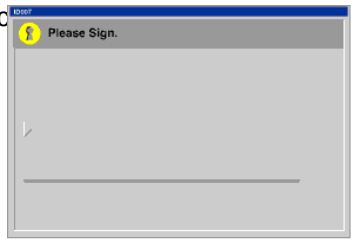
- Random scene – street, airports
  - Much more difficult
  - Law enforcement applications
  - Privacy issues

Voice Verification

- Principle: speech dynamics are affected by physical structure of mouth, vocal chords, sinus, etc.
- A voice signature can typically be formed from speech features, with very high accuracy
  - Each syllable typically has few dominant frequencies (formants)
  - More accurate when user repeats a previously recorded sentence
- Weaknesses: taped replay, environmental noise, illness, richness of spoken language
- Applications: access control, call centers

Signature Verification

- Static verification
- Dynamic verification
  - Curvature, changes in x-y sign, ac



Biometrics Characteristics

Type	Accuracy	Ease Use	Acceptance
Fingerprint	High	Medium	Low
Hand Geom	Medium	High	Medium
Voice	Medium	High	High
Retina	High	Low	Low
Iris	Medium	Medium	Medium
Signature	Medium	Medium	High
Face	Low	High	High

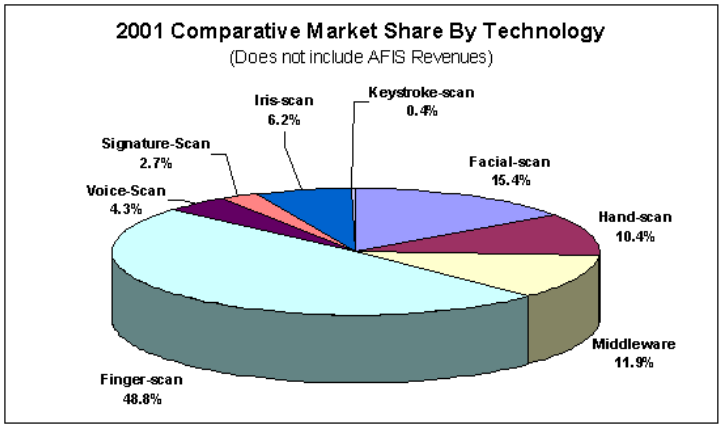
Weaknesses of Biometrics

- False positive rate
- In identification applications: misidentification
- Replay attack, e.g., cut finger, tape replay

- Health concerns
- Privacy concerns

Biometric Market

- Int'l Biometric Group



Rising interests in Biometrics

- September 11 created rising interest in biometrics: number of public/private organizations engaged, quality and number of products produced and deployed.
- Officials are "beefing-up" security at airports and other key installations by implementing biometric tools.
- Interest fueled by rise in hacker activities. September 11 identity theft, for example, has heightened awareness

qui

Behavioral Security

The 5th Factor: How you behave

- Idea: the way user behaves may help identify, or authenticate him/her
- For example
  - What time of the day you access a certain application?
  - At what frequency do you perform a certain operation
  - What type of access to which information you require?
  - Did you login from home or work?
- Premise for authentication: a user's behavioral pattern changes only slowly over time.
- Advantage: relatively cheap, through software
- Typically, to be used in conjunction with another factor
  - e.g., use behavior profiling to supplement password authentication
- Acceptance to this new form will grow, especially in areas like intrusion detection…

