



UNIVERSITA' DEGLI STUDI DI PARMA
Dipartimento di Ingegneria dell'Informazione

Cryptography: Authentication

Luca Veltri

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

Corso di Sicurezza nelle reti di telecomunicazioni, a.a. 2004/2005

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

Message authentication
(data origin authentication, integrity check)

Possibili minacce

1. Violazione
 - accesso ai contenuti dei messaggi da parte di persone o processi non autorizzati
2. Analisi del traffico
 - individuazione di schemi di traffico (frequenza e durata della conversazione, diensione dei messaggi, etc.)
3. Mascheramento (spoofing)
 - inserimento di messaggi provenienti da una sorgente fasulla
4. Modifica dei contenuti
 - alterazione dei contenuti dei messaggi (inserimento, cancellazione, modifica)
5. Modifica della sequenza
 - modifica della sequenza dei messaggi
6. Modifica temporale
 - ritardo o ripetizione dei messaggi
7. Ripudio dell'origine
 - l'origine nega di aver inviato un messaggio
7. Ripudio della destinazione
 - la destinazione nega di aver ricevuto il messaggio

Contromisure

- i casi 1 e 2 riguardano la segretezza dei messaggi
 - crittografia
- i problemi 3,4,5,6 riguardano l'autenticità dei messaggi
 - autenticazione dei dati
- i problemi 7 e 8 riguardano ancora l'autenticità dei dati (origine/destinazione)
 - origine: certificati
 - destinazione: certificati + specifici protocolli

Message Authentication

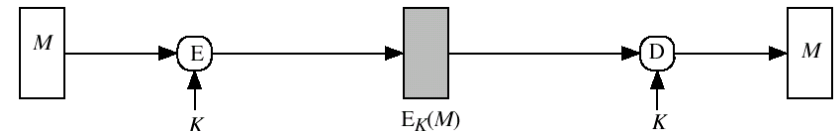
- Message authentication is concerned with:
 - **protecting the integrity of a message**
 - **validating identity of originator**
 - **non-repudiation of origin (dispute resolution)**
- Three alternative functions used:
 - **message encryption**
 - **message authentication code (MAC)**
 - **hash function**

5

Message Encryption (secret-key)

- If secret-key (symmetric) encryption is used:
 - **encryption provides both privacy and origin authentication**
 - **however, need to recognize corrupted messages (checksum/ MIC)**

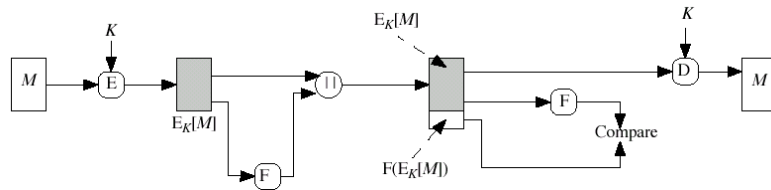
Symmetric encryption: confidentiality and origin authentication



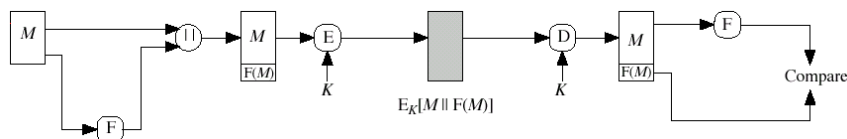
6

Message Encryption (secret-key)

External error control (checksum)



Internal error control (MIC)



7

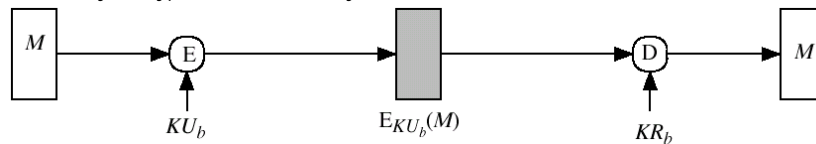
Message Encryption (public-key)

- if public-key encryption is used:
 - **encryption with public key provides no confidence of sender**
 - **since anyone potentially knows public-key**
 - **however if**
 - sender "signs" message using their private-key
 - then encrypts with recipients public key
 - have both secrecy and authentication
 - **again need to recognize corrupted messages**
 - **but at cost of two public-key uses on message**

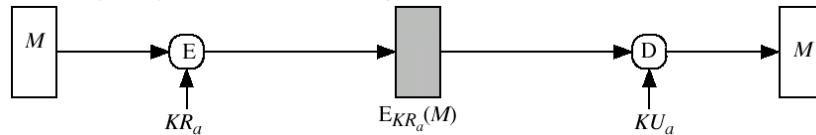
8

Message Encryption (public-key)

