Authentication Mechanisms

INF-744: Security and Privacy for IoT

Diego F. Aranha

Institute of Computing – University of Campinas

Introduction

Parties need to establish and verify if they are communicating with other legitimate parties.

Authentication is important:

- Identify the other party (*Identification*)
- Allow access to resources
- Access control (*Authorization*)

Important: There are many different ways of achieving this!

User authentication

There are three types of *authentication factors*:

1. Something you know. Ex:

User authentication

There are three types of *authentication factors*:

1. Something you know. Ex: password, passphrase
2. Something you have. Ex:

User authentication

There are three types of *authentication factors*:

1. Something you know. Ex: password, passphrase
2. Something you have. Ex: cryptographic key/token, smart card
3. Something you are. Ex:

User authentication

There are three types of *authentication factors*:

1. Something you know. Ex: password, passphrase
2. Something you have. Ex: cryptographic key/token, smart card
3. Something you are. Ex: fingerprint, voice, iris (biometrics)

There are also many advantages and disadvantages to these options!

Multifactor and mutual authentication

Authentication factors can be combined:

- Typical ATM machine: 2-factor authentication:
  1. Bank-issued card or one-time password
  2. Password

  Important: Modern ATMs also have biometric devices.

- Brazilian voting machine: 2-factor authentication:
  1. Government-issued document
  2. Biometrics (picture and fingerprint)

Mutual authentication

Ideally, multiple parties should be authenticated to all other parties.

Password authentication

Algorithm:

1. Server keeps a database of (client, password)
2. Server asks the client and password at every access
3. To authenticate, index the database with the client name and verify password.
4. Block user if too many incorrect attempts

Advantages:

Password authentication

Algorithm:

1. Server keeps a database of (client, password)
2. Server asks the client and password at every access
3. To authenticate, index the database with the client name and verify password.
4. Block user if too many incorrect attempts

Advantages:

- Convenient and familiar to most users
- Efficient in terms of computation and storage
- Easy to deploy and integrate (OAUTH single sign-on)

Disadvantages:

Password authentication

Algorithm:

1. Server keeps a database of (client, password)
2. Server asks the client and password at every access
3. To authenticate, index the database with the client name and verify password.
4. Block user if too many incorrect attempts

Advantages:

- Convenient and familiar to most users
- Efficient in terms of computation and storage
- Easy to deploy and integrate (OAUTH single sign-on)

Disadvantages:

- Passwords can be **reused** (data breaches...)
- Passwords can be **weak**
- Hard to measure if a password is good

Password authentication

Password storage

Password storage is a classical common source of security problems. Attackers can obtain password file and impersonate users after some effort.

Problem: Should passwords be stored in plaintext?

Password authentication

Password storage

Password storage is a classical common source of security problems. Attackers can obtain password file and impersonate users after some effort.

Problem: Should passwords be stored in plaintext?

Problem: Should passwords be encrypted?

Password authentication

Password storage

Password storage is a classical common source of security problems. Attackers can obtain password file and impersonate users after some effort.

Problem: Should passwords be stored in plaintext?

Problem: Should passwords be encrypted?

Solution: No, password p should be processed through a one-way function and $H(p)$ should be stored instead!

Password authentication

Problem: Users pick the same passwords. Adversary with idle computing power can iterate over many choices of p and “invert” H using a big table of common inputs.

Password authentication

Problem: Users pick the same passwords. Adversary with idle computing power can iterate over many choices of p and “invert” H using a big table of common inputs.

Solution: Salt password hashes and store per-user random **salt** s and store $H(s, p)$.

Password authentication

Problem: Users pick the same passwords. Adversary with idle computing power can iterate over many choices of p and "invert" H using a big table of common inputs.

Solution: Salt password hashes and store per-user random **salt** s and store $H(s, p)$.

Problem: GPUs, FPGAs and ASICs allow **online** attacks! (Check recommended reading about breaking `qeadzcwrsfxv1331`)

Password authentication

Problem: Users pick the same passwords. Adversary with idle computing power can iterate over many choices of p and "invert" H using a big table of common inputs.

Solution: Salt password hashes and store per-user random **salt** s and store $H(s, p)$.

Problem: GPUs, FPGAs and ASICs allow **online** attacks! (Check recommended reading about breaking `qeadzcwrsfxv1331`)

Solution: Use expensive key derivation functions (PBKDF2, `bcrypt`) and lots of storage (`scrypt`, Argon2). If possible, compute a MAC! (Check recommended reading about **pepper**)

One-time passwords

Problem: Passwords may leak or be stolen by the adversary.

One-time passwords (OTP)

OTP schemes require a shared password and compute a different password for every authentication attempt.

Common solutions:

- List of passwords
- Password cards
- Compute passwords using keyed function
- SMS message (**not recommended anymore**)

Important: Pay attention if different authentication factors are stored in different places, otherwise they are the same factor!

