Authentication protocols

Authentication: Definition

- Proof that an entity has an attribute it claims to have

  —Hi, I'm Joe
  —Prove it!
  —Here are my Joe's credentials
  —Credentials accepted/not accepted

  —Hi, I'm over 18
  —Prove it!
  —Here is the proof
  —Proof accepted/not accepted

Authentication: Proof Types

- Something we know
  - A secret memorized (or written down…) by Joe
- Something we have
  - An object/token solely held by Joe
- Something we are
  - Joe's Biometry

- Multi-factor authentication
  - Simultaneous use of different proof types

Authentication: Goals

- Authenticate interactors
  - People, services, servers, hosts, networks, etc.

- Enable the enforcement of authorization policies and mechanisms
  - Authorization $\Rightarrow$ authentication

- Facilitate the exploitation of other security-related protocols
  - e.g. key distribution for secure communication

Authentication: Requirements

- Trustworthiness
  - How good is it in proving the identity of an entity?
  - How difficult is it to be deceived?
  - Level of Assurance (LoA)
- Secrecy
  - No disclosure of secret credentials used by legitimate entities

Authentication: Requirements

- Robustness
  - Prevent attacks to the protocol data exchanges
  - Prevent on-line DoS attack scenarios
  - Prevent off-line dictionary attacks
- Simplicity
  - It should be as simple as possible to prevent entities from choosing dangerous shortcuts
- Deal with vulnerabilities introduced by people
  - They have a natural tendency to facilitate or to take shortcuts

Authentication:
Entities and deployment model

- Entities
  - People
  - Hosts
  - Networks
  - Services / servers
- Deployment model
  - Along the time
    - Only when interaction starts
    - Continuously along the interaction
  - Directionality
    - Unidirectional
    - Bidirectional

Authentication interactions:
Basic approaches

- Direct approach
  - Provide credentials
  - Wait for verdict
- Challenge-response approach
  - Get challenge
  - Provide a response computed from the challenge and the credentials
  - Wait for verdict

Authentication of people: Direct approach w/ known password

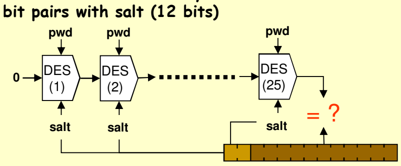
- A password is checked against a value previously stored
  - For a claimed identity (username)

- Personal stored value:
  - Transformed by an unidirectional function
  - Windows: digest function
  - UNIX: DES hash + salt
  - Linux: MD5 + salt

DES *hash* = $DES_{pwd}^{25}(0)$

$DES_k^n(x) = DES_k(DES_k^{n-1}(x))$

Permutation of 12 subkeys´ bit pairs with salt (12 bits)



Authentication of people: Direct approach w/ known password

- Advantage
  - Simplicity!

- Problems
  - Usage of weak keys
    - They enable dictionary attacks
  - Transmission of passwords along insecure communication channels
    - Eavesdroppers can easily learn the password
    - e.g. Unix remote services, PAP

Top Ten 2013
by Splashdata

1. 123456
2. password
3. 12345678
4. qwerty
5. abc123
6. 123456789
7. 111111
8. 1234567
9. iloveyou
10. adobe123

Authentication of people: Direct approach with biometrics

- People get authenticated using body measures
  - **Biometric samples**
  - Fingerprint, iris, face geometrics, voice timber, manual writing, vein matching, etc.
- Measures are compared with personal records
  - **Biometric references** (or **template**)
  - Registered in the system with a previous enrolment procedure

Authentication of people: Direct approach with biometrics

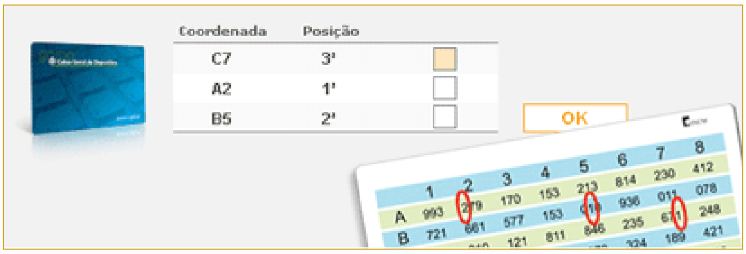
- Advantages
  - People do not need to use memory
    - Just be their self
  - People cannot chose weak passwords
    - In fact, they don't chose anything
  - Authentication credentials cannot be transferred to others
    - One cannot delegate its own authentication

