


PURDUE UNIVERSITY

Computer Security CS 426 Lecture 6

Entity Authentication

Elisa Bertino
Purdue University
IN, USA
bertino@cs.purdue.edu



PURDUE UNIVERSITY

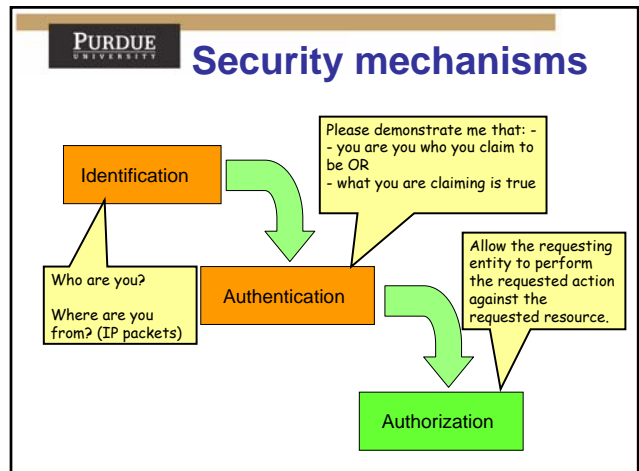
TOC

- Entity Authentication
- Password-based authentication
- Challenge-response authentication
- Digital certificate

PURDUE UNIVERSITY

Security mechanisms - Authentication

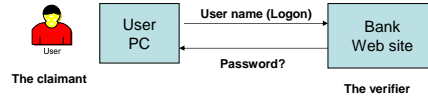
- Authenticating an object (data/message) means confirming its provenance, whereas authenticating an individual/an entity consists of *verifying his/her/its identity*
- **Entity authentication:** in computer security, entity authentication is the process of attempting to verify the **digital identity** of another party.
- An entity can be a person, a process/**program**, a machine:
 - For example, SSL authenticates the client (program) to the server application (and possibly viceversa).
 - SSL does not authenticate the client's user to the application running on the server.



Authentication

- The entity whose identity needs to be proved is called the *claimant*
- The party that tries to prove the identity of the claimant is called the *verifier*
- Entity authentication can be:
 - Unidirectional
 - Bidirectional (for example SSL when it authenticates the client (program) to the server application and viceversa).

Example



Data-origin vs. Entity Authentication

- Message (data origin) authentication might not happen in real time, entity authentication does:
 - When BOB authenticates a message (for example an e-mail) from Alice, Alice might not be connected to her e-mail system.
 - When Alice tries to access the Bank web site, or she gets cash from an ATM, the Bank web site/ATM must perform Alices' authentication *before* allowing any other message to be exchanged.
- Goal of authentication:
 - Message (data origin) authentication must be repeated for each message
 - Entity authentication authenticates the claimant for the entire duration of a session

Authentication

The entity authentication process can be based on:

- **Something known** (to the claimant *and* to the verifier): a password, a PIN, a secret key
- **Something possessed** by the claimant: one-time password token, a mobile phone, a smart-card/USB token
- **Something inherent** to the claimant: fingerprints, voice, facial characteristics, retinal pattern, vein pattern (biometrics)

The above authentication factors can be combined to achieve multi-factor authentication

Authentication mechanisms

Entity authentication mechanisms:

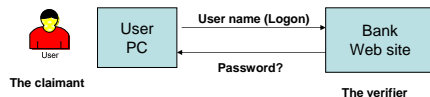
- Username/password
 - Fixed password
 - One-Time password
- Challenge Response
 - Using symmetric key
 - Using asymmetric keys

9

Fixed password

- A password is associated with each user which remains valid for a certain period of time (usually defined by the organizations' policy):
 - The password is not generated on a per-access request
- The verifier must store the user name and his/her associated password (information at rest)
- The password is stored as plaintext at the verifier site
- To perform the authentication, the claimant transmits the password over the network (information in transit)

Example



Fixed password possible attacks and vulnerabilities

- Attack1 - Eavesdropping:
 - Case 1: At the user PC (Eve can watch Alice when she types the password)
 - Case 2: Over the network (sniffing)
- Vulnerabilities leading to attack 1:
 - Case 1: Eve trusts Alice
 - Case 2: the password is sent as plaintext (unencrypted) over the network
- Countermeasures:
 - Case 1: Alice should not trust Eve
 - Case 2: encrypt the password when sending it over the network

