# Cryptography: Authentication

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Corso di Sicurezza nelle reti, a.a. 2009/2010

http://www.tlc.unipr.it/veltri



## Alcune possibili minacce di una comunicazione dati

- Violazione confidenzialità
  - **accesso ai contenuti dei messaggi da parte di persone o processi non autorizzati**
  - **analisi del traffico**
    - individuazione di schemi di traffico in base a frequenza e durata della conversazione, diensione dei messaggi, etc.
- Tampering
  - **modifica dei contenuti**
    - alterazione dei contenuti dei messaggi (inserimento, cancellazione, modifica)
- Spoofing
  - **falsificazione del mittente**
    - inserimento di messaggi provenienti da una sorgente fasulla
- Replay/Reflection
  - **ritardo o ripetizione dei messaggi**
  - **modifica della sequenza dei messaggi**
  - **modifica del destinatario dei messaggi**
- Repudiation
  - **Ripudio dell'origine**
    - l'origine nega di aver inviato un messaggio
  - **Ripudio della destinazione**
    - la destinazione nega di aver ricevuto un messaggio





## Alcune contromisure

- La prima minaccia riguarda la segretezza dei messaggi
  - **Contromisure: crittografia, mascheramento del traffico**
- La seconda minaccia riguarda l'integrità dei messaggi
  - **Contromisure: Message Integrity Check (MIC)**
- Le altre minacce riguardano in modo diverso l'autenticità dei messaggi, dell'origine, o della destinazione
  - **Contromisure: MIC, Message Authentication Code (MAC), firma digitale**



## Message authentication

(data origin authentication, integrity check)

# Message Authentication

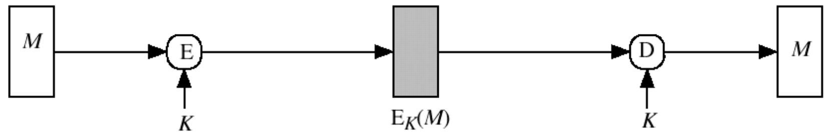
- Message authentication is concerned with:
  - **protecting the integrity of a message**
  - **validating identity of originator**
  - **non-repudiation of origin (dispute resolution)**
- Three alternative functions used:
  - **secret or public key encryption algorithms**
  - **secret + hash functions**
  - **secret + ad-hoc Message Authentication Code (MAC) functions**





# Secret-key Encryption

- Symmetric encryption:
  - **encryption provides both confidentiality and origin authentication**
  - **however, need to recognize corrupted messages (MIC)**

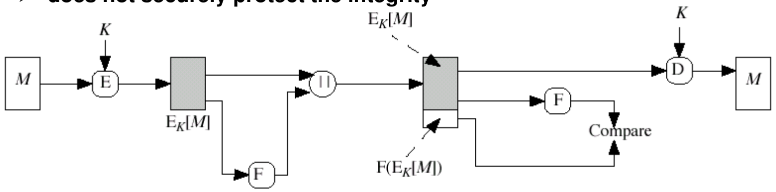




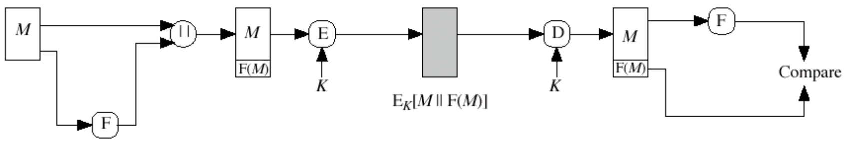



# Secret-key Encryption + Hash

- External error control (checksum):
  - **does not securely protect the integrity**



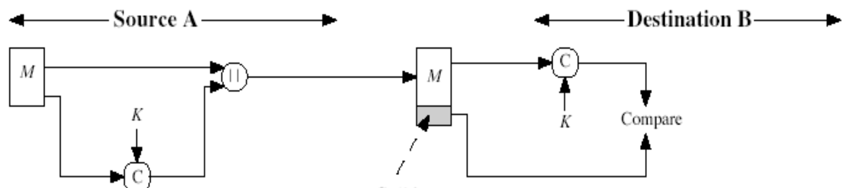

- Message Integrity Check (MIC):
  - **example through internal error control - Manipulation Detection Code (MDC)**