Authentication of people: Direct approach with biometrics

- Problems
  - Biometrics are still being improved
    - In many cases they can be easily cheated
  - People cannot change their credentials
    - Upon their robbery
  - Credentials cannot be (easily) copied to others
    - In case of need in exceptional circumstances
  - It can be risky for people
    - Removal of body parts for impersonation of the victim
  - Its not easy to deploy it remotely
    - Requires trusting the remote sample acquisition system
  - Can reveal personal sensitive information
    - Diseases

Authentication of people: Direct approach with one-time passwords

- One-time passwords
  - Passwords that can be used just once
- Example: bank codes



Authentication of people:
Direct approach with one-time passwords

- Advantage
  - They can be eavesdropped, nevertheless attackers cannot impersonate the password owner
- Problems
  - Interactors need to know which password they should use at different occasions
    - Requires some form of synchronization
  - People may need to use extra resources to maintain or generate one-time passwords
    - Paper sheets, computer programs, special devices, etc.

Authentication of people:
Direct approach with one-time passwords

RSA SecurID

Example:
RSA SecurID

- Personal authentication token
  - There are also software modules for handhelds (PDAs, smartphones, etc.)
- It generates a unique number at a fixed rate
  - Usually one per minute (or 30 seconds)
  - Bound to a person (User ID)
  - Unique number computed with:
    - A 64-bit key stored in the card
    - The actual date
    - A proprietary digest algorithm (SecurID hash)
    - An extra PIN (only for some tokens)

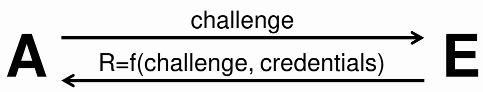
Example:
RSA SecurID

- One-time password authentication
  - A person generates an OTP combining a User ID with the current token number

    OTP = User ID, Token Number
- An RSA ACE Server does the same and checks if they match
  - It also knows the person's key stored in the token
  - There must be a synchronization to tackle clock drifts
    - RSA Security Time Synchronization
- Robust against dictionary attacks
  - Keys are not selected by people

Challenge-response approach: Generic description

- The authenticator provides a challenge
- The entity being authenticated transforms the challenge using its authentication credentials
- The result is sent to the authenticator
- The authenticator check the result
  - Produces a similar result and checks if they match
  - Transforms the result and checks if it matches the challenge or a related value



Challenge-response approach: Generic description

- Advantage
  - Authentication credentials are not exposed
- Problems
  - People may require means to compute responses
    - Hardware or software
  - The authenticator may have to have access to shared secrets
    - How can we prevent them from using the secrets elsewhere?
  - Offline dictionary attacks
    - Against recorded challenge-response dialogs
    - Can reveal secret credentials (passwords, keys)

Authentication of people:
Challenge-response with smartcards

- Authentication credentials
  - The smartcard
    - e.g. Citizen Card
  - The private key stored in the smartcard
  - The PIN to unlock the private key
- The authenticator knows
  - The corresponding public key
  - or some personal identifier
    - which can be related with a public key through a (verifiable) certificate

Authentication of people:
Challenge-response with smartcards

- Signature-based protocol
  - The authenticator generates a random challenge
    - Or a value not used before
  - The card owner ciphers the challenge with its private key
    - PIN-protected
  - The authenticator decrypts the result with the public key
    - If the output matches the challenge, the authentication succeeds
- Encryption-based protocol
  - Possible when private key decryption is available

A $\xrightarrow{\text{challenge}}$ **Card Owner**

K_{pub} $\xleftarrow{R=E(K_{priv},\ \text{challenge})}$  **Card, pin, K_{priv}**

Authentication of people:

Challenge-response with memorized password

- Authentication credentials
  - Passwords selected by people
- The authenticator knows
  - All the registered passwords; or
  - A transformation of each password
    - Preferable option
    - Preferably combined with some local value
      - Similar to the UNIX salt

Authentication of people:

Challenge-response with memorized password

- Basic challenge-response protocol
  - The authenticator generates a random challenge
  - The person computes a transformation of the challenge and password
    - e.g. a joint digest: response = digest (challenge, password)
    - e.g. an encryption response = $E_{password}$ (challenge)
  - The authenticator does the same (or the inverse)
    - If the output matches the response (or the challenge), the authentication succeeds
  - Examples
    - CHAP, MS-CHAP v1/v2, S/Key