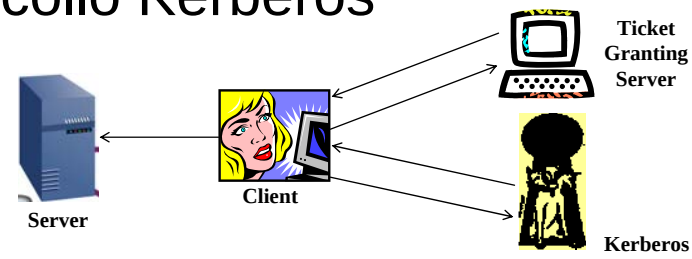
Autenticazione Client/Server

Kerberos

Kerberos

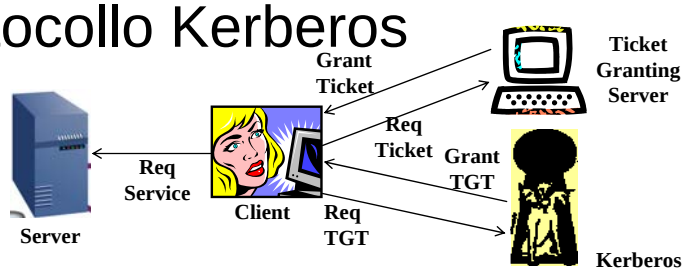
- Servizio di autenticazione Client / Server
  - Utilizzato come un servizio di rete che permette ad utenti e server di autenticarsi reciprocamente
  - Usa chiavi simmetriche convenzionali come prova dell'identità (DES)
  - Sviluppato al MIT dal Progetto Athena.
- Tipi di attacco per cui è stato sviluppato
  - Impersonificazione di altri utenti
  - Alterazione dell'identità di un dispositivo
  - Attacchi *replay*
- Requisiti
  - Sicurezza:
    - *eavesdropper* non possono ottenere sufficienti informazioni
    - Kerberos stesso dovrebbe essere sicuro
  - Affidabilità ed alta disponibilità
  - Trasparenza per l'utente

Protocollo Kerberos



- Kerberos condivide diverse chiavi segrete con ogni entita` nella rete
- La conoscenza della chiave segreta equivale ad una dimostrazione di identita`
- Non vogliamo mandare chiave segreta in chiaro…
- Clients sono utenti ma anche applicazioni software

Protocollo Kerberos

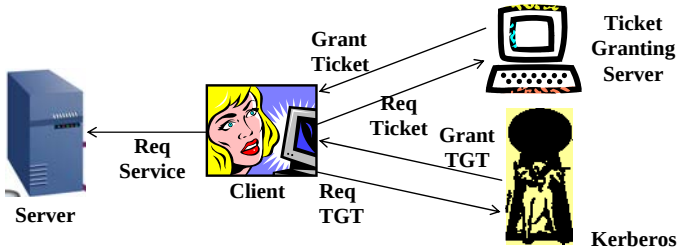


Due tipi di credenziali:

- Ticket: Valido per singolo cliente e singolo server
  $T(c,s) = s, E_{Ks}(c, a, v, K_{c,s})$
  - c-client, s-server, a-indirizzo client, v-tempo di validità, $K_{c,s}$ chiave di sessione. Usato come “passi” fino alla scadenza

  Es: TGT (Ticket-Grant Ticket): chiave di sessione tra Client e TGS – ci possono essere molti TGServers
- Autenticatore: Generato ogni volta che si vuole accedere a un servizio $A(c,s) = E_{Kc,S}(c, t, k)$
  - t-time stamp, k-chiave di sessione addizionale
  - Usato una sola volta, ma client può generarne quanti vuole

Kerberos Protocol



- Req TGT: Send c, tgs
- Grant TGT: Gen $K_{c,tgs}$; Send $E_{Kc}(K_{c,tgs})$, $E_{Ktgs}(T(c, tgs))$
- Req Ticket: Gen A(c,tgs); Send $E_{Kc,tgs}(A(c,tgs))$, $E_{Ktgs}(T(c,tgs))$, s
- Grant Ticket: Gen $K_{c,s}$; Send $E_{Kc,tgs}(K_{c,s})$, $E_{Ks}(T(c, s))$
- Req Service: Gen A(c,s); Send $E_{Kc,s}(A(c, s))$, $E_{Ks}(T(c, s))$

