Authentication, Authorization and Access Control

19CSE311 Computer Security

Jevitha KP

Department of CSE

Entity authentication

- Entity authentication is a technique designed to let one party prove the identity of another party.

- An **entity** can be a person, a process, a client, or a server.

- 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**.

- When Bob tries to prove the identity of Alice, Alice is the **claimant**, and Bob is the **verifier**.

Data-Origin Versus Entity Authentication

- **Message authentication (or data-origin authentication)** might not happen in real time; entity authentication does.

    - In **Message authentication**, Alice sends a message to Bob. When Bob authenticates the message, **Alice may or may not be present in the communication process.**

        - When Alice requests **entity authentication**, there is no real message communication involved until Alice is authenticated by Bob. **Alice needs to be online and to take part in the process.** Only after Alice is authenticated can messages be communicated between Alice and Bob.

        - Eg: Data-origin authentication is required, when an email is sent from Alice to Bob. Entity authentication is required, when Alice gets cash from an automatic teller machine.

- Second, message authentication simply authenticates **one message;** the process needs to be repeated for **each new message**. Entity authentication **authenticates the claimant** for the **entire duration of a session.**

Verification Categories (Authentication Factors)

- In entity authentication, the claimant must identify herself to the verifier. This can be done with one of three kinds of witnesses:

- **Something known:** This is a **secret known only by the claimant** that can be checked by the verifier. Examples are a password, a PIN, a secret key, and a private key.

- **Something possessed:** This is **something that can prove the claimant's identity**. Examples are a passport, a driver's license, an identification card, a credit card, and a smart card.

- **Something inherent:** This is an **inherent characteristic of the claimant.** Examples are conventional signatures, fingerprints, voice, facial characteristics, retinal pattern, and handwriting

Authentication Schemes

- **Password based authentication**
  - **Fixed Password**
  - **One-time Password**
- Challenge-Response Authentication
- Zero-Knowledge Protocols
- Biometrics

Password based Authentication

- The simplest and oldest method of entity authentication is the password-based authentication, where the password is something that the **claimant knows.**

- A password is used when a user needs to access a system to use the system's resources (login).

- Each user has a **user identification that is public, and a password that is private**.

- We can divide these authentication schemes into two groups:

  - the **fixed password** and

  - the **one-time password.**

Fixed Password

- A fixed password is a password that is used over and over again for every access.

- Three approaches are discussed based on fixed passwords

- Multifactor Authentication approach

1 - User Id and Password File

- In the very rudimentary approach, the system keeps a table (a file) that is sorted by user identification.

- To access the system resources, the user sends her user **identification** and **password**, in **plaintext**, to the system.

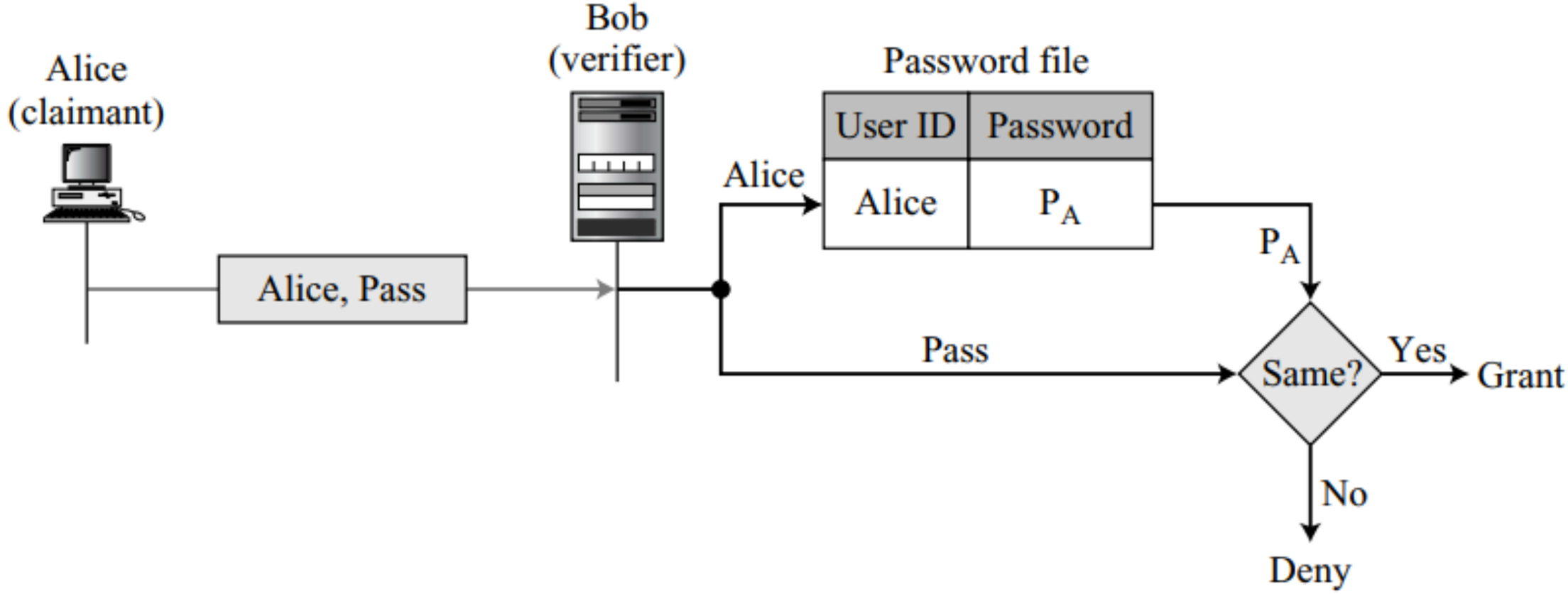
- The system uses the **identification** to find the **password** in the table.