PAP e CHAP

(RFC 1334, 1992, RFC 1994, 1996)

- Protocols used in PPP (*Point-to-Point Protocol*)
  - Unidirectional authentication
    - Authenticator is not authenticated
- PPP developed in 1992
  - Mostly used for dial-up connections
- PPP protocols used by PPTP VPNs
  - e.g. vpn.ua.pt

PAP e CHAP

(RFC 1334, 1992, RFC 1994, 1996)

- PAP (*PPP Authentication Protocol*)
  - Simple UID/password presentation
  - Insecure cleartext password transmission
- CHAP (*CHallenge-response Authentication Protocol*)

  Aut → U: authID, challenge
  U → Aut: authID, MD5(authID, pwd, challenge), identity
  Aut → U: authID, OK/not OK

  - The authenticator may require a reauthentication anytime

MS-CHAP (Microsoft CHAP)

(RFC 2433, 1998, RFC 2759, 2000)

- Version 1
  - A → U: authID, C
  - U → A: R
  - A →U: OK/not OK

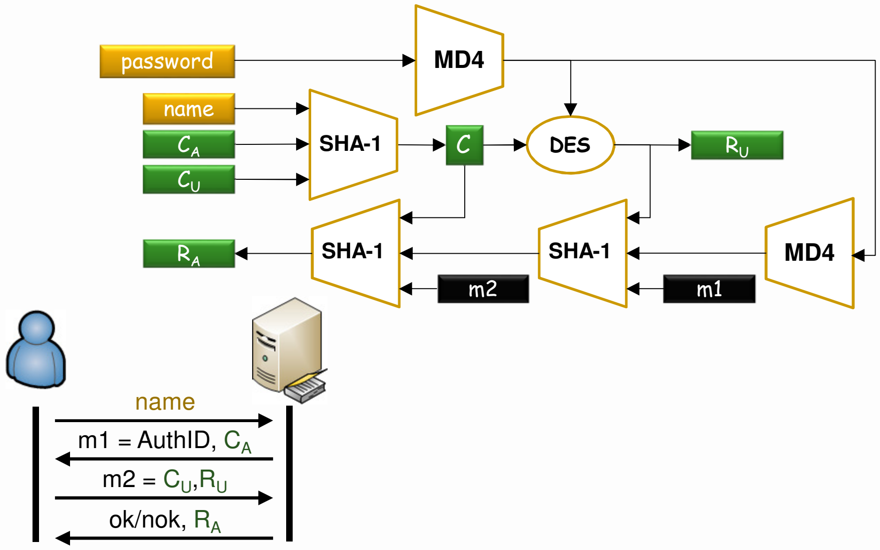
  $R = DES_{PH}(C)$
  PH = LMPH ou NTPH
  LMPH = DEShash(password)
  NTPH = MD4(password)

- Version 2
  - A → U: authID, C_A ← m1
  - U → A: C_U , R1 ← m2
  - A → U: OK/not OK, R2

  $R1 = DES_{PH}(C)$
  $C = SHA(C_U, C_A, username)$
  PH = MD4(password)
  R2 = SHA(SHA(MD4(PH), R1, m1), C, m2)

- Mutual authentication
- Passwords can be updated

MS-CHAP v2



S/Key
(RFC 2289, 1998)

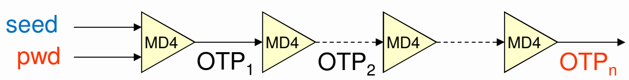
- Authentication credentials
  - A password (pwd)
- The authenticator knows
  - The last used one-time password (OTP)
  - The last used OTP index
    - Defines an order among consecutive OTPs
  - An seed value for the each person's OTPs
    - The seed is similar to a UNIX salt

S/Key
(RFC 2289, 1998)

- Authenticator setup
  - The authenticator defines a random seed
  - The person generates an initial OTP as:

    $OTP_n = h^n$ (seed, pwd), where h = MD4

    - Some S/Key versions use MD5 or SHA-1 instead of MD4
  - The authenticator stores seed, n and OTP_n as authentication credentials