# Message Authentication Code (MAC)




(a) Message authentication

# Message Authentication Code (MAC)

- a MAC is a cryptographic checksum, generated by an algorithm that creates a small fixed-sized block
  - **depending on both message and a secret key K**
    - $MAC = C_K(M)$
  - **condenses a variable-length message M to a fixed-sized authenticator**
    - it need not be reversible
    - is a many-to-one function
      - potentially many messages have same MAC
      - but finding these needs to be very difficult
- appended to message as a **signature**
- receiver performs same computation on message and checks it matches the MAC
- provides assurance that message is unaltered and comes from sender





# Message Authentication Code (cont.)

- In case secrecy is also required
  - **use of encryption with separate key**
  - **can compute MAC either before or after encryption**
  - **is generally regarded as better done before**
- why use a MAC?
  - **sometimes only authentication is needed**
  - **sometimes need authentication to persist longer than the encryption (eg. archival use)**
- MAC is similar but not equal to digital signature





# Requirements for MACs

- MAC functions have to satisfy the following requirements:
  - **knowing a message and MAC, is infeasible to find another message with same MAC**
  - **MACs should be uniformly distributed**
  - **MAC should depend equally on all bits of the message**





# Using Symmetric Ciphers for MACs

- Can use any block cipher chaining mode and use final block as a MAC
- Data Authentication Algorithm (DAA) is a widely used MAC based on DES-CBC
  - **using IV=0 and zero-pad of final block**
  - **encrypt message using DES in CBC mode**
  - **and send just the final block as the MAC**
    - or the leftmost M bits of final block
- But final MAC is now too small for security (≤64bit)

# MAC Security

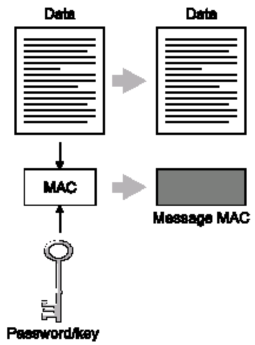
- **cryptanalytic attacks**
  - **like block ciphers, brute-force attacks are the best alternative**





# Hash Message Authenication Code (H-MAC)

- H-MAC (RFC2104)
  è l'applicazione di una funzione di hash in combinazione con una chiave segreta: solo chi possiede la chiave può generare l'hash








# HMAC

- Specified as Internet standard RFC2104
- Uses hash function on the message:

  $HMAC_K$ = Hash[(K$^+$ XOR opad) || Hash[(K$^+$ XOR ipad)|| M)]]

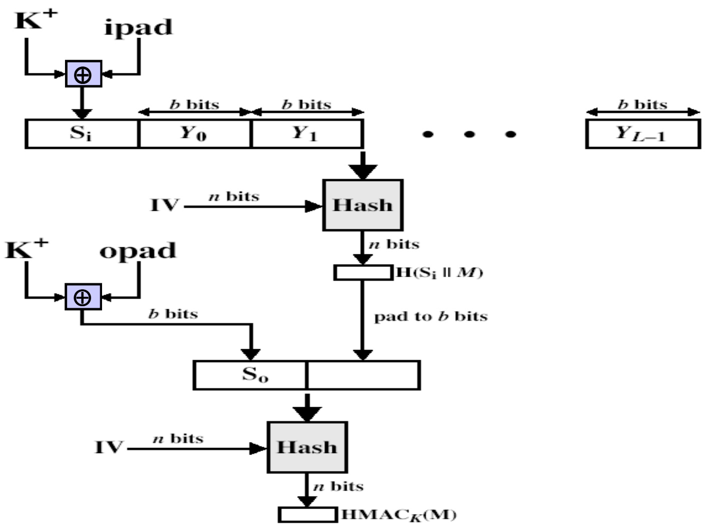
  **where K$^+$ is the key padded out to size**
  **and opad, ipad are specified padding constants**