S/Key authentication

S/Key is a OTP scheme which produces a limited number of authentications using a one-way function f.

Setup:

1. Pick a random number R
2. Compute $x_1 = f(R)$
3. Compute $x_2 = f(x_1) = f(f(R))$
4. Compute $x_i = f^i(R)$
5. Compute and store $x_n = f(x_{n-1})$

S/Key authentication

S/Key is a OTP scheme which produces a limited number of authentications using a one-way function f.

Setup:

1. Pick a random number R
2. Compute $x_1 = f(R)$
3. Compute $x_2 = f(x_1) = f(f(R))$
4. Compute $x_i = f^i(R)$
5. Compute and store $x_n = f(x_{n-1})$

Usage:

1. Store x_n in a password file
2. Alice presents x_{n-1} as password
3. Host checks if $x_n = f(x_{n-1})$
4. If successful, replace x_n with x_{n-1}

Password managers

Problem: Users share the same passwords! A data breach may allow an attacker to impersonate users.

Solution: Use a **password manager**!

Advantages:

Password managers

Problem: Users share the same passwords! A data breach may allow an attacker to impersonate users.

Solution: Use a **password manager**!

Advantages:

- Passwords can be generated at random (**ideal**)
- Data breaches have no impact if master password is strong. **Suggestion:** Diceware method.
- Sometimes incompatible with website requirements

Disadvantages:

Password managers

Problem: Users share the same passwords! A data breach may allow an attacker to impersonate users.

Solution: Use a **password manager**!

Advantages:

- Passwords can be generated at random (**ideal**)
- Data breaches have no impact if master password is strong. **Suggestion:** Diceware method.
- Sometimes incompatible with website requirements

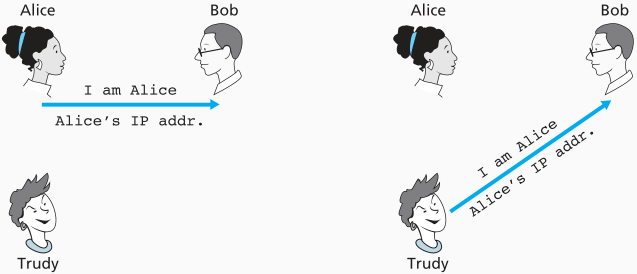
Disadvantages:

- All passwords can be computed from **master password**
- Cloud-based synchronization may be risky
- Password managers can be vulnerable

Problem: How to authenticate over the network?

Authentication protocol

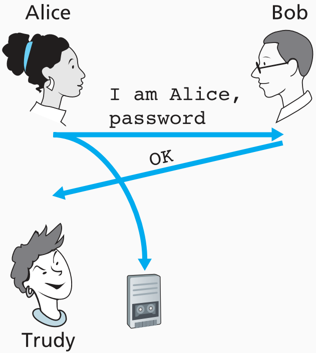
Protocol 1: Simple protocol using IP address.

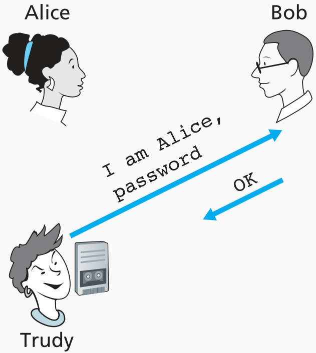


Important: What attacks are possible?

Authentication protocol

Protocol 2: Password-based authentication.

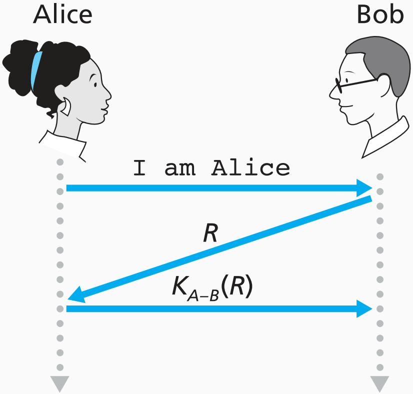




Important: What attacks are possible?

Authentication protocol

Protocol 3: Challenge-response authentication using symmetric keys.



Problem: How can we improve protocol by using asymmetric cryptosystem?