$A(c,s) = E_{Kc,s}(c, t, k)$ $\quad T(c,s) = s, E_{Ks}(c, a, v, K_{c,s})$

Altre caratteristiche di Kerberos

- Replicazione di Kerberos
  - In grandi organizzazioni, è possibile replicare il TGT/Ss, con una copia che svolge il ruolo di *master* e le altre in modalità read-only.
- *Realms (domini?)*
  - Normalmente i servizi di rete sono divisi in gruppi gestiti da server Kerberos distinti
  - E' possibile stabilire una relazione di reciproca "fiducia" tra due *realm* definendo il Kerberos TGS come un server nell'altro realm

Caratteristiche di sicurezza di Kerberos

- Kerberos verifica l'identità del cliente attraverso la chiave e confronta identità ed indirizzo con quelli contenuti in un database.
- Il ticket T(c,tgs/s) è dato al client ma è "bloccato".
- Il server verifica il client attraverso la chiave di sessione nell'autenticatore.
- Timestamp sono usati per assicurare la sincronicità e per limitare la validità del ticket originale (tipicamente 8 ore).
- Con una semplice aggiunta il client può verificare il server.

Attacchi a Kerberos

- Kerberos stesso immagazzina molte chiavi e deve risiedere su un sistema altamente protetto.
- Tickets e Autenticatori potrebbero essere "riciclati" nell'intervallo di vita a loro concesso (8 ore!). Il server dovrebbe conservare le richieste recenti e controllare possibili "replay". Tipicamente non lo fa.
- Kerberos TGT protetto da DES encryption. Attacchi basati su passwd-guessing.
- L'avversario potrebbe intercettare molti TGTs e cercare di decriptarli offline. I client ***devono comunque*** usare password sicure!
- Malicious sw: rimpiazzare clienti Kerberos cosi' da memorizzare passwd e chiavi.
- Modificando il clock del server l'avversario potrebbe riciclare i ticket. Hosts devono sincronizzare i clock frequentemente. Gestione di network time protocols altamente insicuro…
- Kerberos tipicamente non dotato di crittografia a chiave pubblica. Lo sara` + Gestione delle chiavi basata su smart card

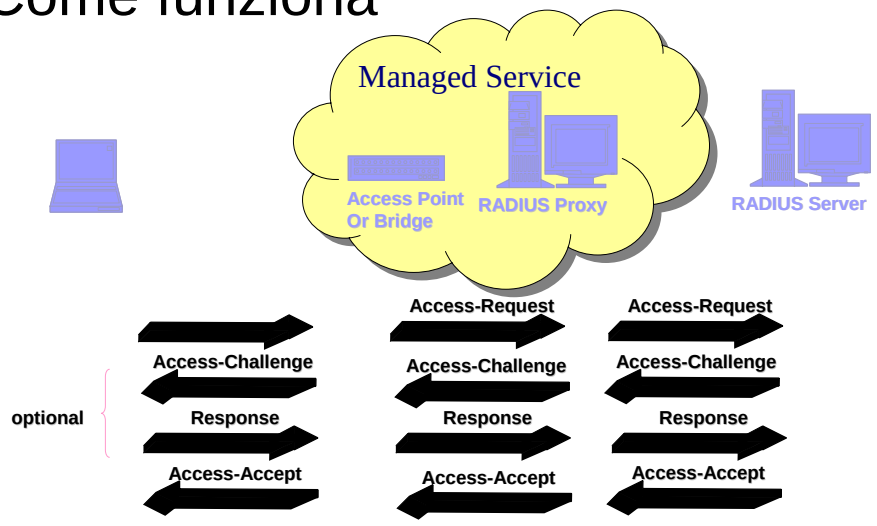
Servizio di autenticazione di utenti remoti

RADIUS

RADIUS

- Remote Authentication Dial In User Service
  - Originariamente sviluppato per l'accesso remoto su linea telefonica
- Protocollo di rete client/server ampiamente utilizzato
  - Implementato nel *transport layer* (usa UDP)
  - I client sono tutti i tipi di Network Access Servers (NAS)
  - Fornisce le 3A (autenticazione, autorizzazione, accounting)
  - Esempio: NT4.0 IAS
- Supporta utenti remoti e *mobile*
  - porte fisiche (modem, DSL, wireless)
  - porte virtuali (extranets, VPNs)
- Permette un controllo centralizzato/remoto e l'accounting.
- Il Proxy RADIUS permette l'autenticazione distribuita.