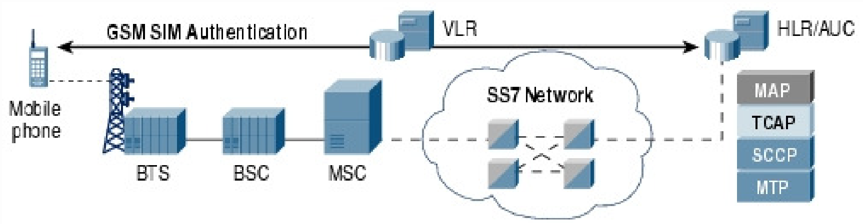
S/Key:
Authentication protocol

- Authenticator sends seed & index of the person
  - They act as a challenge
- The person generates index-1 OTPs in a row
  - And selects the last one as result
  - result = $OPT_{index-1}$
- The authenticator computes h (result) and compares the result with the stored OPT_{index}
  - If they match, the authentication succeeds
  - Upon success, stores the recently used index & OTP
    - index-1 and $OPT_{index-1}$

Authentication of people:
Challenge-response with shared key

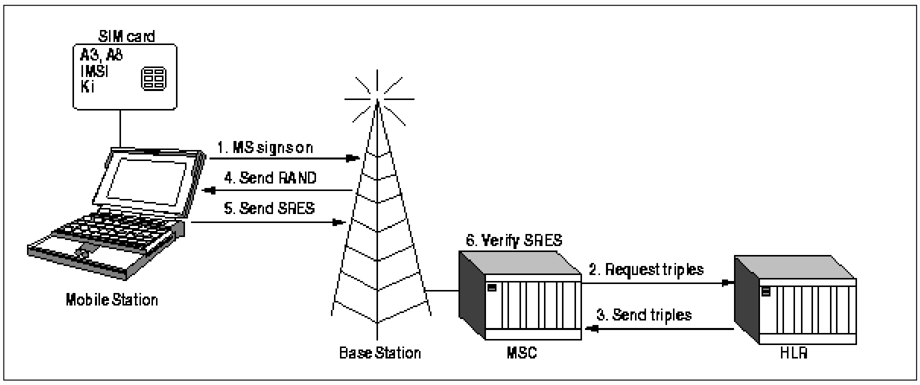
- Uses a shared key instead of a password
  - More robust against dictionary attacks
  - Requires some token to store the key
- Example:
  - GSM

GSM:
Authentication architecture



- Based on a secret key shared between the HLR and the Mobile Phone
  - 128 Ki, stored in the Mobile Phone SIM card
  - Can only be used after entering a PIN
- Algorithms (initially not public):
  - A3 for authentication
  - A8 for generating a session key
  - A5 for encrypting the communication
- A3 e A8 performed by the SIM card
  - Can be freely selected by the operator

GSM:
Mobile Phone authentication



GSM:
Mobile Phone authentication

- MSC fetches trio from HLR
  - **RAND, SRES, Kc**
  - In fact more than one are requested
- HLR generates RAND and corresponding trio using subscriber's Ki
  - RAND, random value (128 bits)
  - SRES = A3 (Ki, RAND) (32 bits)
  - Kc = A8 (Ki, RAND) (64 bits)
- Usually operators use COMP128 for A3/A8
  - Recommended by the GSM Consortium
  - [SRES, Kc] = COMP128 (Ki, RAND)

Host authentication

- By name or address
  - DNS name, IP address, MAC address, other
  - Extremely weak, no cryptographic proofs
    - Nevertheless, used by many services
    - e.g. NFS, TCP *wrappers*
- With cryptographic keys
  - Keys shared among peers
    - With an history of usual interaction
  - Per-host asymmetric key pair
    - Pre-shared public keys with usual peers
    - Certified public keys with any peer