Fixed password possible attacks and vulnerabilities

- Attack 2 – Password guessing:
 - Eve can log into the system (example: the Bank web site) and try to guess Alice's password by trying different combinations of characters
- Vulnerabilities leading to attack 2:
 - Alice chose a short password (and the application did not require a longer password) and
 - the verifier did not implement a maximum retry policy (i.e. if a wrong password is submitted for 3 times, then block the user name account)
- Countermeasures:
 - the verifier should enforce a password of a minimum length, composed by a mix of numerical and special characters; the verifier should enforce a maximum retry policy

Fixed password possible attacks and vulnerabilities

- Attack 3 – The attacker can access the password file
- Vulnerabilities leading to attack 3:
 - The password file is not read/write protected
- Countermeasures:
 - Protect the password file

Fixed password storing password hash

- When the password is created, the verifier stores the hash of the password instead of the plaintext password. The hash values are known to the verifier only.
- When the user sends his user name (ID) and the password, the verifier creates a hash of the password and the compares it with the stored password hash. If there is a match the user is authenticated successfully.
- If an attacker tries to obtain the password of a specific user, and he succeeds in accessing the password repository, it is difficult for him to guess the password from the hash value
- However, a brute force attack against the password repository is still possible.

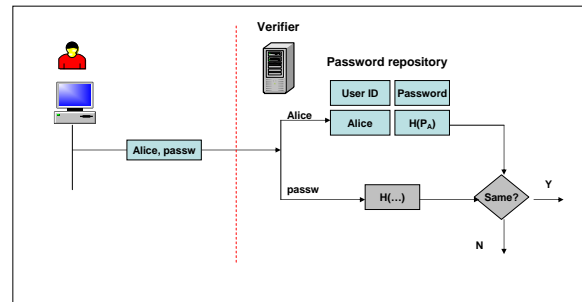
Brute force attack

- Brute force attacks can be made less effective by *obfuscating* the data (as it is more difficult to determine when one has succeeded in breaking the code)
- In cryptography, obfuscation refers to encoding the input data before it is sent to a hash function (or other encryption scheme). This technique helps making brute force attacks unfeasible, as it is difficult to determine the correct cleartext.

Dictionary attack

- A dictionary attack is a technique for defeating a cipher or authentication mechanism by trying to determine its decryption key or passphrase by searching a large number of possibilities.
- In contrast with a brute force attack, where all possibilities are searched through exhaustively, a dictionary attack only tries possibilities which are most likely to succeed, typically derived from a list of words in a dictionary.
 - Generally, dictionary attacks succeed because many people have a tendency to choose passwords which are short (7 characters or fewer), single words in a dictionary, or are simple variations that are easy to predict, such as appending a single digit to a word.

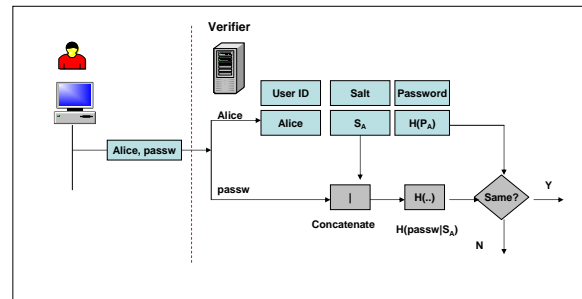
Password hash



Fixed password password salting

- When the verifier creates the password string, a **random string (the salt) is concatenated to the password**. The salted password is then hashed.
- The user ID (account name), the salt and the hash are stored in the password repository.
- When a user asks for access, he/she provides his/her ID (account name) and the password; the application extracts the salt from the password repository, concatenates it to the password provided by the user, makes a hash of it and compares the hash with the stored one.
- If the two hashes matches, then the user is successfully authenticated

Password salting



Password salting

- ☞ Password salting makes brute force attacks more difficult:
 - For example, if the password is 6 digit long and the salt is 4 digit long, hashing is done over a 10-digit value
 - The attacker needs to create a list of 10 millions items (10^{10}) and create a hash for each of them. The comparison of the attacker hashes with the stored hashes takes much longer.
 - Thus, salting is effective if the salt is a very long random number
- ☞ However, Alice's password might be still stolen...

