Authentication Protocols

Ibéria Medeiros

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa

PASSWORD BASED NETWORK AUTHENTICATION

Authentication with Passwords

- Alice wants to remotely authenticate to Bob
  - through a password
  - but Alice machine does not have any special support (e.g., a asymmetric private key) but we trust its code

- Possible solutions

  eavesdropping

  1. send the password through the network

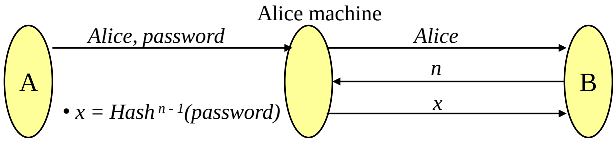
  man-in-the-middle

  2. make anonymous DH, and then encrypt the password with the resulting key
  3. create an hash of the password to obtain a key, and then use one of the symmetric key authentication protocol

  dictionary attack

  or ....
  - » one time password scheme (based on Lamport's hash)
  - » strong password protocols

Lamport Hash: One-time Passwords



- *Knows (n, Hash^n(password))*
- *Hash(x) = Hash^n(password)?*
  if yes, then save (n-1, x) and auth OK

+ each authentication uses a different password, avoiding security problems even if the adversary **eavesdrops the channel** or **reads the database of Bob**
- if $n=1$, Alice needs to send a new pair *(n, Hash^n(new_password))* to Bob in a secure way
- Men-in-the-middle attack to capture x, break connection to Alice and then act as Alice
- the adversary could act as Bob to get information that later could allow the personification of Alice; when Alice attempts to authenticate with Bob, the adversary sends her a small $M < n$ (e.g., $M=50$); with M, the adversary can personificate Alice for a number of times
- Extension: calculate ***z = Hash^n(password || salt || B)*** and send to Bob *(n, salt, z)*
  + the same password can be re-used across several servers
  + the same password can be used when n reaches 1
  + avoids attack (or increases the difficulty) where *Hash^n(password)* is pre-computed for all passwords and the Bob´s database is stolen

Real life limitations of one-time password

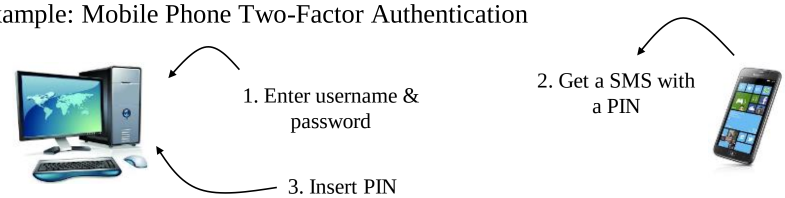
- **Luuuk Bank Theft Scheme Used Man-in-the-Browser Attack (June 2014)**

A bank theft scheme dubbed Luuuk stole 500,000 euros (US $681,000) from 190 account holders at an unnamed European bank in just one week. The thieves used a **man-in-the-browser attack** to steal account credentials and transferred stolen funds into accounts controlled by money mules. The thieves likely took advantage of **one-time passcodes** and skimmed the money **at the same time** that the legitimate customers were conducting online transactions. Luuuk targeted people in Italy and Turkey. The scheme was discovered in January 2014 when Kaspersky Lab found a command-and-control server for malware used to conduct man-in-the-browser attacks. Within days it had been wiped and shut down.

So, one-time passwords can be helpful, but further checks should be used!