Service / server authentication

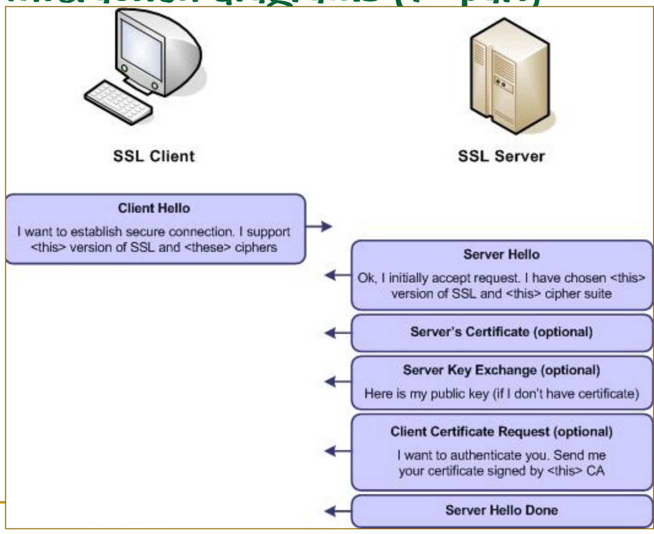
- Host authentication
  - All co-located services/servers are indirectly authenticated
- Per-service/server credentials
  - Shared keys
    - When related with the authentication of people
    - The key shared with each person can be used to authenticate the service to that person
  - Per-service/server asymmetric key pair
    - Certified or not

TLS (Transport Layer Security, RFC 2246): Goals

- Secure communication protocol over TCP/IP
  - Created from SSL V3 (Secure Sockets Layer)
  - Manages per-application secure sessions over TCP/IP
    - Initially conceived for HTTP traffic
    - Actually used for other traffic types
- Security mechanisms
  - Communication confidentiality and integrity
    - Key distribution
  - Authentication of communication endpoints
    - Servers (or, more frequently, services)
    - Client users
    - Both with asymmetric key pairs and certified public keys

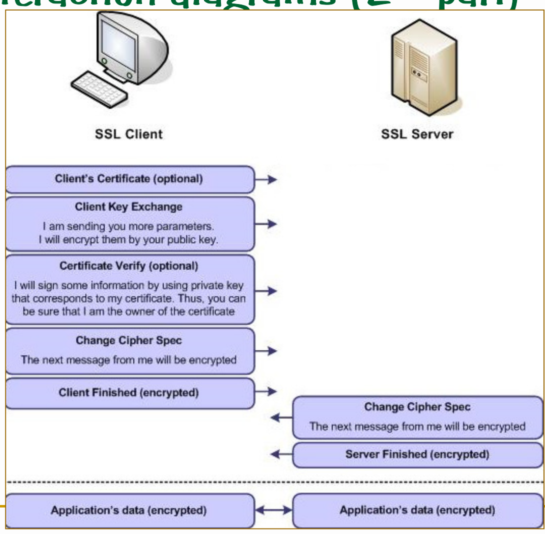
SSL/TLS:

Interaction diagrams (1st part)



SSL/TLS:

Interaction diagrams (2nd part)



SSH (Secure Shell): Goals

- Alternative to the Telnet protocol/application
  - Manages secure consoles over TCP/IP
  - Initially conceived to replace telnet
  - Actually used for other applications
    - Secure execution of remote commands (rsh/rexec)
    - Secure copy of contents between machines (rcp)
    - Secure FTP (sftp)
    - Creation of arbitrary secure tunnels (inbound/outbound/dynamic)
- Security mechanisms
  - Communication confidentiality and integrity
    - Key distribution
  - Authentication of communication endpoints
    - Servers / machines
    - Client users
    - Both with different techniques

SSH: Authentication mechanisms

- Server: with asymmetric keys pair
  - Inline public key distribution
    - Not certified!
  - Clients cache previously used public keys
    - Caching should occur in a trustworthy environment
    - Update of a server's key raises a problem to its usual clients
- Client users: configurable
  - Username + password
    - By default
  - Username + private key
    - Upload of public key in advance to the server

Authentication metaprotocols

- Generic authentication protocols that encapsulate other authentication specific protocols

- Examples
  - EAP (Extensible Authentication Protocol)
    - Used in 802.11 (Wi-Fi)
  - ISAKMP(Internet Security Association and Key Management Protocol)
    - Used in IPSec

Single Sign-On (SSO)

- Unique, centralized authentication for a set of federated services
  - The identity of a client, upon authentication, is given to all federated services
  - The identity attributes given to each service may vary
  - The authenticator is called Identity Provider (IdP)

- Examples
  - SSO authentication at UA
    - Performed by a central IdP (idp.ua.pt)
    - The identity attributes are securely conveyed to the service accessed by the user

Authentication services

- Trusted third parties (TTP) used for authentication
  - But often combined with other related functionalities

- AAA services
  - Authentication, Authorization and Accounting
  - e.g. RADIUS