- If the password sent by the user matches the password in the table, access is granted; otherwise, it is denied.

User id and Password file

P_A: Alice's stored password

Pass: Password sent by claimant



Attacks

- **Eavesdropping.**
    - Eve can watch Alice when she types her password.
    - As a security measure, do not show the characters a user types.
    - Eavesdropping can be more sophisticated form by listening to the line and intercept the message, thereby capturing the password.

Attacks

- **Stealing a password.**
    - The second type of attack occurs when Eve tries to **physically steal Alice's password.**
    - This can be prevented if Alice does not write down the password and instead she just retains in memory.
    - For this reason the password should be very simple or else related to something familiar to Alice.
    - But this makes the password vulnerable to other types of attacks.

Attacks

- **Accessing a password file.**
  - Eve can hack into the system and get access to the ID/ password file.
  - Eve can read the file and find Alice’s password or even change it.
  - To prevent this type of attack, the file can be read/write protected.
  - However, most systems need this type of file to be readable.

Attacks

- **Guessing.**
    - Using a guessing attack, Eve can log into the system and try to guess Alice's password by **trying different combinations of characters.**
    - The password is particularly vulnerable if the user is allowed to choose a **short password** (a few characters).
    - It is also vulnerable if Alice has chosen something **trivial**, such as her birthday, her child's name, or the name of her favorite actor.
    - To prevent guessing, **a long random password is recommended**, something that is not very obvious.
    - Random password - people forget such a password, Alice might store a copy of it somewhere, which makes the password subject to stealing.

2 - Hashing the password

- A more secure approach is to store the hash of the password (instead of the plaintext password) in the password file.

- Any user can read the contents of the file, but, because the hash function is a one-way function, it is almost impossible to guess the value of the password.

