Authentication and Key Distribution

Ibéria Medeiros

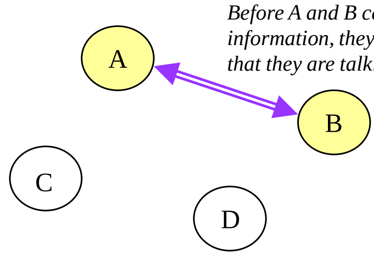
Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa

1

Authentication in a Distributed System

- Main objective:
  - allow a component to determine with whom it is talking to,
    and eventually
    support the creation of a shared secret key that can be used to provide secure communication

Before A and B can initiate the exchange of information, they need to be assured that they are talking with the right person

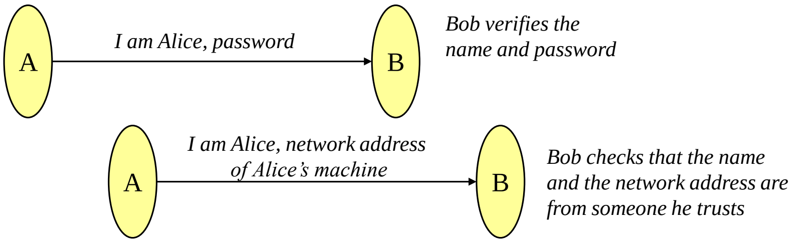


Applications:
- banking systems
- ....

2

Unilateral Authentication

- Only ensures the authenticity of the party that initiates the communication
- Two historical forms to achieve this:



Important problems

» someone that can listen to the network can get Alice's password, and later on can contact Bob pretending to be Alice
» someone can emulate the network address of the sender
» Alice has not guarantee that is talking with Bob

Example Attacks

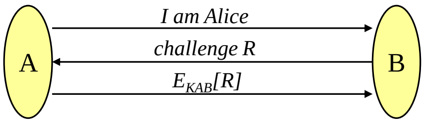
- **Eavesdrop** the network
  - see the message contents
  - obtain information that allows the future personification of Alice and/or Bob
  - obtain information that allows the personification of Alice in a Bob's replica
  - get data that supports the brute force attack on the passwords/secrets
- Initiate the communication and **personificate** Alice
  - Malory should be able to convince Bob that it is Alice
  - get data that supports brute force attacks on the password
  - obtain information that allows the personification of Alice in a future authentication
  - get data allows the personification of Bob to Alice
  - make Bob sign or decrypt something
- **Fake the network address of Bob** and wait for a connection of Alice
  - convince Alice that is Bob
  - get data that supports brute force attacks on the password

Attack Examples (*cont*)

- obtain information that allows the personification of Bob in a future authentication
- get data allows the personification of Alice to Bob
- make Alice sign or decrypt something

- **Read the database of secrets** of Alice/Bob
  - personificate Bob/Alice to Alice/Bob
  - decrypt messages exchanged previously
- Place **himself between** Alice and Bob and change messages
  - get data that supports brute force attacks on the password
  - continue the communication between Alice and Bob with a session hijacking
  - change/reorder/repeat messages without detection
- Mix of the above attacks
  - (Listen to the network + Read the DB of secrets of Alice and Bob)
    - » Malory should not be able to read previous conversations
  - (Read DB of Bob + Listen to the network)
    - » should not be able to personificate Alice to Bob

Unilateral Authentication with Shared Secret



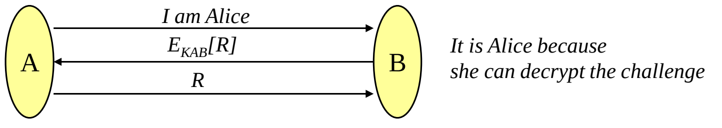
It must be Alice because she can encrypt the challenge

NOTE: the third operation does not need to be an encryption, since we only need a cryptographic transformation based in K_{AB} and R