Two Factor Authentication Solution

- One way to increase substantially the difficulty of carrying out the previous attack is to use two-factor authentication
  - something that the user possesses (e.g., USB stick token, a bank card, a key)
  - something that the user knows (e.g., username, password, PIN)
  - something that the user is and is inseparable from, a physical characteristic of the user (e.g., fingerprint, eye iris, voice, typing speed)
  - somewhere that the user has access to such as certain geographical location (GPS) or specific console terminal, etc.
- Example: Mobile Phone Two-Factor Authentication



Problems with Mobile Two Factor Authentication

- Swiss Bank Accounts Targeted in DNS and Malware Attacks (July 2014)

1. User computer is compromised (e.g., pressing a malicious link in a email & downloading some malware)

2. Malware runs:
a) changes the system's DNS server to one under the attackers' control;
b) installs a new root SSL public key;
c) deletes itself

3. User connects to a fake bank site and provides usersername & password

4. User is asked to provide PIN by downloading a fake mobile app (instead of receiving SMS)

5. The rogue app provides a fake password, but steals and forwards the real SMS to a malicious phone

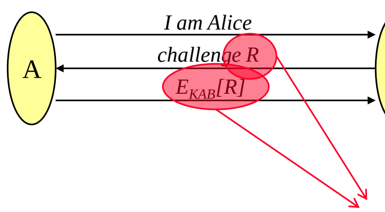
But **the way to go** is to use at least "two-factor authentication"

- **FBI Urges Use of Two-Factor Authentication (October 6, 2015)**

  The FBI is encouraging small- and medium-sized businesses and Internet users in general to use **two-factor authentication** to safeguard personal information. The FBI (did this) as part of this year's National Cyber Security Awareness Month. In a related story, a coalition of government agencies, technology companies, and security experts met in Washington, DC, earlier this week to discuss ways to move toward stronger, two-factor authentication.

Authentication Protocols with Strong Passwords

- Objective: prevent brute force attacks on the password even if someone
  - **observes the messages** exchanged during the authentication
  - can **personificate Alice** or **Bob**



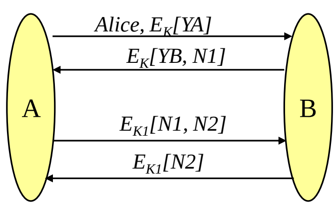
KAB is obtained from the password

Bob checks that it is Alice because he can decrypt the challenge

Allows a *off-line* brute force attack to be performed!

EKE – Encrypted Key Exchange

- *Since she knows the password, choose XA and calculates*
  $YA = a^{XA} \bmod p$
  $K = hash(password)$
- *Choose N2 and calculate $K1 = a^{XAXB} \bmod p$*



- *Knows K, choose XB and N1, and calculate $YB = a^{XB} \bmod p$*
- *Calculates $K1 = a^{XAXB} \bmod p$*

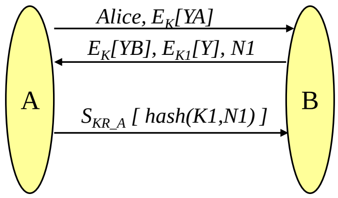
+ even if someone **listens to the channel** or **personificates Alice/Bob**, it is **not** possible to get information that supports the discovery of the password through a brute force attack

= it is possible to do an **online** brute force attack, but the probability of success is small and the attack can be detected

\- **Sophisticated attack**: $a^{XA} \bmod p$ is less than p, therefore
  - if one tries a password to decrypt the first message, and if the result is larger than p, then one can eliminate it immediately
  - if p is small, then it is possible to eliminate many passwords

SRP - Secure Remote Password

- *Choose XA and calculate YA and since she knows the password, calculate*
  K = func1(password)
  K^* = func2(password)
- *calculate*
  $K1 = a^{XAXB} \bmod p$
  and get private key K_{R_A} from Y



- *Knows Alice, K, Alice´s public key, and Y = E_{K^*}[Alice´s private key]*
- *Choose XB and N1 and calculates YB, calculates $K1 = a^{XAXB} \bmod p$*
- *Checks the signature and verifies the hash*

+ Even if the adversary **obtains the database of Bob**, he should **not** be able to impersonate Alice (unless a brute force attack on database data allowed him to obtain the password)

- NOTES: show that
  - someone that steals the database of Bob cannot personificate Alice
  - the protocol provides mutual authentication