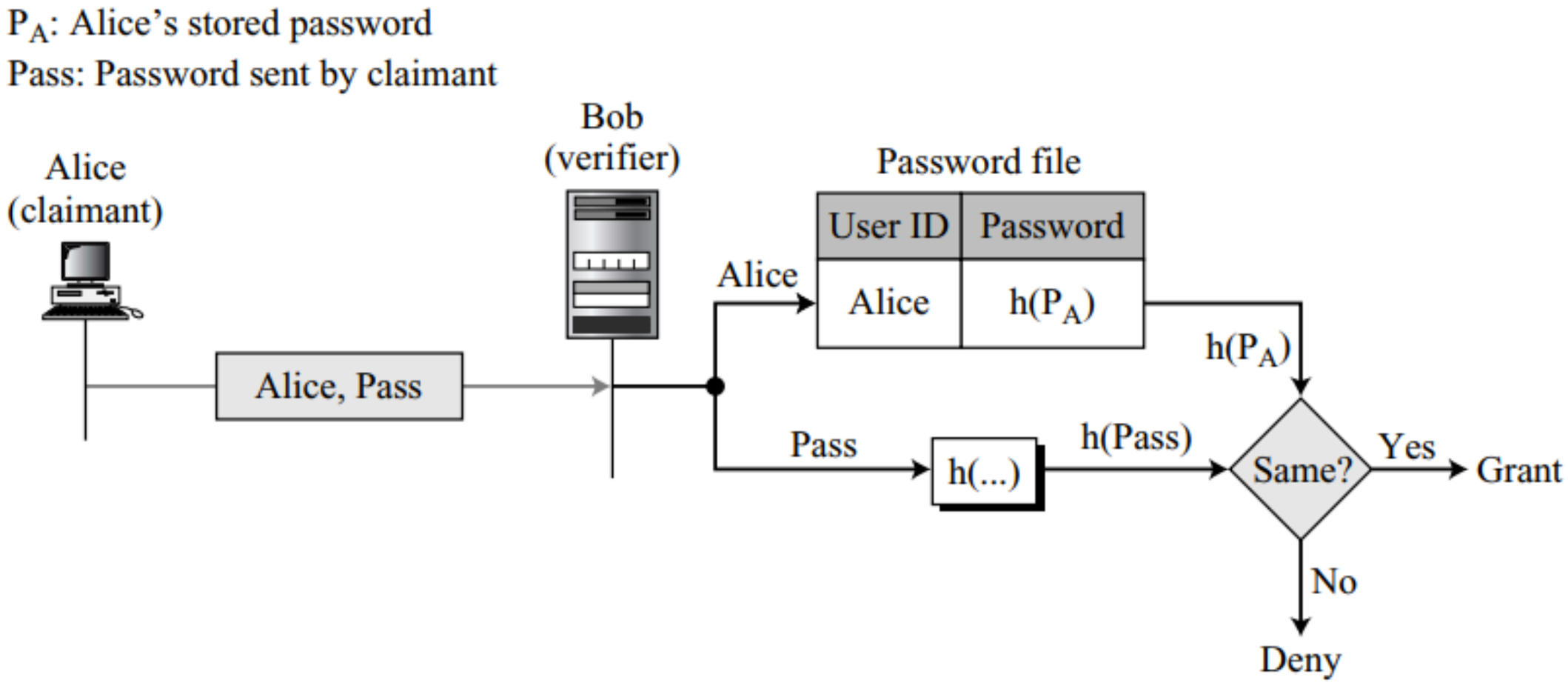
- When the password is created, the system hashes it and stores the hash in the password file.

- When the user sends the ID and the password, the system creates a hash of the password and then compares the hash value with the one stored in the file.

- If there is a match, the user is granted access; otherwise, access is denied.

- In this case, the file does not need to be read protected.

