Entity Authentication

Dr. Kunwar Pal
CSE
NITJ

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 the former, Alice sends a message to Bob. When Bob authenticates the message, Alice may or may not be present in the communication process.
- On the other hand, 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 she is authenticated can messages be communicated between Alice and Bob.
- 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

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.

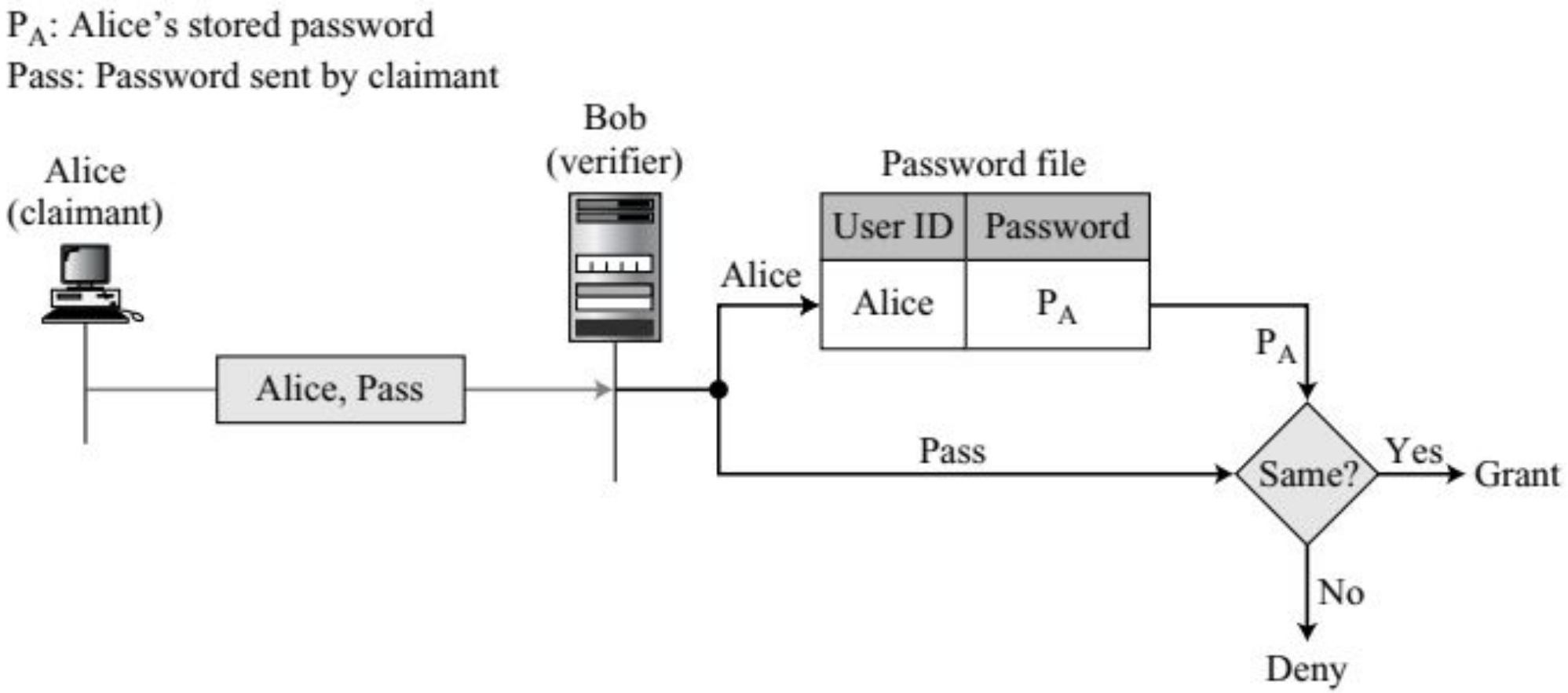
PASSWORDS

- 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.**

- First Approach: 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.

