INFORMATION SECURITY

General protection techniques (cont.): Authentication

Access to a computer

- user presents an identifier (name, *login*)
- system demands a confirmation (e.g. password matched to the identifier)

Remote communication

- party1 sends identifier to party2
- party2 challenges party1 with a fresh number that should be enciphered (e.g. with a predefined shared key)

Notice - *2 steps:*

- presentation (of subject)
- validation (proof of authenticity)

Authentication of subject: steps

- Step 1: presentation (of subject) [sometimes called: *identification*[1]]
- Step 2: validation (proof of authenticity) [sometimes called: *authentication*]

Definition of *authentication*

- binding of an identifier to a subject
  - or: certification of an user's identity
- sometimes: certification of a physical place
  - e.g. machine's location in the Net (origin of a communication)...
  - e.g. geographical location

[1] Note: this occasional use of “identification” is unfortunate. In reality, identification is the process of binding an identifier to an individual, as yet unknown (i.e. for whom no label, or name, was yet presented).

Authentication system's deployment

- setup phase [FIG]
  - generation and storage of subjects' authentication data in system
  - seldom repeated
    - e.g. when user changes his/her authentication data
- usage phase [FIG]
  - normal procedure for authentication of subjects
  - constantly repeated
    - e.g. when user daily enters a system

...Authentication system deployment (cont.)

Set up phase

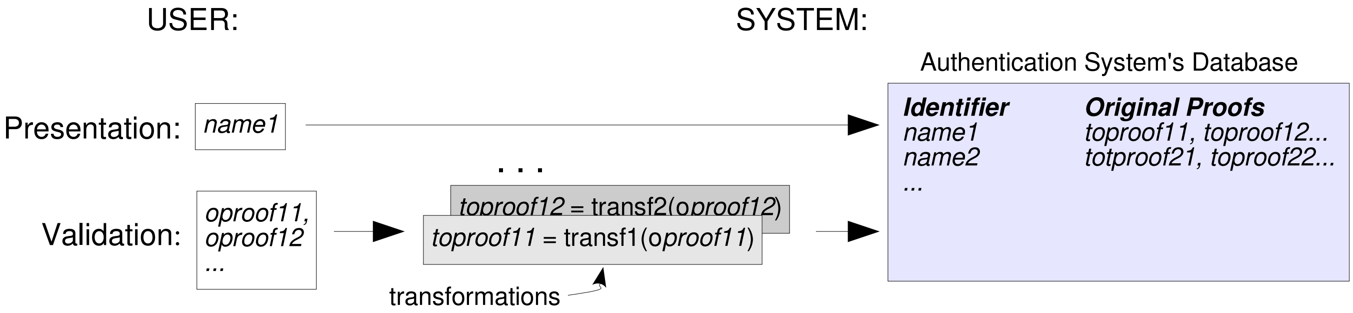


Fig. **Setting up** an authentication system: generation and storage of original proofs.
(Notice that what is usually stored are transformations of original proofs.)