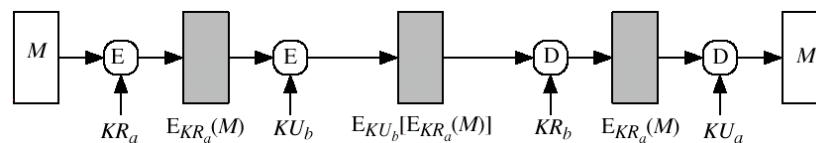
Public-key encryption: confidentiality



Public-key encryption: authentication/signature



Public-key encryption: confidentiality, authentication/signature



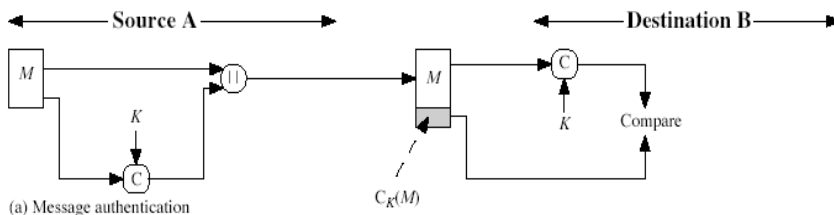
9

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

10

Message Authentication Code



11

Message Authentication Code (cont.)

- can also use encryption for secrecy
 - **generally use separate keys for each**
 - **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)**
- note that a MAC is not a digital signature

12

Requirements for MACs

- taking into account the types of attacks
- need the MAC to satisfy the following:
 - **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**

13

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 (≤ 64 bit)

14

MAC Security

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

15

HMAC

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

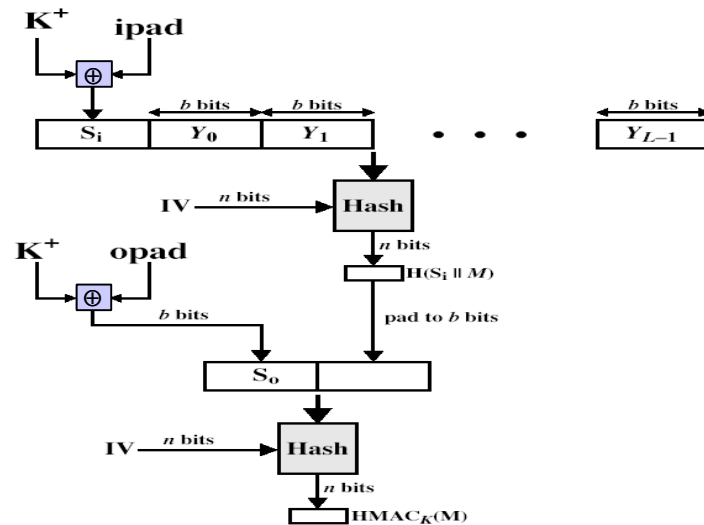
$$\text{HMAC}_K = \text{Hash}(K^+ \text{ XOR opad} \parallel \text{Hash}(K^+ \text{ XOR ipad} \parallel M))$$

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

16

HMAC



Peer entity authentication

17

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

19

Secrets-based Authentication

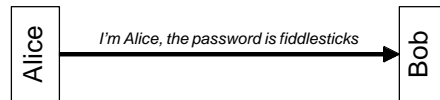
(.. Its not who you are. It's what you know.)

- Basic system uses passwords
 - **Can be easily intercepted**
- 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)

20

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**

21

Password security

- Do not send in clear except over short secure channels
- Choose good passwords
- Force changing passwords periodically
- Avoid keeping password in memory longer than necessary to generate the user's master key (KDC)
- Send hash of (key+nonce) for authentication
(**against replay attacks**)
- Add salt before hashing passwords for pw database
(**against reflection attacks**)
- Add realm name to password before hashing for pw db
(**against reflection attacks**)

22

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
- Passwds can be stored
 - **encrypted**
 - **hashed**

23

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**

24

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**

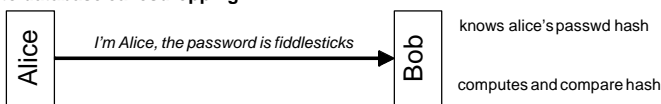
25

Replay and reflection

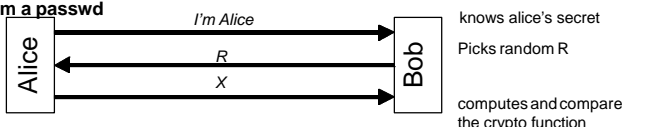
- Countermeasures against replay and reflection attacks include
 - **use of sequence numbers**
 - generally impractical
 - **timestamps**
 - needs synchronized clocks
 - **challenge/response**
 - using unique nonce, salt, realm values