SECURITY HANDSHAKES PROVIDING DIFFERENT PROPERTIES

Mutual Authentication & Session Key Establishment

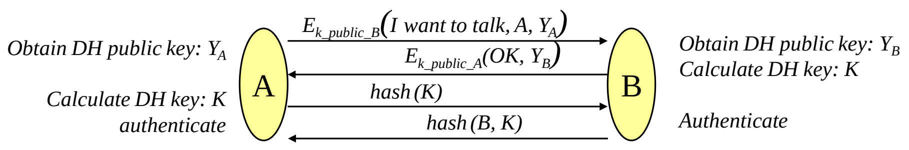
- Consider a protocol that allows two entities to ***authenticate mutually*** and that results in the creation of a new ***session key***
  - the <u>session key</u> helps to protect the channel against the attacks on
    - » data confidentiality and integrity
    - » session hijacking
  - a <u>sequence number</u> is used to prevent the channel from attacks
    - » replay
    - » reordering

NOTE: a *new unpredictable* key should be established for each session, and each packet should have a distinct sequence number; if sequence numbers *wrap around*, then a new key has to be negotiated

NOTE1: *both sides* should contribute to the session key, so that as long as one side has a good random generator, the key is sufficiently unpredictable

Perfect Forward Secrecy (PFS)

- *PFS* is ensured if it is impossible for the adversary to decrypt a conversation of two parties even if
  - she records the entire encrypted session
  - and later on breaks both peers and steals their long-term secrets
- The normal way to achieve this objective is to use a temporary session key, which is derived from *local* information in the node and is ***forgotten*** after the channel terminates (think about Diffie-Hellman)



NOTES:

- Is this protocol really secure from the attacks that you already know? (e.g., MITM)

Perfect Forward Secrecy (cont)

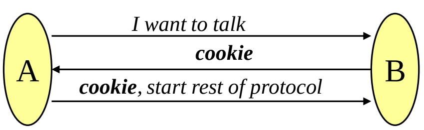
- It is also advisable to *periodically exchange* the temporary key with a new one, to ensure that even if a node is compromised and all keys become visible to the adversary, previous messages can not be read (only the current ones)

- *PFS* also protects the channel from the attack where the adversary already got the long-term keys (e.g., *asymmetric encryption keys*), but now can only listen to the network (a *passive attack*)
  - important if one wants to have some level of security with ***escrow systems***

- NOTE:
  - Can we provide protection even with *active attacks* and *an escrow system*?

Denial of Service Protection

- Attacks that attempt to prevent correct hosts from utilizing a certain service by making the server use its resources (memory and/or CPU) with authentication attempts (associated many times with IP spoofing)
  - ***Cookies*** are unpredictable values that need to be echoed by the other side



cookie = hash(IP of A, secret_of_B)
... forget cookie ... (why?)
Is cookie equal to hash(IP of A, secret_of_B)?
If yes, continue the protocol ...

NOTES:

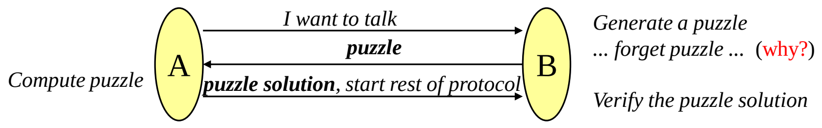
» Bob does not do any significant computation to generate the cookie

» cookies could only be used when the server is being swamped with packets

» protects only from a subset of the attacks but does not cost much (basically one extra round trip delay) …

Denial of Service Protection (cont)

- ***Puzzles*** are based on the idea that if the server is getting swamped with requests, it will require the initiators to do more computation in order to connect



NOTES:

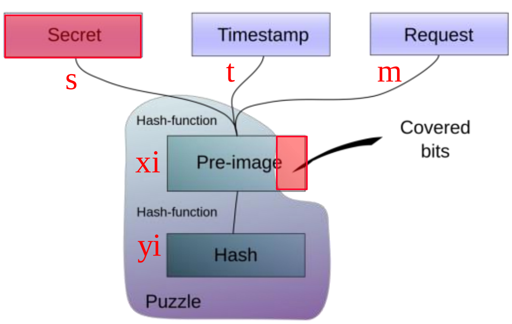
» B can require arbitrary amounts of computation from A by varying the difficulty of the puzzle