- Overhead is just 3 more hash calculations than the message needs alone
- Any of MD5, SHA-1, RIPEMD-160 can be used





# HMAC



# Public-key Encryption

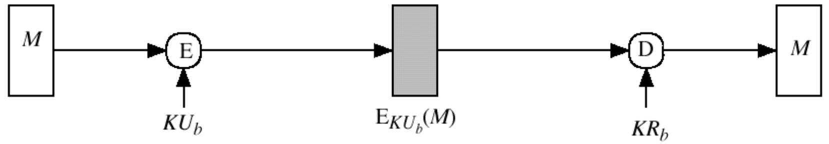
- if public-key encryption is used
  - **encryption with public key provides no proof of sender (no sender authentication)**
    - since anyone potentially knows public-key
  - **both secrecy and authentication if**
    - sender "signs" message using their private-key
    - then encrypts with recipients public key
  - **problems**
    - the result is the same cost of two public-key encryption
    - need to recognize corrupted messages for integrity check



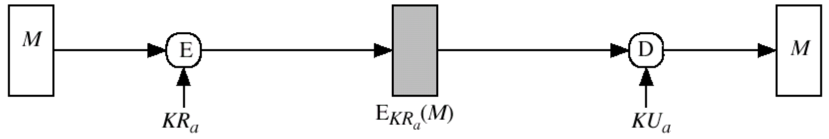


# Public-key Encryption (cont.)

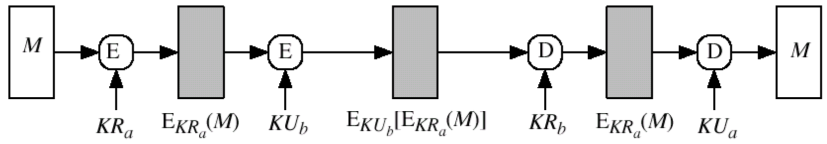
Public-key encryption: confidentiality




Public-key encryption: authentication/signature




Public-key encryption: confidentiality + authentication/signature






# Peer entity authentication



# Autenticazione

- User to host
  - verifica dell'identità di utente che accede ad una risorsa/computer…
- Host to host
  - …si occupa della verifica dell'identità dei sistemi di computer…
- User to user
  - …si dà prova dell'identità di un utente ad un altro utente…
- Identificazione personale

# Secrets-based Authentication

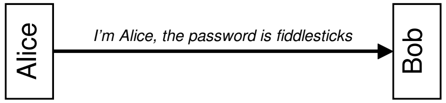
- Secrets-based Authentication
  - **"Its not who you are. It's what you know"**
- Basic system uses passwords
  - **Can be easily intercepted**
- Password protection
  - **encrypt/hash the password**
    - The encrypted/hashed form can still be intercepted
  - **modify the encryption/hashing so the encrypted/hashed value changes each time (challenge/response mechanism, one-time password, etc)**





# Password-based authentication

- Main problem: eavesdropping




- On-line password guessing
  - **direct password search**

    defense/trick:
    - maximum number of attempts
    - slow down
- Off-line password guessing
  - **the intruder captures a quantity derived by a passwd**
    - e.g. a challenge response, or a hash within a database
  - **off-line passwd search with arbitrary amount of power**
  - **sometimes referred as dictionary attack**





# Storing passwords

- Several possibilities:
  - **user passwd individually stored into each host**
  - **host retrieve the passwd from one location (authentication storage node)**
  - **host send user's information to a authentication facilitator node (Authentication Server) that performs authentication and tells the response (e.g. yes/no)**
    - *"Putt all your eggs in one basket,and then watch that basket very carefully."*
- Last two cases require a security association between the host and the authentication node
- Passwords can be stored
  - **encrypted**
  - **hashed**