One-time password

- A one-time password (OTP) is a password that is used only once:
 - If an attacker can get a (still unused) one-time password, she can use it only for a limited period of time
 - This prevents eavesdropping and replay attacks

22

OTP – Hash Chain

- A hash chain is a successive application of a [cryptographic hash function](#) $h(x)$ to a string.
- For example, $h(h(h(h(x))))$ gives a hash chain of length 4, often denoted $h^4(x)$
- [Lamport](#) suggested the use of hash chains as an OTP mechanism
- A server which needs to provide [authentication](#) may store a hash chain rather than a [plain text](#) password
- For example,
 - a server begins by storing $h^{1000}(\text{password})$, where *password* is provided by the user
 - When the user wishes to authenticate, he supplies $h^{999}(\text{password})$ to the server.
 - The server computes $h(h^{999}(\text{password})) = h^{1000}(\text{password})$ and verifies that this matches the hash chain it has stored. It then stores $h^{999}(\text{password})$ for the next time the user wishes to authenticate.
 - An eavesdropper seeing $h^{999}(\text{password})$ communicated to the server will be unable to re-transmit the same hash chain to the server for [authentication](#) since the server now expects $h^{998}(\text{password})$. Due to the [one-way property](#) of [cryptographically secure hash functions](#), it is impossible for the eavesdropper to reverse the hash function and obtain an earlier piece of the hash chain. In this example, the user could authenticate 1000 times before the hash chain is exhausted.

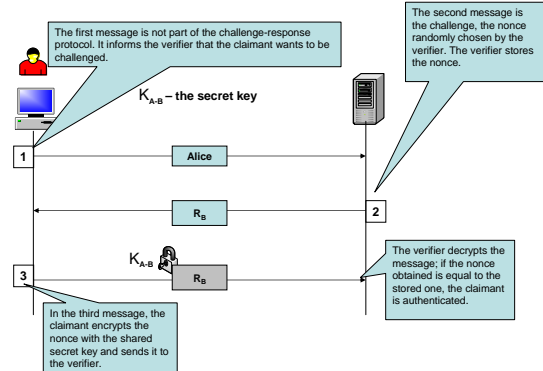
Challenge-response authentication protocol

- With password authentication the claimant proves her identity by demonstrating to the verifier that she knows a secret (the password)
 - this secret has to be revealed by the claimant, so it is susceptible to interception by an adversary
- ☞ In challenge-response authentication the claimant proves that she knows a secret without sending it (every time)

Challenge-response with symmetric key

- ☞ The claimant and the verifier share a secret (a symmetric key)
- ☞ The challenge is a time-varying value sent by the verifier. Two possibilities:
 - The challenge is a *nonce* (*nonce = number used only once*)
 - The challenge is a timestamp
- ☞ The claimants' response is the result of a function (encryption) applied by the claimant to the challenge

Nonce challenge



Nonce challenge

- A nonce prevents a replay attack of the third message.
- The attacker (Eve) replays the third message, pretending it to be a new authentication request sent by the claimant (Alice), *but*
- When the verifier (Bob) receives the attacker replayed third message, the value of R_B is no longer valid.

Timestamp challenge

- The time-varying value is a timestamp.
- To be applicable, this approach requires that the clocks of the claimant and of the verifier are synchronized.
- In this case, the claimant sends her user ID and her timestamp encrypted with the symmetric key shared by the claimant and the verifier.
- Timestamp prevents replay attack
- Impersonation still possible, if the secret key is stolen

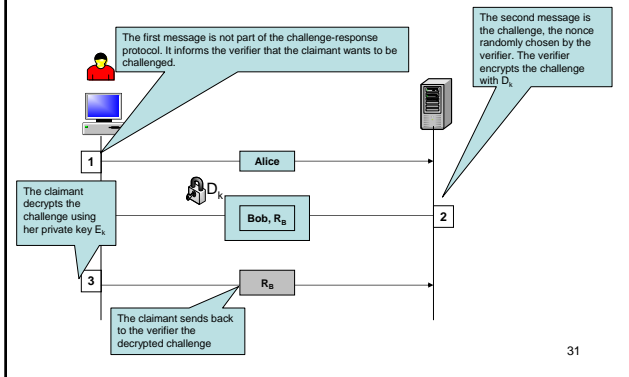
Authentication based on asymmetric key

- Now the secret is the private key of the claimant
- The verifier has the corresponding claimants' public key
- The claimant must prove to the verifier that she possesses the private key corresponding to (her) public key