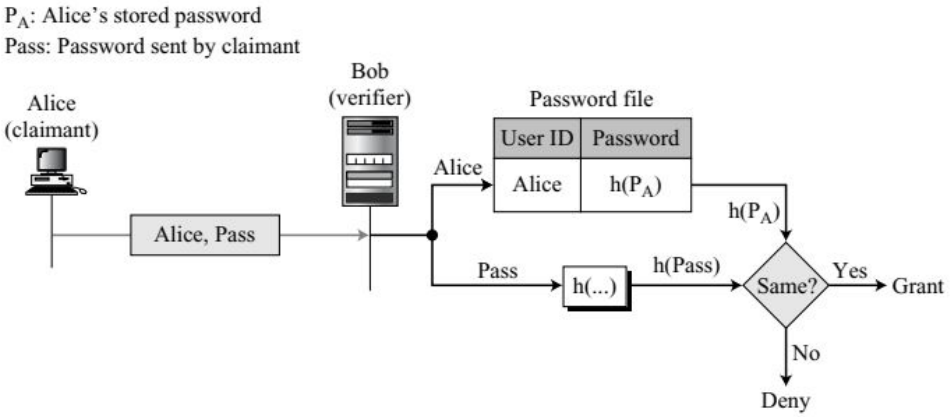

P_A: Alice's stored password
Pass: Password sent by claimant
Alice
(claimant)
Bob
(verifier)
Password file
User ID
Password
Alice
P_A
Alice
Alice, Pass
P_A
Pass
Same?
Yes
Grant
No
Deny

Attacks on the First Approach

- **Eavesdropping:** Eve can watch Alice when she types her password. Most systems, as a security measure, do not show the characters a user types. Eavesdropping can take a more sophisticated form. Eve can listen to the line and intercept the message, thereby capturing the password for her own use.
- **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 commits it to 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.
- **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 by the public.
- **Guessing:** Using a guessing attack, Eve can log into the system and try to guess Alice's password by trying different combinations of characters.

Second Approach

- 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.



Dictionary Attack

- The hash function prevents Eve from gaining access to the system even though she has the password file.
- However, there is still the possibility of dictionary attack.
- In this attack, Eve is interested in finding one password, regardless of the user ID.