Hashing the password



Attacks

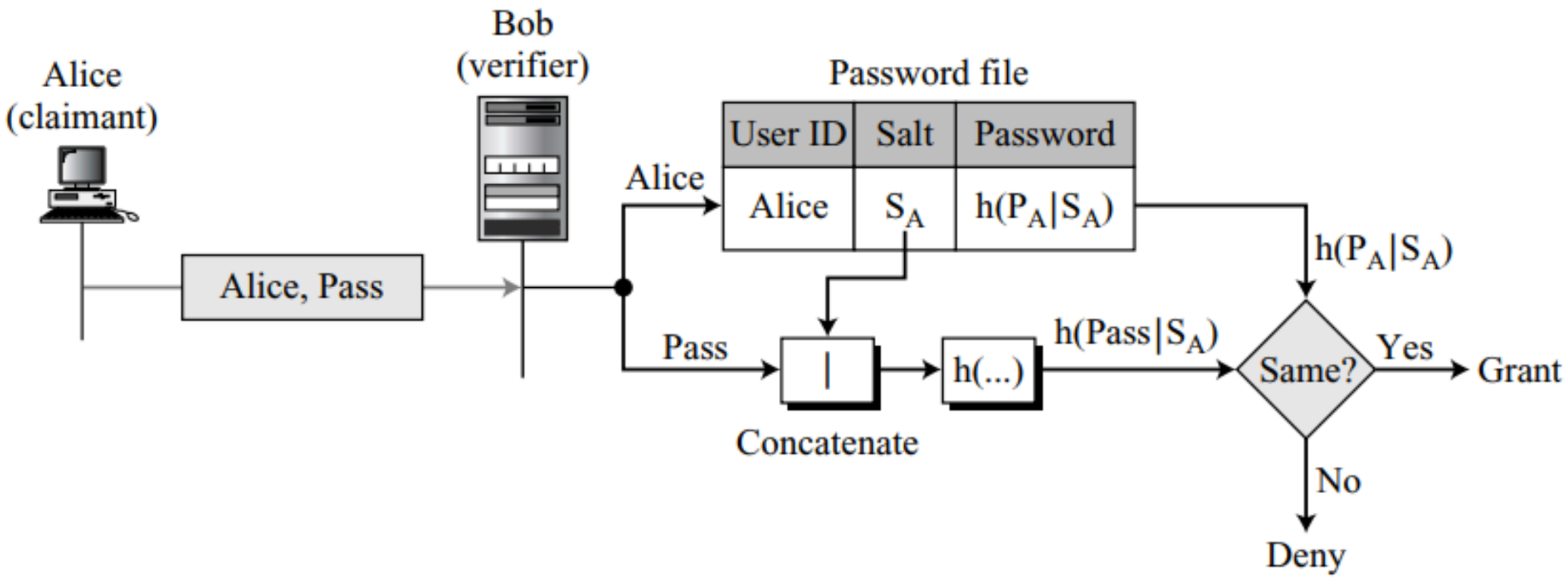
- **Dictionary Attack**
- The hash function prevents Eve from gaining access to the system even though she has the password file.
- But there is a possibility of dictionary attack.
- In this attack, Eve is interested in finding one password, regardless of the user ID.
- For example, if the password is 6 digits, Eve can create a list of 6-digit numbers (000000 to 999999), and then apply the hash function to every number; the result is a list of **one million hashes.**
- She can then get the password file and search the second-column entries to find a match.
- This could be programmed and run offline on Eve's private computer.
- After a match is found, Eve can go online and use the password to access the system.
- Next approach shows how to make this attack more difficult.

3 - Salting the password

- The third approach is called **salting** the password.
- When the password string is created, a random string, called the **salt**, is **concatenated to the password.**
- **The salted password is then hashed.**
- The ID, the salt, and the hash are then stored in the file.
- Now, when a user asks for access, **the system extracts the salt, concatenates it with the received password**, **makes a hash out of the result**, and compares it with the hash stored in the file.
- If there is a match, access is granted; otherwise, it is denied.

Salting the password

P_A: Alice's password
S_A: Alice's salt
Pass: Password sent by claimant



Attacks

- Salting makes the dictionary attack more difficult.
- If the original password is 6 digits and the salt is 4 digits, then hashing is done over a 10-digit value.
- This means that Eve now needs to make a list of 10 million items and create a hash for each of them.
- The list of hashes has 10 million entries, and the comparison takes much longer.
- Salting is very effective, if the salt is a very long random number.
- The UNIX operating system uses a variation of this method.

Multi factor Authentication

- In this approach, **two identification techniques** are **combined.**
- A good example of this type of authentication is the use of an ATM card with a PIN (personal identification number).
- The card belongs to the category **“something possessed ”** and the PIN belongs to the category **“something known”.**
- The PIN is a password that enhances the security of the card.
- If the card is stolen, it cannot be used unless the PIN is known.
- The PIN number, however, is traditionally very short so it is easily remembered by the owner.
- This makes it vulnerable to the **guessing type of attack.**

One-Time Password

- A one-time password is a password that is used only once.

- This kind of password makes eavesdropping and salting useless.

- Three approaches are discussed.

First Approach

- In the first approach, the user and the system agree upon a **list of passwords**.
- **Each password on the list can be used only once.**
- There are some drawbacks to this approach.
    - First, the system and the user must keep a long list of passwords.
    - Second, if the user does not use the passwords in sequence, the system needs to perform a long search to find the match.
- This scheme makes eavesdropping and reuse of the password useless.
- The **password is valid only once and cannot be used again**

Second Approach

- In the second approach, the user and the system agree to **sequentially update the password.**

- The user and the system agree on an original password, P1, which is valid only for the first access.

- During the first access, the user generates a new password, P2, and encrypts this password with P1 as the key. P2 is the password for the second access.

- During the second access, the user generates a new password, P3, and encrypts it with P2; P3 is used for the third access.

- In other words, **Pi is used to create Pi+1.**

- If Eve can guess the first password (P1), she can find all of the subsequent ones.