Come funziona



Meccanismi di sicurezza di RADIUS

- Il client ed il server RADIUS condividono un segreto (normalmente inserito come una stringa di caratteri, cioè una password).
- Ogni richiesta riceve un autenticatore (*nonce*).
- I messaggi sono criptati usando uno cifratore di flusso (*stream cipher*), generato applicando MD5 al segreto ed all'autenticatore
  - Viene fatto lo XOR del testo in chiaro (utente e password) con lo *stream*
  - Si utilizza lo stile *Chained CBC* se la password è troppo lunga
- Alcune debolezze sono state riportate
  - MD5 non è stato sviluppato per essere un cifratore di flusso
  - Facendo lo XOR di testi cifrati un malintenzionato ottiene lo XOR dei due testi in chiaro; se le password hanno lunghezza diversa, il suffisso della più lunga appare come testo in chiaro
  - Allo stesso modo è possibile realizzare offline un attacco al segreto condiviso
- Tra le possibili soluzioni è stato proposto l'uso della cifratura simmetrica.
- Ancora meglio, lo scambio richiesto da RADIUS può essere criptato con IPSec.

Sicurezza Web

Secure Socket Layer (SSL)

Transport Layer Security (TLS)

Sicurezza dell'accesso Web

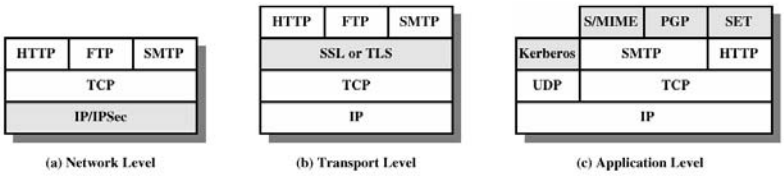
- In teoria, l'accesso al Web è una semplice interazione client-server
  - Protocolli tipo Kerberos potrebbero essere applicati
- Caratteristiche speciali dell'accesso al Web
  - I Web server sono spesso "esposti", accessibili a chiunque
  - Spesso i Web server devono essere connessi a database "critici", e possono essere pericolosi se compromessi
  - Il software applicativo è sviluppato velocemente per i web server, senza tener conto, molto spesso, del problema della sicurezza
  - Gli utenti del Web spesso non sono soggetti alle regole "*corporate*"
  - Gli utenti del Web sono spesso del tutto sconosciuti
  - Non si può contare sul fatto che gli utenti del Web "svolgano con diligenza" il loro compito in un protocollo di sicurezza.

Minacce alla sicurezza del Web

- Integrità ⇐ System Security ⇐ Communication Security
  - Corruzione dei dati sui server
  - Corruzione dei messaggi
- Confidenzialità ⇐ System Security ⇐ Communication Security
  - Furto dei dati da un server o da un client
  - Intercettazioni della comunicazione ⇐ System Security
  - Info sulla configurazione di rete ⇐ Communication Security
  - Info sul traffico di rete
- Interruzione ⇐ System Security
  - Denial of Service e DDOS
- Autenticazione ⇐ Communication Security ⇐ System Security
  - Impersonificazione di utenti legittimi
  - Creazione di dati “fasulli” (server o client)

Alternative possibili per la sicurezza del Web

- Livello network: IPSec
- Livello applicativo
- Protocolli SSL/TLS
  - Come protocollo “on top” di TCP nei livelli trasporto e sessione
  - Come parte del software applicativo: browser sul lato client e web server (SSL è stato sviluppato da Netscape)



Secure Socket Layer (SSL)

- Sviluppato da Netscape come parte del browser
  - SSLv3 è stato sottoposto ad un processo di revisione pubblica
  - Transport Layer Security (TLS) è stato disegnato come successore a SSLv3
- SSL è un protocollo a livello sessione, in cui ogni sessione può consistere di più connessioni.
- SSL presenta due livelli
  - SSL Record Protocol fornisce i servizi di sicurezza di base, *e.g.* https
  - Il protocollo di Handshake è usato per iniziare le sessioni
  - Il protocollo di Alert per il "*peer messaging*"