Authentication based on asymmetric key

- In this scenario, the verifier encrypts the challenge using the claimants' public key
- The claimant decrypts the challenge using her private key
- The claimant sends back to the verifier the decrypted challenge

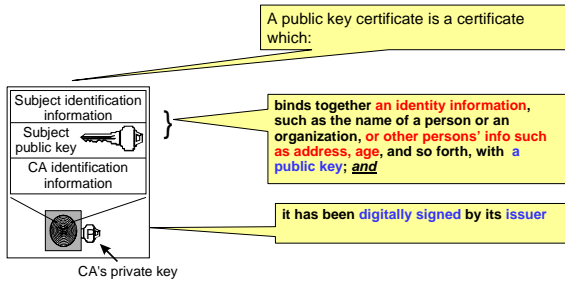
Nonce challenge



Credential and Digital Certificate

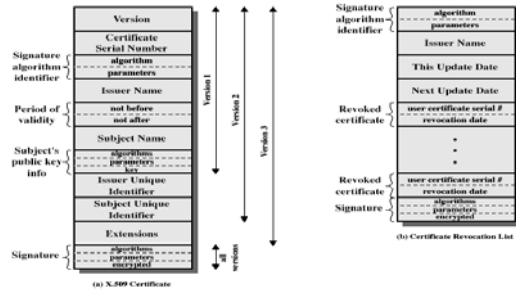
- A credential is a statement by an issuer on some properties of the subject of the credential
- A digital certificate contains the digital signature of the certificate-issuing authority so that anyone can verify that the certificate is real:
 - Anyone can verify that what is asserted by the certificate (claims) is true

Public Key certificates



• A digital certificate has a well defined lifetime

Certificate standards: X.509



X.509 certificate content

- A X.509 certificate is issued by a Certification Authority (CA). It contains the following info:
 - version (1, 2, or 3)
 - serial number (unique within the CA) identifying the certificate
 - signature algorithm identifier
 - issuer X.509 name (CA)
 - period of validity (from - to dates)
 - subject X.509 name (distinguished name -DN -)
 - CN=Java Duke, OU=Java Software Division, O=Sun Microsystems Inc, C=US
 - subject public-key info (algorithm, parameters, key)
 - issuer unique identifier (v2+)
 - subject unique identifier (v2+)
 - extension fields (v3)

X.509 Distinguished Name

DN Field	Abbrev.	Description	Example
Common Name	CN	Name being certified	CN=Joe Average
Organization or Company	O	Name is associated with this organization	O=Snake Oil, Ltd.
Organizational Unit	OU	Name is associated with this organization unit, such as a department	OU=Research Institute
City/Locality	L	Name is located in this City	L=Snake City
State/Province	ST	Name is located in this State/Province	ST=Desert
Country	C	Name is located in this Country (ISO code)	C=XZ

References

- Robert Morris and Ken Thompson: Password Security: A Case History
www.cs.unibo.it/~montreso/doc/papers/Morris-PasswordSecurity.pdf
- Password salt:
http://en.wikipedia.org/wiki/Password_salt
- One-time password:
http://en.wikipedia.org/wiki/One-time_password
- IETF RFC 2289 - A One-Time Password System
<http://www.faqs.org/rfcs/rfc2289.html>
- NIST FIPS 112 PASSWORD USAGE (and guidelines) (NOTE: recommended reading)
<http://www.itl.nist.gov/fipspubs/fip112.htm>

References

- NIST Cryptographic Toolkit
<http://csrc.nist.gov/CryptoToolkit/tkencryption.html>
- NIST FIPS PUB 113 COMPUTER DATA AUTHENTICATION
<http://www.itl.nist.gov/fipspubs/fip113.htm>
- NIST FIPS 196 Entity Authentication Using Public Key Cryptography
<http://csrc.nist.gov/publications/fips/fips196/fips196.pdf>
- RFC 3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
<http://www.ietf.org/rfc/rfc3280.txt>