+ the adversary even if he can listen to the channel cannot get K_{AB}; if Bob uses different challenges, the adversary can not personificate Alice
- Alice does not authenticate Bob; if Malory can personificate Bob, then she can read all packets sent to Bob and answer to them with Bob's network address
- after the initial exchange, the adversary can hijack the connection if they do not encrypt the messages; only needs to send messages with the address of Alice (and eventually, *but not necessarily,* has to read the messages returned to Alice address)
- allows off-line attacks on key K_{AB}; Malory listens to the connection and gets R and $E_{KAB}[R]$; later on can try to discover K_{AB} through a brute force attack

Unilateral Authentication with Shared Secret (cont)

- Malory can read Bob's key database (either by compromising Bob's machine or a backup) , and later on can personificate Alice (particularly relevant if Alice re-uses the key in other servers or machines)

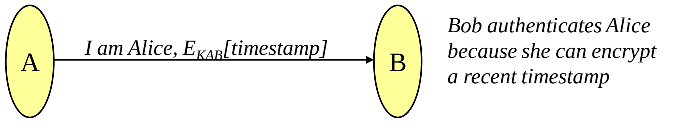


= similar security characteristics as before, although it requires $E_{KAB}[R]$ to be invertible (which sometimes can be a disadvantage)

\+ if R is a recognizable quantity with a limited lifetime (R = current_clock_time || random_number), then Alice also gets some guarantees that is talking with Bob

\- if R has a well-known format, then the adversary can get several $E_{KAB}[R]$ and perform a brute force attack on K_{AB} (without having to listen to the channel)

Unilateral Authentication with Shared Secret (cont)



Assumption: *Alice and Bob clocks are approximately synchronized*

\+ exchanges the minimum amount of messages (substitutes password with $E_{KAB}[timestamp]$)

\+ simplifies the protocol on the Bob's side -- no need to generate a challenge and store it

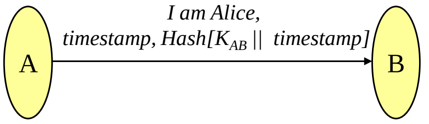
\+ this kind of authentication can be used in request/response type of communication (RPC)

\- a sufficiently fast adversary can personificate Alice; eavesdrops the channel and immediately re-uses $E_{KAB}[timestamp]$ (to avoid this problem, Bob needs to keep in a cache the timestamps that were received and are still valid within the time skew)

\- a fast adversary can personificate Alice if key K_{AB} is employed in several servers; Malory listens to the channel and immediately re-uses $E_{KAB}[timestamp]$ in another server (to avoid this problem, one should transmit $E_{KAB}[B \,||\, timestamp]$)

Unilateral Authentication with Shared Secret (cont)

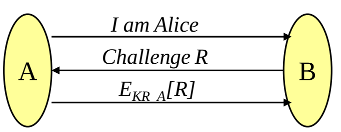
- an adversary that can convince Bob to set his clock back can personificate Alice; Malory re-uses an $E_{KAB}[timestamp]$ that was eavesdropped previously; this is a serious problem because sometimes people forget that security might be related to the clock
- the clock synchronization operation has to be performed in a secure way; this protocol needs to resort to a different authentication mechanism if clocks are too desynchronized



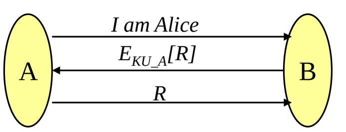
Bob authenticates Alice because she can create an hash of a recent timestamp together with the key

Similar protocol but uses a hash function

Unilateral Authentication with Public Key



Bob authenticates Alice because she can encrypt the challenge with her private key



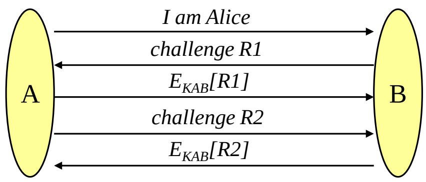
Bob authenticates Alice if she can decrypt the challenge with her private key

+ the database of public keys of Bob does not need to be confidential, although it is necessary to protect it from changes
- [*version 1*] the adversary can make Alice sign some arbitrary data; Malory waits for the initial contact from Alice, and then sends the data to Alice using Bob's network address
- [*version 2*] the adversary can make Alice decrypt some arbitrary data, which was previously encrypted by Alice