Daily usage phase

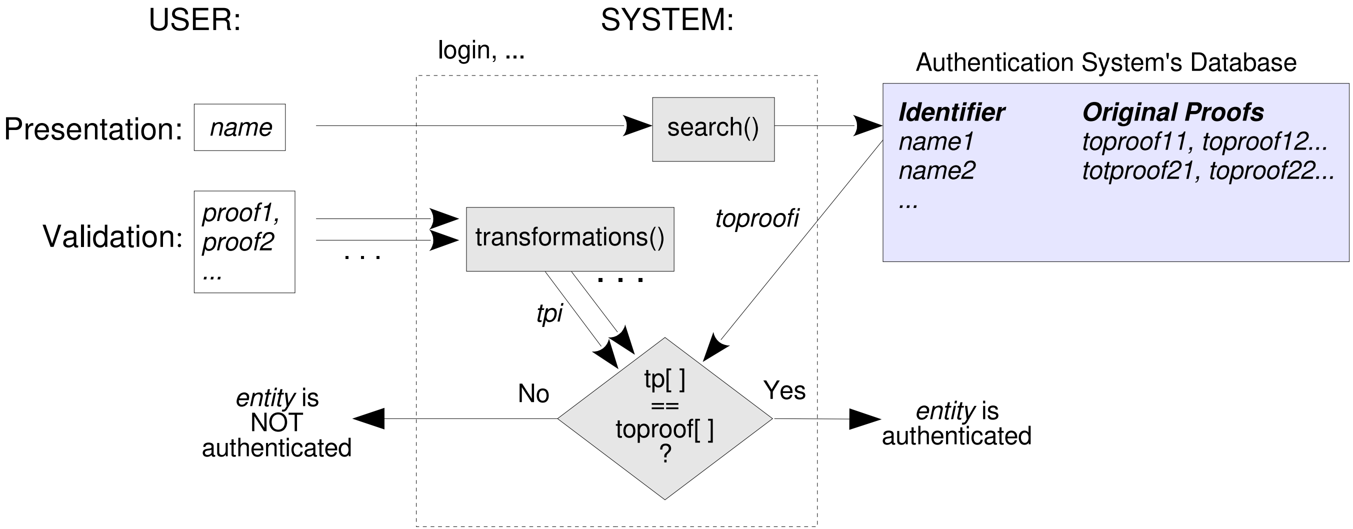
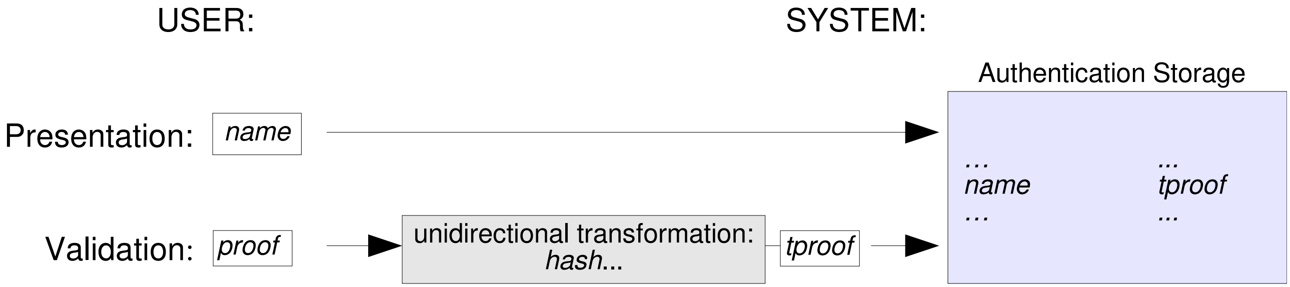


Fig. **Using** an authentication system: validating the proofs, comparing them with those stored in the setup phase.

Validation

- several possible methods (see below)
  - one or more may be necessary at a time
- needs an initial phase for populating the Authentication System's Database
  - must be repeated for each change of authentication data
- impersonation of users should be prevented
  - storage of unidirectional transformation of validation's proofs
    - `transf`*`i`*`(proof`*`i`*`)` in pictures!



Generation: unidirectional transformation of proofs.

Validation's methods

- proof by possession:
    - of knowledge: e.g. knowing a personal password
    - of object: e.g. having a personal card
    - of passive property: e.g. having a specific fingerprint
    - of ”active” property (trait): e.g. keying with a certain speed or hitting force
- proof by origin (...): e.g. request comes from a predefined machine or geographical place

Note on Terminology:

- in the literature, usually: proof by knowledge, by possession, by property, by trait correspond to the variants of *proof by possession* presented above.

Validation by proof of (possession of) knowledge

Memorizable information

- specially important in face-to-face authentication
- system demands (besides the presentation name, e.g. `loginname`):
  - presentation, e.g. `loginname`
    - validation, e.g. `password`
    - questions whose answers the user should know

Dynamic challenge-response exchange

- specially important in remote authentication
- system presents a <u>never seen before value</u> that the user has to:
  - (**secret algorithm**) - process[1] in a secret way and return the result
  - (**private key**) **-** process in a public way with a private key[2] and return the result

[1] usually by means of a computing device

[2] cryptographic key!