» a puzzle can be something like "*what 27-bit number has a digest of X?*"

» the effect of the puzzle is slowing down the adversary

» there are some potential limitations with this solution (e.g., slow hosts might take too long; puzzle solving in parallel; distributed denial of service)

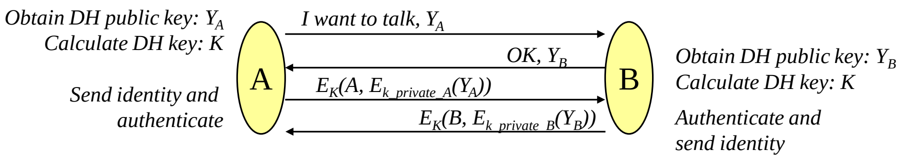
Example puzzle



- The client sends to the server a request *m* that is unique
- The server calculates
  - *xi = hash(s, t, m)*
  - *yi = hash(xi)*
  - *xi* has *n* bits
- The server sends to the client
  - *<t, m, yi, (n-k) bits of xi>*
- The client makes a brute force effort to find the missing *k* bits of *xi* by experimenting different values and checking if they result in *yi*
- The client returns
  - *<t, m, xi>*
- The server validates *t* and *xi*

Identity Hiding

- The objective is to hide the identity of the two peers from an adversary observing network (passive attack) or acting on the messages (active attack)
  - the simplest solution is based on anonymous Diffie-Hellman, which works well for passive attacks but is vulnerable to man-in-the-middle attacks
  - another solution, which works for passive attacks and only reveals A identity for active attacks (i.e., before the adversary is discovered)



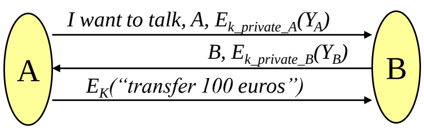
NOTES for active attacks:

» the protocol can be modified to hide Alice identity instead

» if Alice already knows Bob public encryption key or if they share a secret key then it is possible to hide both identities

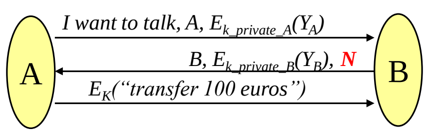
Live Partner Reassurance

- The objective is to **allow for DH parameter reuse** (on B side) and still protect the channel from the **replay of old messages**, namely old messages used to set up the connection



*Since the DH computation of Y_B consumes some resources, B would like to re-use Y_B, but now we are **vulnerable to replay attacks***

*K = hash(**N**, DH key)*



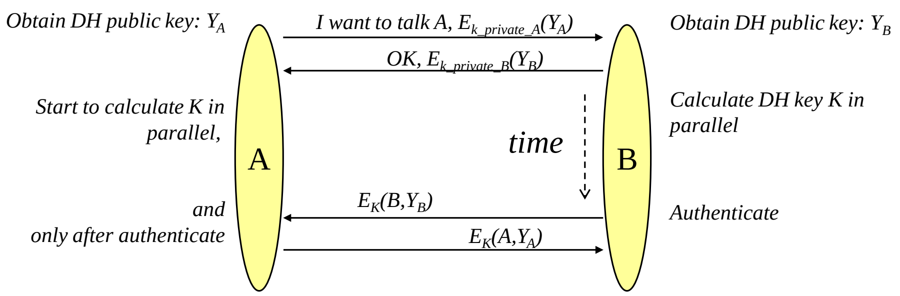
NOTES:

» the nonce could also act as a stateless cookie (but our previous algorithm for cookie generation would need to be improved)

» the protocol could also be modified to ensure A that B is alive!

Parallel Computation

- The objective is to improve the performance by allowing the computation of the DH key exponentiation in parallel



Bibliografia

- C. Kaufman, R. Perlman, M. Speciner, *Network Security: Private Communication in a Public World (2 edition), 2002: pag 291-301, and 403-420*