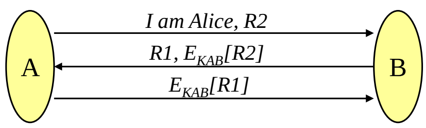
  *(RULE: **the same key should not be re-used for different proposes**, unless there is some coordination to prevent one protocol to break the other; for example, encrypt R with an identifier that corresponds to authentication)*

Mutual Authentication

- [version1] Mutual authentication in both directions

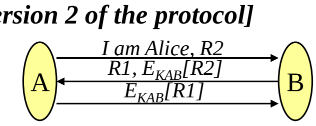


- [version 2] Optimizing the protocol to 3 messages



Reflection Attack

- Objective: the adversary wants to personificate Alice ***[version 2 of the protocol]***



1. contacts Bob and sends *R2*
2. gets from Bob the values: *R1* and $E_{KAB}[R2]$

   (at this moment it is not possible to continue the attack because he does not know how to create $E_{KAB}[R1]$)

3. starts a new connection to Bob, in parallel with the first one, and sends *R1*
4. gets from Bob the values: *R3* e $E_{KAB}[R1]$

   (at this moment he can return to the first connection)

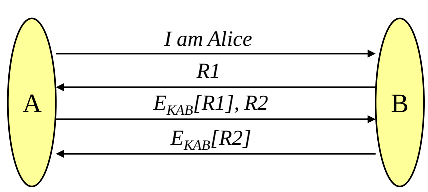
5. completes the authentication by sending to Bob: $E_{KAB}[R1]$

How to prevent the reflection attack?

- Solution1 : Use different keys to authenticate Bob and Alice
  - use two completely different keys
  - derive the key to authenticate Bob from the key used to authenticate Alice (e.g., key 1 = K_{AB}; key 2 = K_{AB} + 1)
- Solution 2: Use challenges with different formats
  - in one direction uses even challenges and on the other odd challenges
  - encrypt the identifier of the receiver together with the challenge

Mutual Authentication (cont)

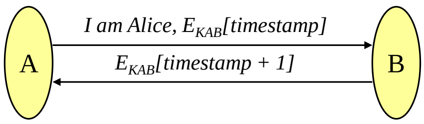
- [version 3] Try to make offline brute force attacks more difficult to perform



Rule for the design of security authentication protocols

The person who initiates the protocol should prove its identity first

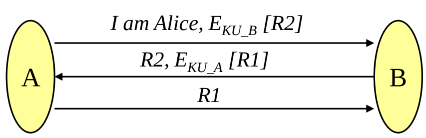
Mutual Authentication with a Timestamp



Assumption: *Alice and Bob clocks are approximately synchronized*

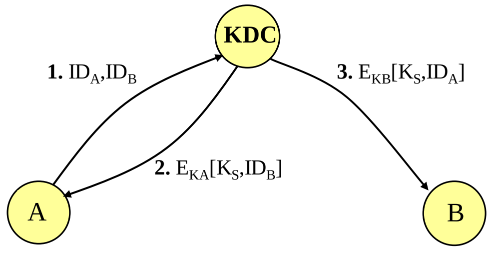
+ easier to add to request/response protocols (e.g., RPC)
= ..... *other characteristics mentioned previously* .....
- the use of *timestamp + 1* in the response could allow the personification of Alice (it is better to use a flag together with the timestamp or different keys)

Mutual Authentication with Public Key



- .... *some of the problems mentioned previously* ....
- How are the public keys obtained? (e.g., Alice connects to a new machine, and as expected she and Bob do not know each other public keys)
  - sign Bob's certificate with his public key with the private key of Alice
- How does Alice's machine get hers private key given that Alice only knows the password? (Note: it is simple to convert a password in a secret key, but typically it is impossible to convert a password in a private key due to the math requirements)
  - encrypt the private key with the password and save it with Bob