...Validation by proof of (possession of) knowledge (cont.)

Example of challenge-response exchange

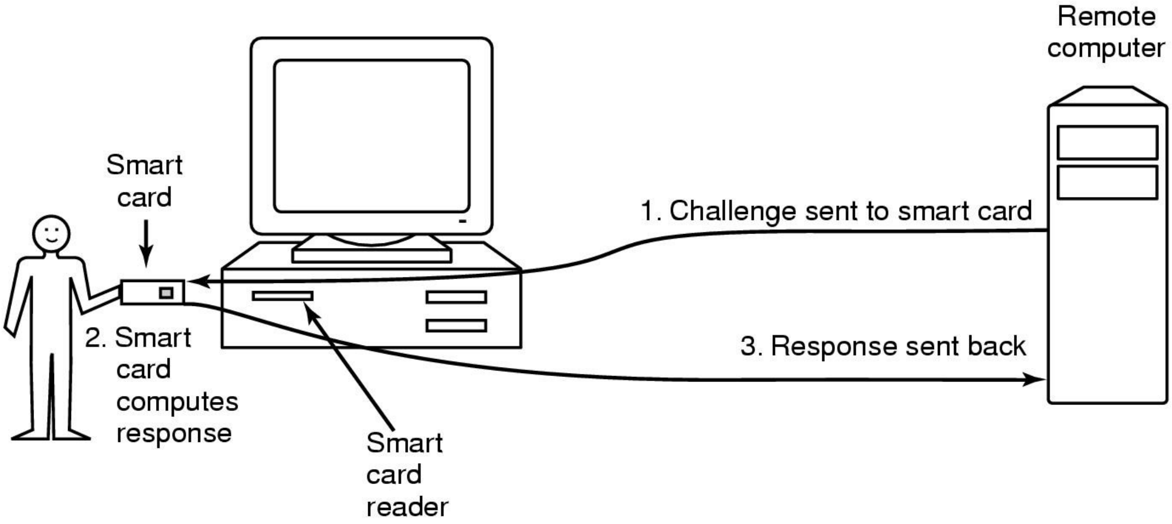
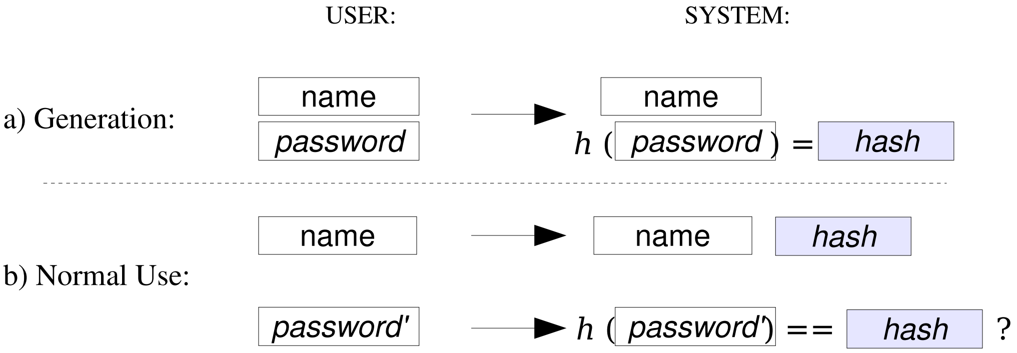


Fig. Example of challenge-response technique with a smart-card.

...Validation by proof of (possession of) knowledge (cont.)

Memorizable information: (secret) passwords

- individual proof (can be used by a group, but...)
- strength: difficulty of being guessed by someone
- weakness: easiness of (careless) disclosure by “owner” (e.g. writing down...)
- typical authentication (FIG): comparison of *hashes* (not of plain passwords!)



USER:

SYSTEM:

a) Generation:

name → name

password → *h* (*password*) = *hash*

b) Normal Use:

name → name *hash*

password' → *h* (*password'*) == *hash* ?