Third Approach - Lamport one-time Password

- In the third approach, the user and the system create a **sequentially updated password** using a hash function.

- This was devised by Leslie Lamport.

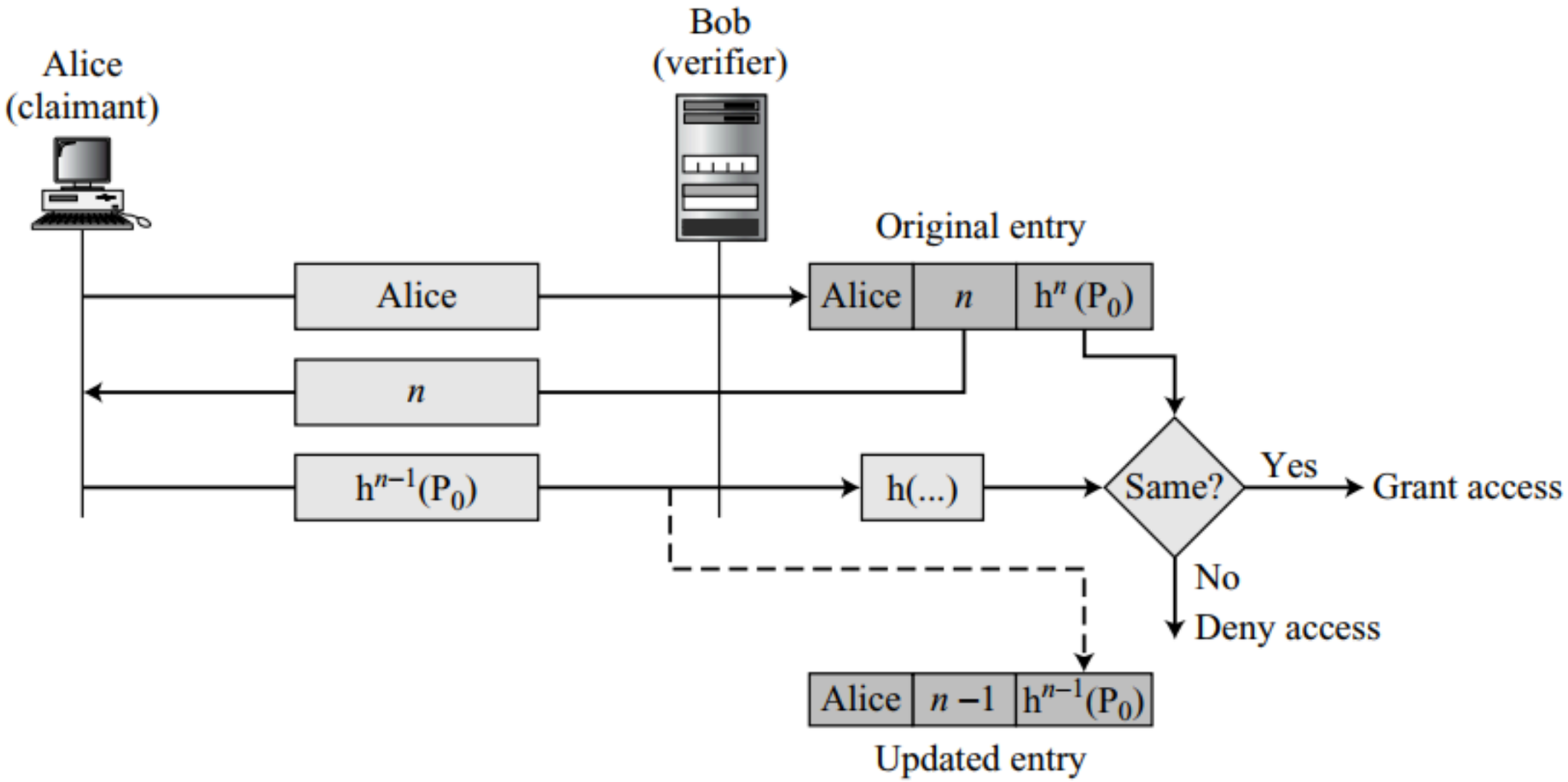
- Here, the user and the system agree upon an **original password, P0, and a counter, n.**

- The system calculates **$h^n(P0)$**, where **h^n** means **applying a hash function n times.**

- The system **stores the identity of Alice, the value of n,** and the **value of $h^n(P0)$**

$$h^n(x) = h(h^{n-1}(x)) \quad h^{n-1}(x) = h(h^{n-2}(x)) \quad \ldots \quad h^2(x) = h(h(x)) \quad h^1(x) = h(x)$$

Lamport one-time password



Lamport one-time password

- When the system receives the response of the user in the third message, **it applies the hash function to the value received** to see if it matches the value stored in the entry.