26

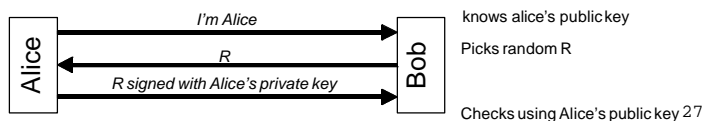
Eavesdropping and server database reading

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

sequenceDiagram
 participant Alice
 participant Bob
 Note over Alice: I'm Alice, the password is fiddlesticks
 Alice->>Bob: I'm Alice, the password is fiddlesticks
 Note over Bob: knows alice's passwd hash
computes and compare hash

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

sequenceDiagram
    participant Alice
    participant Bob
    Note over Alice: I'm Alice
    Alice->>Bob: I'm Alice
    Note over Bob: knows alice's secret  
Picks random R
    Bob-->>Alice: R
    Note over Alice: computes crypto function X=f(R)
    Alice->>Bob: X
    Note over Bob: computes and compare the crypto function
    
```
- Protection against both using asymmetric cryptography:



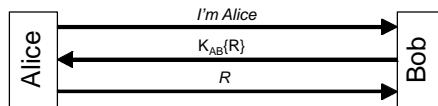
Authentication with shared secret



- 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 impersonate Alice**

28

Authentication with shared secret (variant 1)

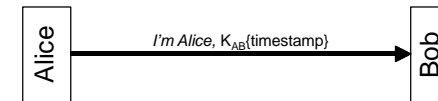


● differences:

- requires reversible cryptography
- if R is a recognizable quantity, Carol can mount an offline password-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

29

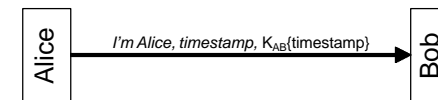
Authentication with shared secret (variant 2)



● differences:

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

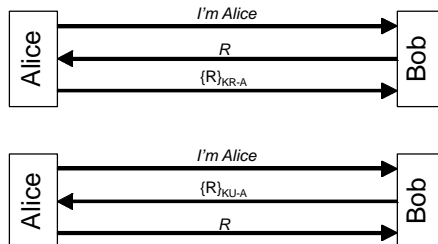
● variant:



30

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:

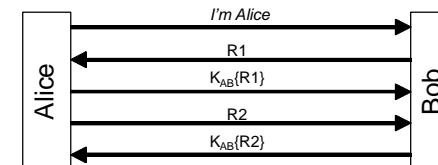
- if you can trick Alice into signing something, you can impersonate Alice

● contromisure:

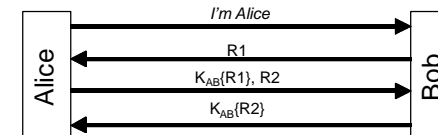
- general rule, not use the same key for two different purpose unless the design for all uses are coordinated
- e.g. impose enough structure to be signed (nonce, realm, timestamp, etc.)

31

Mutual authentication with shared secret



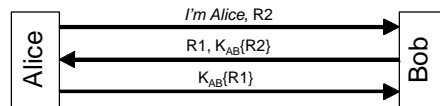
● or shorter..



32

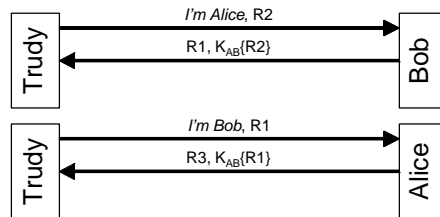
Mutual authentication with shared secret

or shorter..



● but:

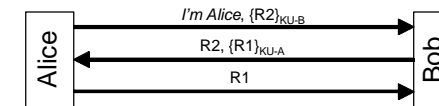
➤ Reflection attack



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

33

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

34

One-time passwords

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



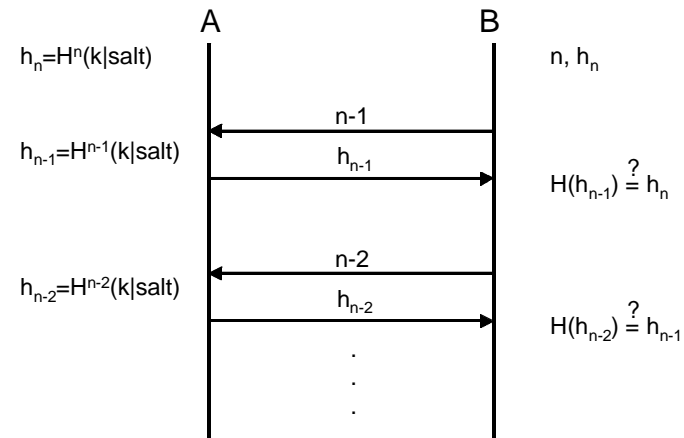
35

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

36

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.
```