...Memorizable information: (secret) passwords

Use of passwords: typical attacks

- simple guessing (trial and error)
- educated guessing (use of a dictionary or social information)
- utilization of old, but still active, passwords
- social engineering (e.g. phishing)

...Memorizable information: (secret) passwords

Use of passwords: technical protections

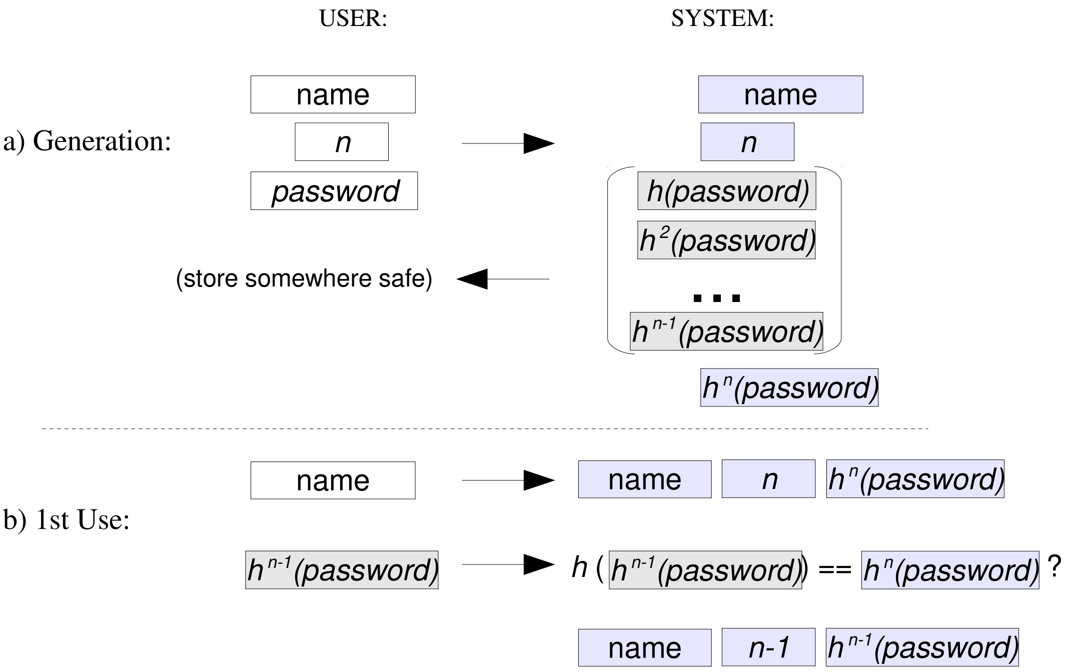
- classical:
  - burden on system:
    - increase of authentication's difficulty (e.g. calculation times)
    - detection of failed authentication attempts
  - burden on user:
    - periodic change passwords (*password aging*)
    - no re-usage of previous passwords (*password logging*)
    - use of “unguessable” passwords (e.g. `%/tkP6qL*bx«`)
- different:
  - one-time passwords (not so memorizable...)
  - avoidance of repeated authentications: Single Sign-On (SSO)

One-time passwords*

- passwords can be used just once
  - static or dynamic generation
- user and system must agree on each and every password
  - (**static generation**) - both have a list (kept on paper or in electronic device)
  - (**dynamic generation**) – both have means of password generation (could be a problem for user)
- Example: *Lamport's hash*! [FIG]
  - Implementations:
    - *OTP System*, IETF STD 61 (orig.: *S/Key System*, RFC1760)
    - OPIE, *One time Passwords In Everything*, Unix-like package

* PT: senhas de utilização única (ou descartáveis)

The Lamport's system of **one-time passwords**



USER:

SYSTEM:

a) Generation:

name

n

password

name

n

h(password)

$h^{2}(password)$

...

$h^{n-1}(password)$

$h^{n}(password)$

(store somewhere safe)

b) 1st Use:

name

name *n* $h^{n}(password)$

$h^{n-1}(password)$

$h\,(h^{n-1}(password)) == h^{n}(password)$?

name *n-1* $h^{n-1}(password)$

Single Sign-On, SSO*

- single, initial authentication for all sessions on all machines
- allows use of cryptographic keys
- possible implementations:
  - *password wallet*
  - federated authentication
- problems:
  - safe keeping of single password (even with wallets!)
  - session hijacking
    - partial solution: periodical new authentication!