- If there is a match, access is granted; otherwise, it is denied.

- The system then decrements the value of n in the entry and replaces the old value of the password **$h^n(P0)$** with the new value **$h^{n-1}(P0)$**.

- When the user tries to access the system for the second time, the value of the counter it receives is n – 1. The third message from the user is now **$h^{n-2}(P0)$.**

- When the system receives this message, **it applies the hash function** to get **$h^{n-1}(P0)$**, which can be compared with the updated entry.

- **The value of n in the entry is decremented each time there is an access.**

- When the value becomes 0, the user can no longer access the system; everything must be set up again.

- For this reason, the value of n is normally chosen as a large number such as 1000.

Password based Authentication Disadvantages

- In password authentication, the claimant needs to send her secret (the password) to the verifier; this is subject to eavesdropping by Eve.

- In addition, a dishonest verifier could reveal the password to others or use it to impersonate the claimant.

CHALLENGE-RESPONSE

- In challenge-response entity authentication, the claimant's secret is not sent to the verifier.
- The claimant proves that she knows a secret without sending it.
- The claimant applies a function on the challenge sent by the verifier that includes her secret.
- In some challenge-response methods, the verifier actually knows the claimant's secret, which could be misused by a dishonest verifier.
- In other methods, the verifier can extract some information about the secret from the claimant by choosing a preplanned set of challenges.
- Techniques based on symmetric-key cipher, asymmetric-key cipher, keyed-hash functions, digital signatures.

ZERO-KNOWLEDGE

- In zero-knowledge authentication, the claimant does not reveal anything that might endanger the confidentiality of the secret.

- The claimant proves to the verifier that she knows a secret, without revealing it.

- The interactions are so designed that they cannot lead to revealing or guessing the secret.

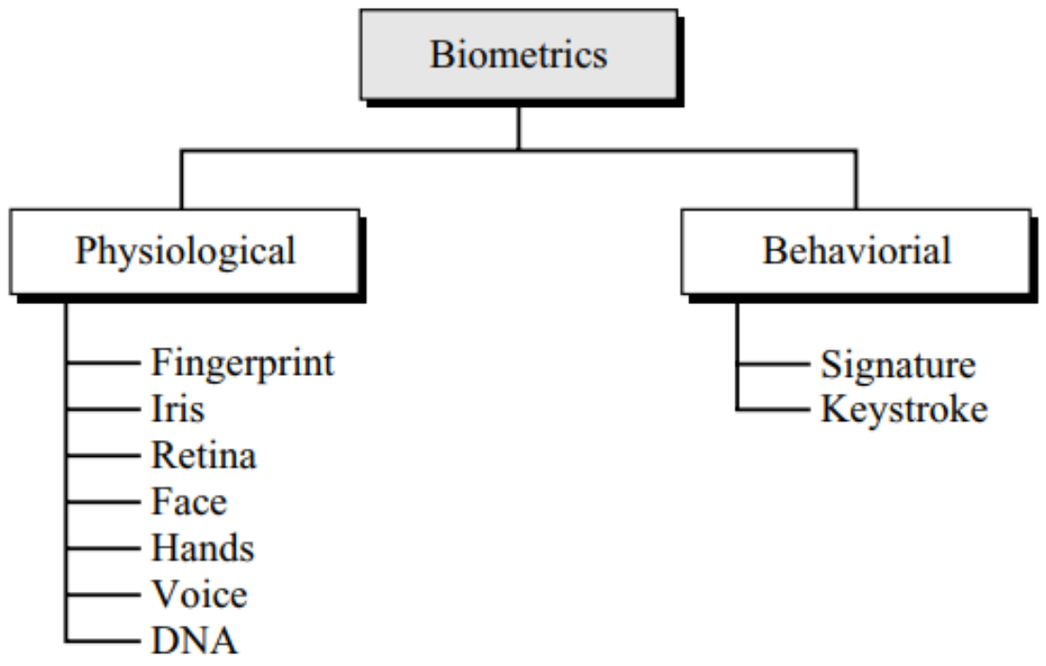
- After exchanging messages, the **verifier only knows that the claimant does or does not have the secret**, nothing more.