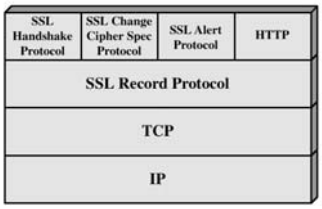
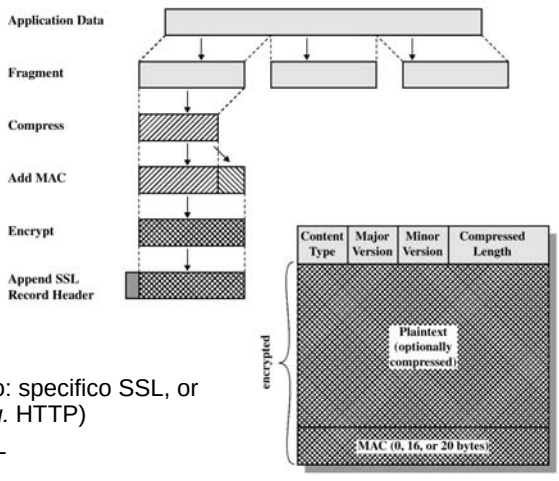


Figure 7.2 SSL Protocol Stack

- Stati della sessione SSL:
  - Algoritmi di sicurezza
  - Chiavi "*Master*"
  - Metodi di compressione
  - Certificati

SSL Record Protocol

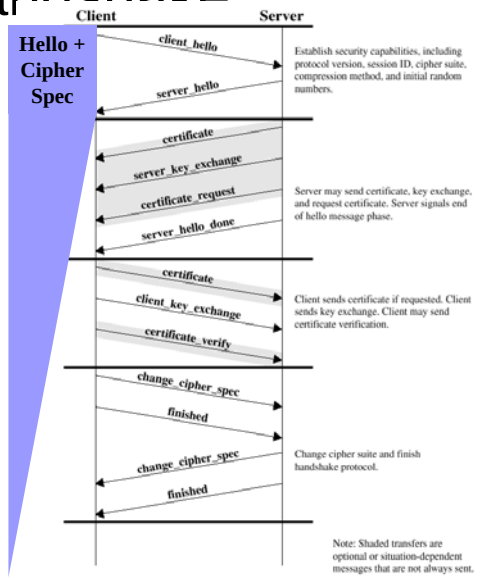
- Servizi: Confidenzialità, Integrità dei Messaggi
- Più algoritmi sono permessi
- HMAC è quello standard



- Header:
  - Tipo di contenuto: specifico SSL, or applicazione (*e.g.* HTTP)
  - Versione del SSL

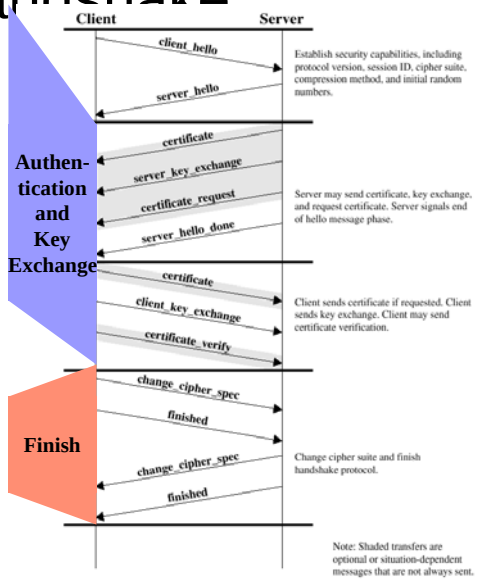
Protocollo di Handshake

- Il client suggerisce; il server sceglie
- Versione SSL: viene utilizzata la versione più bassa
- *Nonce*: timestamp+random
- Session ID: esistente o nuova
- Alternative, in ordine di preferenza decrescente, per
  - Scambio delle chiavi
  - Algoritmi di cifratura
  - Algoritmi per il MAC
  - Parametri
- Metodi di compressione supportati