Exercise:

Explain if the computer system of FEUP has Single Sign-on. Present some shortcomings of the current system.

* PT: autenticação única

Validation by proof of (possession of) property

- desirable when user is physically present
- verification of
  - fingerprints
  - eyes (iris or retina)
  - palm (lines or veins)
  - voice
  - facial features
  - keyboard use (proof by trait…)
  - ...
- Problems:
  - false positives and false negatives!
  - intrusive or potentially dangerous methods!

Validation by proof of origin

- detect the computer from where the authentication is being attempted
  - e.g. does it belong to the local network?
- detect the geographical position of subject (and computer) from where the authentication is being attempted
  - e.g. by Global Positioning System, GPS

Multi-factor validation

- combine different techniques!
  - e.g. two-factor authentication: PIN[1] + physical card
- general validation rule!

[1] Personal Identification Number

Authentication protocols: (dynamic) challenge-response

- important where user's physical intervention is not possible or required
  - e.g. remote communication
- proof of knowledge, typically of challenge-response type
- based on the use of pre-distributed keys
- generally use *nonces*

Nonce

- piece of data that is both:
  - fresh
  - not guessable (random)
- normally, is random number generated when about to be used
- binds two messages in a challenge-response sequence

...Authentication protocols: (dynamic) challenge-response (cont.)

Mutual authentication by (preset) secret, shared key

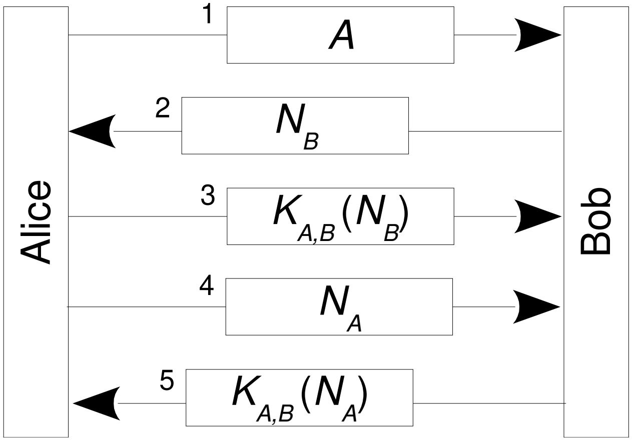


Fig. Authentication protocol by shared key ($K_{A,B}$): N_A and N_B are *nonces*.

...Authentication protocols: (dynamic) challenge-response (cont.)

Mutual authentication by (preset) public keys

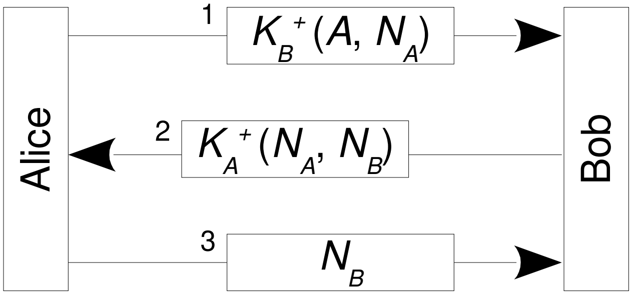


Fig. Authentication protocol by public(s) key(s) (K_A^+ and K_B^+).

...Authentication protocols: (dynamic) challenge-response (cont.)

Authentication by keys: problems

- each subject must keep a key (secret or public) of each of his/her partners
- protocols assume a pre-distribution of needed keys
  - possible solution: use of Key Servers (Key Distribution Centers)
- whoever has <u>the</u> key is <u>the</u> person!

The user identity in the digital society

- Problem with the unequivocal identification of entity...
- Ease of proof forgery (look at spy and sci-fi movies...)
  - and of impersonation if stored authentication data is **exactly** the proof!
- Exacerbation caused by the “virtual” (& remote) interaction with user
- Cryptography is no absolute solution
  - even with public key system (unambiguous pinpoint of entity)
    - (whoever has the key is the person!)
- Future?...