- The result is a **yes/no** situation, just a single bit of information.

- Eg: Fiege-Shamir protocol, Feige-Fiat-Shamir Protocol, Guillou-Quisquater Protocol.

Biometrics

- Biometrics is the measurement of physiological or behavioral features that identify a person **(authentication by something inherent).**
- Biometrics measures features that cannot be guessed, stolen, or shared.



Authorization

Authorization

- Authorization is the process of giving someone **the ability to access a resource.**

- It the process of verifying that a **requested action or service is approved** for a specific entity.

- It the function of specifying access rights/privileges to resources, which is related to general information security and computer security, and to access control in particular.

- More formally, "to authorize" is to **define an access policy.**

Authorization

- Authorization is distinct from authentication which is the process of verifying an entity's identity.

- When designing and developing a software solution, it is important to keep these distinctions in mind.

- A user who has been authenticated is often **not authorized to access every resource** and perform every action that is possible through a system.

- For example, a web app may have both **regular users and admins**, with the admins being able to perform actions the average user is not privileged to do so, even though have been authenticated.

- Authentication is not always required for accessing resources; an unauthenticated user may be authorized to access certain public resources, such as an image or login page, or even an entire web app.

Authorization

- Eg: In house ownership, the owner has **full access rights** to the property (**the resource**) but can **grant other people the right to access it**.

- We say that the owner **authorizes** people to access it.

- For instance, accessing the house is a **permission**, that is, an **action that we can perform** on a **resource**.

- Other permissions on the house may be furnishing it, cleaning it, repair it, etc.

Authorization

- A permission becomes a **privilege (or right)** when it is assigned to someone.

- So, if you assign permission to furnish your house to your interior decorator, you are granting them that **privilege**.

- On the other hand, the decorator may ask you permission to furnish your house. In this case, the **requested permission is a scope**, that is, the action that the decorator would like to perform at your house.

Authorization

- In computer systems, authorization rules are part of an IT discipline called **Identity and Access Management (IAM).**

- Within IAM, **authorization and authentication** help system managers to control who has access to system resources and set client privileges.

- The way that IT systems deal with **authorization services** is very similar to a real-world access control process.

Authorization

- **Authorization Use Case**

- Consider a collaboration tool like Google Docs.

- The application allows you to **create and share** documents**.** Other permissions include being able to **update, delete, comment** on a document.

- If you are the owner of a document, you can share it with someone else and define one or more **access policies.** For example, you can share your document with someone by letting them just **add comments.**