Mediated Authentication

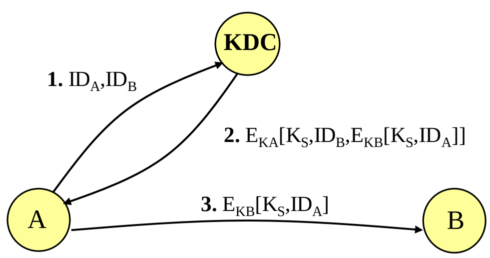


Terminology

KDC - *Key Distribution Center*
$ID_{A/B}$ – identifier of A/B
$K_{A/B}$ – secret key between KDC and A/B
$E_K[X]$ – encrypt X with key K
K_S – session key

+ each entity only needs a secret key (e.g., a key between Alice and the KDC)
+ although the KDC can not check who made the request, since the response comes encrypted, it can not be used by and adversary
= the protocol should be followed by a mutual authentication protocol
- Alice can start sending messages to Bob before **3.** is received
- KDC is required to communicate with Bob (which could create difficulties)

Mediated Authentication (cont)



Terminology

KDC - *Key Distribution Center*
$ID_{A/B}$ – identifier of A/B
$K_{A/B}$ – secret key between KDC and A/B
$E_K[X]$ – encrypt X with key K
K_S – session key

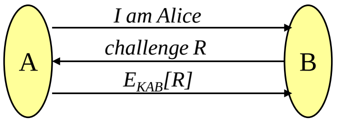
$E_{KB}[K_S, ID_A]$ is called ***ticket***

+ since Alice will have to talk to Bob, she can forward the **ticket** instead of the KDC, avoiding the previous two problems
= gives similar security guarantees
= the protocol should be followed by a mutual authentication protocol
- the ticket does not need to be encrypted in 2.

Key Distribution

- Many times, during the authentication, a new secret key is exchanged to protect the confidentiality and integrity of future communications

Example :



- The generation of the ***session key*** can be based on two kinds of information, the challenge and something related with the long term shared secret key (e.g., $K_S = E_{(KAB+1)}[R]$)
- It is bad idea to use
  - $K_S = E_{KAB}[R]$ because it is sent in the third message
  - $K_S = E_{KAB}[R+1]$ because later on the adversary can substitute Bob and send it as challenge $R+1$ (and get in the third message the key)

Example Methods for Generating a Shared Key (1)

- ***Assumed auth method***: Mutual Authentication with Public Key

1. Alice can choose a random K_S and send $E_{KU\_B}(K_S)$ to Bob

   NOTE1: if the adversary can put himself in the communication between Alice and Bob, he can substitute the session key by exchanging it with $E_{KU\_B}(K_{S1})$
2. ... and also, can sign $S_{KR\_A}[E_{KU\_B}(K_S)]$ with her private key

   NOTE2 : the adversary saves the conversation and later breaks into Bob's machine and obtains the private key (the adversary can read previous messages, and if Alice re-uses K_S, he can perform further attacks)
3. Alice sends $E_{KU\_B}(K_{S1})$ and Bob $E_{KU\_A}(K_{S2})$, and session key $K_S = K_{S1} \oplus K_{S2}$

   NOTE3 : 1) there is no need for the signatures because the attack of NOTE1 no longer works (*?why?*), 2) it is necessary to break both Alice and Bob's machines to obtain K_S

Is there a way to establish a session key that is able to resist to the attack that breaks the security of the two machines?

Example Methods for Generating a Shared Key (2)

- ***Assumed auth method***: Mutual Authentication with Public Key

4. Alice and Bob choose numbers *p* and *a*, and Alice sends S_{KR_A} $[a^{XA} \ mod \ p]$ and Bob sends S_{KR_B} $[a^{XB} \ mod \ p]$, and $K_S = a^{XA\,XB} \ mod \ p$ (Diffie-Hellman)

   NOTE4 : later on, even if the adversary breaks the security of Alice and Bob's machines, he cannot calculate K_S because *XA* and *XB* were forgotten

Bibliografia

- C. Kaufman, R. Perlman, M. Speciner, *Network Security: Private Communication in a Public World (2 edition), 2002: pag 257-287*