# Password and Cryptographic keys

- Converting (string) password into cryptographic keys
  - **e.g. DES secret key obtained as hash of the passwd**
- Sometimes, conversion can be more tricky (and computationally expensive)
  - **due to key properties**
  - **e.g. RSA private keys**

# Authentication attacks

- There are many variations of authentication protocols but it 's very hard to get right
- Possible authentication attacks are:
  - **Impersonation attacks (pretend to be client or server)**
  - **Reflection attacks (re-send the authentication messages elsewhere)**
  - **Replay attacks (a valid message is copied and later resent)**
  - **Steal client/server authentication database**
  - **Modify messages between client and server**





# Replay and reflection

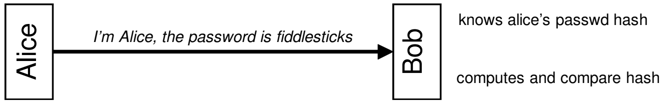
- Countermeasures against replay and reflection attacks include
  - **use of sequence numbers**
    - generally impractical
  - **timestamps**
    - needs synchronized clocks
  - **challenge/response**
    - using unique nonce, salt, realm values



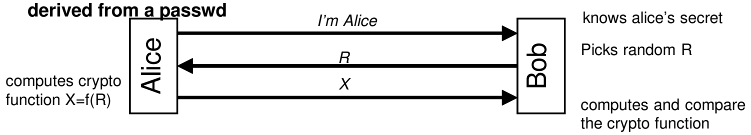


# Eavesdropping and server database reading

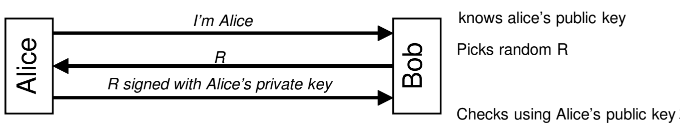
- Protection against server database reading:
  - **vulnerable to eavesdropping**




- Protection against eavesdropping:
  - **vulnerable to database reading, and to offline password guessing if the secret (key) is derived from a passwd**



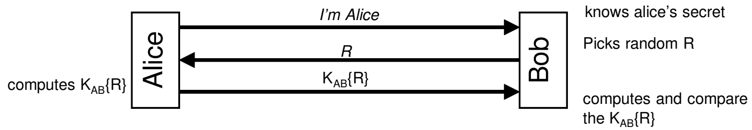

- Protection against both using asymmetric cryptography:








# Authentication with shared secret




- drawbacks:
  - **authentication is not mutual**
  - **an eavesdropper could mount an off-line password guessing attack**
  - **some who read the Bob's passwd-database can later inpersonate Alice**

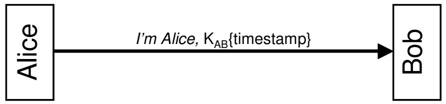
# Authentication with shared secret (variant 1)


- differences:
  - **requires reversible cryptography**
  - **if R is a recognizable quantity, Carol can mount an offline passwd-guessing attack without eavesdropping**
  - **if R is a recognizable quantity with limited lifetime (e.g. a random number concatenated with a timestamp), Alice can authenticate Bob**

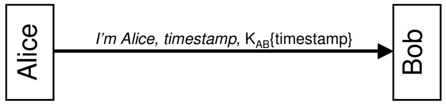




# Authentication with shared secret (variant 2)



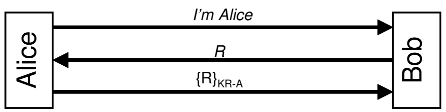

- differences:
  - **this mechanism can be added very easily to a protocol designed for cleartext passwd sending**
  - **more efficent**
  - **several pitfalls due to the time validity (time synchronization between Alice and Bob, authentication with multiple server with the same passwd, etc)**
- variant:



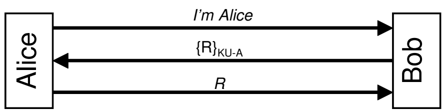





# Authentication with private/public key




or




- property:
  - **the database at Bob is no-longer security-sensitive (must be protected for unauthorized modification, but not from reading)**
- drawback:
  - **if you can trick Alice into signing something, you can impersonate Alice**
- contromisure:
  - **general rule, not use the same key for two different purpose unless the design for all uses are coordinated**
  - **e.g. impose enough structure to be signed (nounce, realm, timestamp, etc.)**



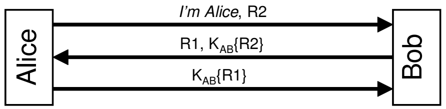


# Mutual authentication with shared secret

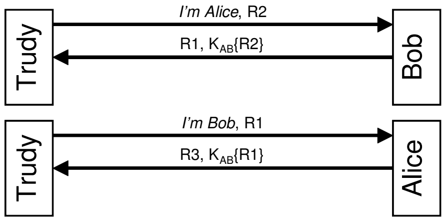

- or shorter..

# Mutual authentication with shared secret

or shorter..




- but:
  - **Reflection attack**

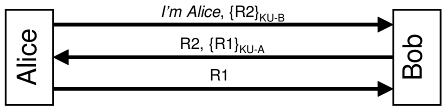



Good general principle of security protocol:
the initiator should be the first to prove its identity





# Mutual authentication with public key




- issues:
  - **how obtaining public key of the peer-entity**
  - **how storing public key of the peer-entity**
  - **how storing own private key**





# One-time passwords

- Static passwords
  - **il "supplicant" e l'"authenticator" sono sicronizzati su una password che non cambia nel tempo**
- One-time passwords
  - **Password generate algoritmicamente ognuna delle quali sarà utilizzabile una sola volta**
    - S-Key (rfc1760)
  - **Smart/token Cards**
    - Applicazioni hardware di sistemi one time password

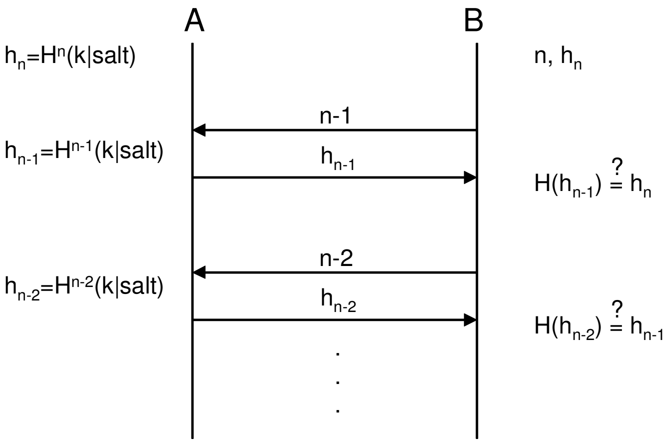





# SKey

- **Sistema per la generazione di password dinamiche**
- **Al login, all'utente viene inviato un seme per la generazione della password**
- **L'utente esegue localmente (es. sul suo host) la generazione della password (in funzione del seme inviato) e la comunica al server**
- **Il server confronta quanto ricevuto con la propria password e, se vi è coincidenza, autentifica l'utente**
- **La cattura della password non permette successivi accessi**

# SKey




Nota: il valore 'salt' permette di riusare la stessa chiave/passwd su sistemi differenti





# Esempio SKey

```
>telnet 193.205.102.131
Trying 193.205.102.131 ...
Connected to 193.205.102.131.
Escape character is '^]'.
Servizio TELNET - Firewall
.........................
Inizio sessione:
CheckPoint FireWall-1 authenticated Telnet server
Login: user_DTCB
SKEY CHALLENGE: 98 user_DTCB
Enter SKEY string: GIST ADA OILY TUNA FRAY RENT
User user_DTCB authenticated by S/Key system.
```