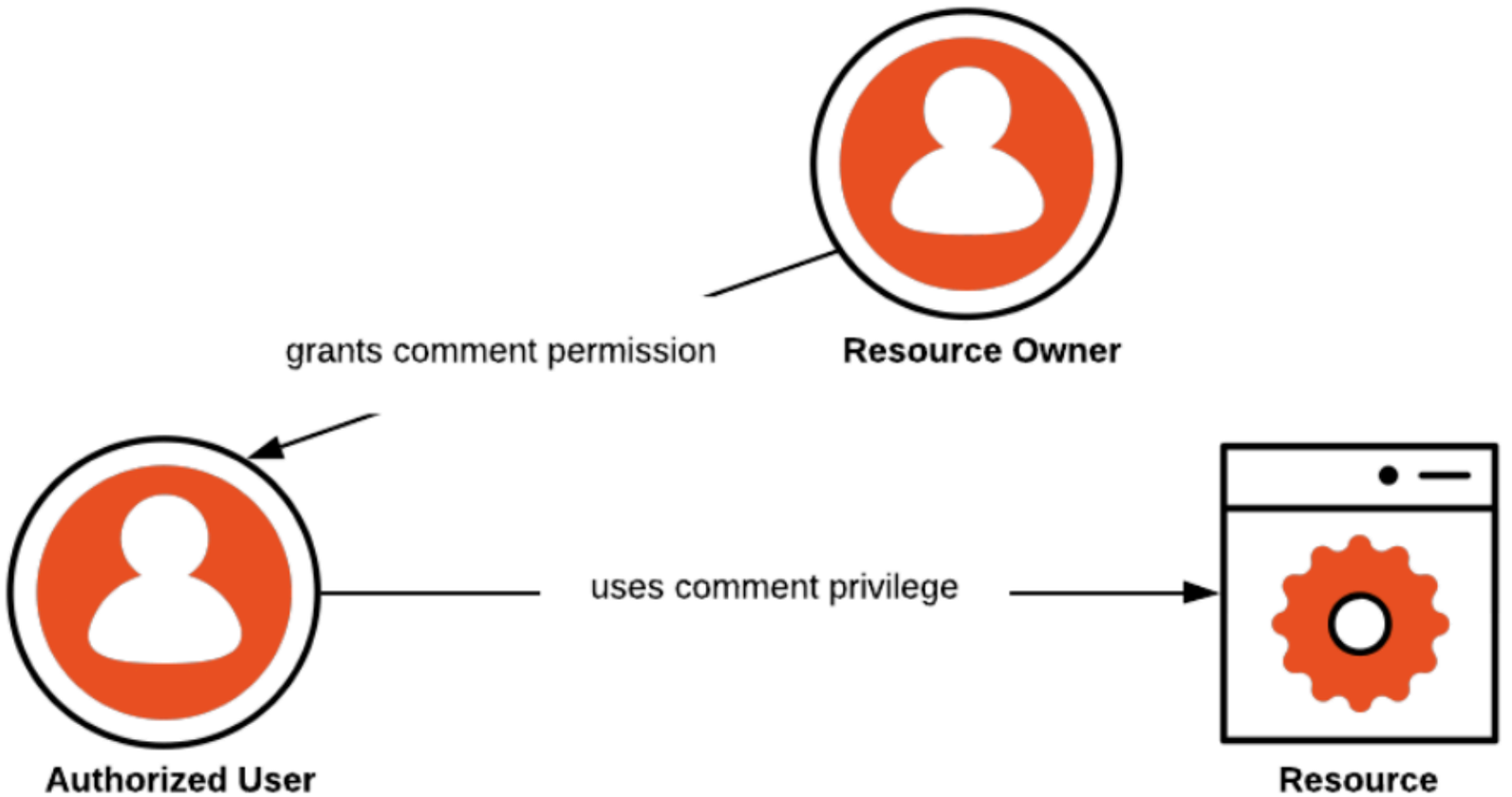
- In this scenario:

- **Resource**: it’s the document

- **Resource Owner:** this is the user that creates a document, the owner of the document

- **Authorized User**: the user who is given comment rights by the Resource Owner

Authorization



Access Control

- Access control in computer systems and networks rely on access policies.

- The access control process can be divided into the following phases:
    - **policy definition phase** where access is authorized, and
    - **policy enforcement phase** where access requests are approved or disapproved.

- Authorization is the **function of the policy definition phase** which precedes the policy enforcement phase where access requests are approved or disapproved based on the previously defined authorizations.

Access Control

- The most Access control mechanisms are
    - **Attribute-Based Access Control (ABAC)**
        - An access control paradigm whereby access rights are granted to users through the use of policies which evaluate attributes (user attributes, resource attributes and environment conditions)
    - **Discretionary Access Control (DAC)**
        - In DAC, the data owner determines who can access specific resources. For example, a system administrator may create a hierarchy of files to be accessed based on certain permissions.

Access Control

- The most Access control mechanisms are
  - **Graph-Based Access Control (GBAC)**
    - Compared to other approaches like RBAC or ABAC, the main difference is that in GBAC access rights are defined using an organizational query language instead of total enumeration.
  - **Identity-Based Access Control (IBAC)**
    - Using this network administrators can more effectively manage activity and access based on individual needs

Access Control

- The most Access control mechanisms are
    - **Role-Based Access Control (RBAC)**
        - RBAC allows access based on the job title. RBAC largely eliminates discretion when providing access to objects. For example, a human resources specialist should not have permissions to create network accounts; this should be a role reserved for network administrators.
    - **Mandatory Access Control (MAC)**
        - In MAC, users do not have much freedom to determine who has access to their files. For example, security clearance of users and classification of data (as confidential, secret or top secret) are used as security labels to define the level of trust.

Access Control

- The most Access control mechanisms are
  - **Rule-Based Access Control (RAC)**
    - RAC method, also referred to as Rule-Based Role-Based Access Control (RB-RBAC), is largely context based.
    - Example of this would be allowing students to use labs only during a certain time of day; it is the combination of students' RBAC-based information system access control with the time-based lab access rules.