Protocollo di Handshake

- Parte il server; quindi il client
- Il server invia il certificato
- Il server invia il messaggio per lo scambio delle chiavi
- Il server può richiedere un certificato al client
- Il client risponde
- Lo stato passa a quello di attesa per il cambio di cifratura
- L'*handshake* è concluso



Attacks to SSL

Using only SSLv2

- Downgrade attack: attacker who can change bits (SSLv2 handshake packets are not integrity checked) can force SSLv2 session to use weaker ciphersuites. Next, brute-force techniques crack small key
- Truncation attack: stop SSLv2 session without server/client knowing it was stop by attacker. If attacker knows format of message and how sent in SSLv2 packets, can change meaning of message

Attacks to SSL

- User assume SSL flawless, correctly implemented in his/her browser, CA performs proper checks before issuing SSL certificates…?
- Not difficult for virus, trojan, worm, malicious Web site, malicious email, etc… to inject a malicious SSL certificate to victim's list of trusted root CA's! Vulnerabilities in Internet Explorer, Windows Media Player, SirCam worm…
- Restrict access to list of trusted root CA's:
  - Windows 98 does not support file and registry permission…
- Default installation of Windows 98 2E,

Attacks to OpenSSL

Remotely exploitable buffer overflows:

- SSLv2 handshake… (malformed key)
- SSLv3 handshake… (large session ID + malformed key)
- Integer overflows (if running on 64-bit platforms)

  Execute arbitrary code on server/client or cause denial of service

ASN.1 Coding errors:

- Malformed certificates parsed incorrectly: remote DOS.

  Denial of Service

SSL/TLS Man-in-the-Middle

- Warning if any of following not true:
  1. Certificate signed by recognized CA
  2. Certificate valid and not expired
  3. Common name on certificate matches DNS name of server
- SSL Web client authenticates server via challenge
- If server responds, it has private key to certificate
- If any of 1-3 not true, client presents user dialogue:

  Information you exchange with this site cannot be viewed or changed by others. However, there is a problem with certificate:
  - Certificate issued by company you have not chosen to trust
  - Certificate expired or not yet valid
  - Name on certificate does not match name of site

  Do you want to proceed? Yes, [No]

- Information you exchange with this site can be viewed or changed by others!

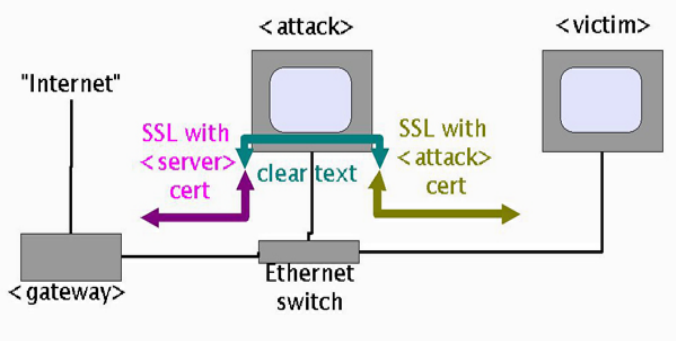
SSL/TLS Man-in-the-Middle

1. Using ARP Poisoning:
   - Insert “attack” host into network traffic between “victim” and “server” (ARP Poisoning)
2. At this point can sniff any traffic from “victim”.

   Didn’t break SSL encrypted traffic yet…
3. Using dsniff (dnsspoof)
   - “attack” masquerades as “victim” ’s destination website – use its certificates to encrypt between “attack” and “victim”
   - “attack” can proxy traffic between “victim” and destination website

This leaves traffic in cleartext to the attacker…

SSL/TLS Man-in-the-Middle



- This is on SSL working properly
- Faulty SSL client can make the attack easier and less risky…

Attacchi a protocolli crittografici-FINE

Gianluigi Me