Cryptographic authentication

Algorithm:

1. Client generates cryptographic keys
2. Server stores public key or symmetric secret key
3. Server generates and sends a challenge at every access
4. To authenticate, verify client response
5. Block user if too many incorrect attempts

Advantages:

Cryptographic authentication

Algorithm:

1. Client generates cryptographic keys
2. Server stores public key or symmetric secret key
3. Server generates and sends a challenge at every access
4. To authenticate, verify client response
5. Block user if too many incorrect attempts

Advantages:

- Convenient and lightweight for machine authentication
- Compatible to symmetric primitives (challenge-response)
- Compatible to asymmetric primitives (smart card)

Disadvantages:

Cryptographic authentication

Algorithm:

1. Client generates cryptographic keys
2. Server stores public key or symmetric secret key
3. Server generates and sends a challenge at every access
4. To authenticate, verify client response
5. Block user if too many incorrect attempts

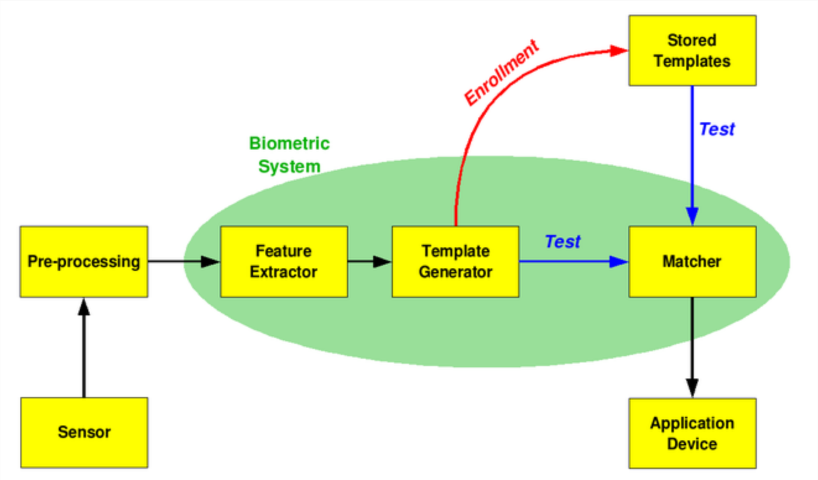
Advantages:

- Convenient and lightweight for machine authentication
- Compatible to symmetric primitives (challenge-response)
- Compatible to asymmetric primitives (smart card)

Disadvantages:

- Beware of common pitfalls in cryptography
- Impossible for humans to handle, additional device
- Physical device can be stolen
- Revoking keys may be harder than changing passwords

Biometric authentication



Biometric authentication

Algorithm:

1. Server keeps a database of (client, biometric data)
2. Client participates in the **enrollment**
3. To authenticate, index the database with the client id and verify fresh biometric.

Many types of biometric information are used:

1. **Intrinsic:** Fingerprint, palm veins and shape, face recognition, iris and retina
2. **Behavioral:** typing rhythm, gait, voice

Biometric authentication

Functionality assumptions/requirements:

1. **Universality**:

Biometric authentication

Functionality assumptions/requirements:

1. **Universality**: all users must have it
2. **Uniqueness**:

Biometric authentication

Functionality assumptions/requirements:

1. **Universality**: all users must have it
2. **Uniqueness**: trait must be unique
3. **Permanence**:

Biometric authentication

Functionality assumptions/requirements:

1. **Universality**: all users must have it
2. **Uniqueness**: trait must be unique
3. **Permanence**: must be preserved with time
4. **Measurability**:

Biometric authentication

Functionality assumptions/requirements:

1. **Universality**: all users must have it
2. **Uniqueness**: trait must be unique
3. **Permanence**: must be preserved with time
4. **Measurability**: trait must be measurable
5. **ACcuracy**:

Biometric authentication

Functionality assumptions/requirements:

1. **Universality**: all users must have it
2. **Uniqueness**: trait must be unique
3. **Permanence**: must be preserved with time
4. **Measurability**: trait must be measurable
5. **ACcuracy**: process should be accurate and robust
6. **Acceptability**:

Biometric authentication

Functionality assumptions/requirements:

1. **Universality**: all users must have it
2. **Uniqueness**: trait must be unique
3. **Permanence**: must be preserved with time
4. **Measurability**: trait must be measurable
5. **ACcuracy**: process should be accurate and robust
6. **Acceptability**: users should accept it
7. **Circumvention**:

Biometric authentication

Functionality assumptions/requirements:

1. **Universality**: all users must have it
2. **Uniqueness**: trait must be unique
3. **Permanence**: must be preserved with time
4. **Measurability**: trait must be measurable
5. **ACcuracy**: process should be accurate and robust
6. **Acceptability**: users should accept it
7. **Circumvention**: how hard to replace it

Biometric authentication

Advantages:

Biometric authentication

Advantages:

- Highly usable for humans, if precision is enough
- Security requirements are intuitive

Disadvantages:

Biometric authentication

Advantages:

- Highly usable for humans, if precision is enough
- Security requirements are intuitive

Disadvantages:

- Beware of invalid assumptions!
- Credentials can never be revoked or replaced
- Highly intrusive to privacy
- **Probabilistic!** (beware of false positives and false negatives)
- What happens in case of a data breach? (**legal framework**)