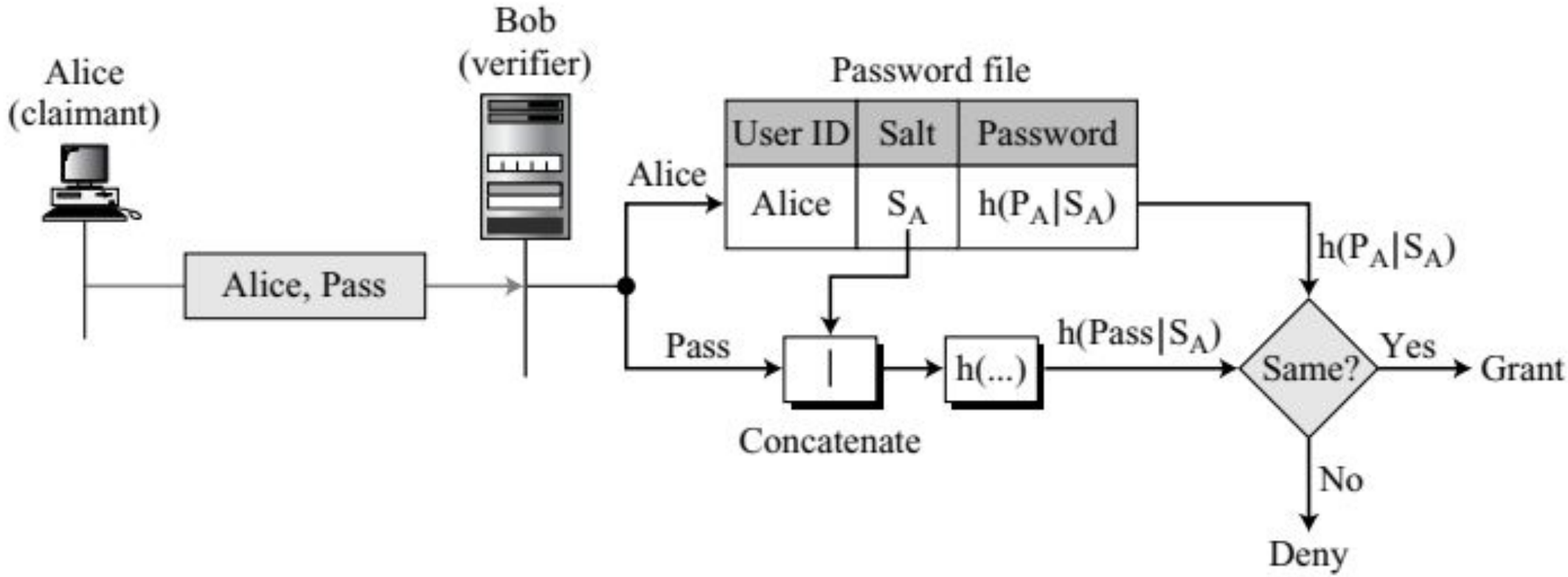
Third Approach

- 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

P_A: Alice's password

S_A: Alice's salt

Pass: Password sent by claimant



Fourth Approach

- In the fourth 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

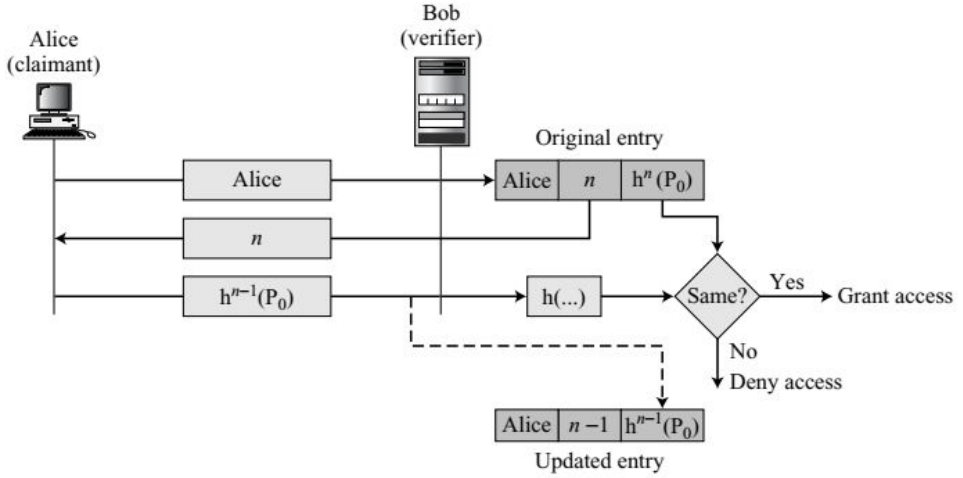
- In the first approach, the user and the system agree upon a list of passwords.
- Each password on the list can be used only once.
- There are some drawbacks to this approach.
- First, the system and the user must keep a long list of passwords.
- Second, if the user does not use the passwords in sequence, 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.
- Of course, if Eve can guess the first password (P1), she can find all of the subsequent ones

Third Approach

- In the third approach, the user and the system create a sequentially updated password using a hash function.
- The user and the system agree upon an original password, P0, and a counter, n.
- The system calculates hn(P0), where hn means applying a hash function n times.
-

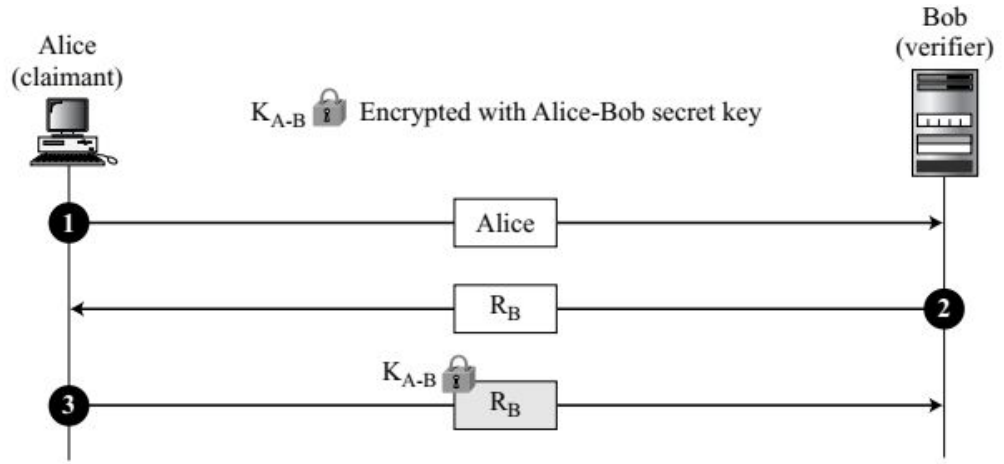


CHALLENGE-RESPONSE

- In password authentication, the claimant proves her identity by demonstrating that she knows a secret, the password.
- However, because the claimant reveals this secret, it is susceptible to interception by the adversary.
- In challenge-response authentication, the claimant proves that she knows a secret without sending it.
- In other words, the claimant does not send the secret to the verifier; the verifier either has it or finds it.
- The challenge is a time-varying value such as a random number or a timestamp that is sent by the verifier.
- The claimant applies a function to the challenge and sends the result, called a response, to the verifier.
- The response shows that the claimant knows the secret.

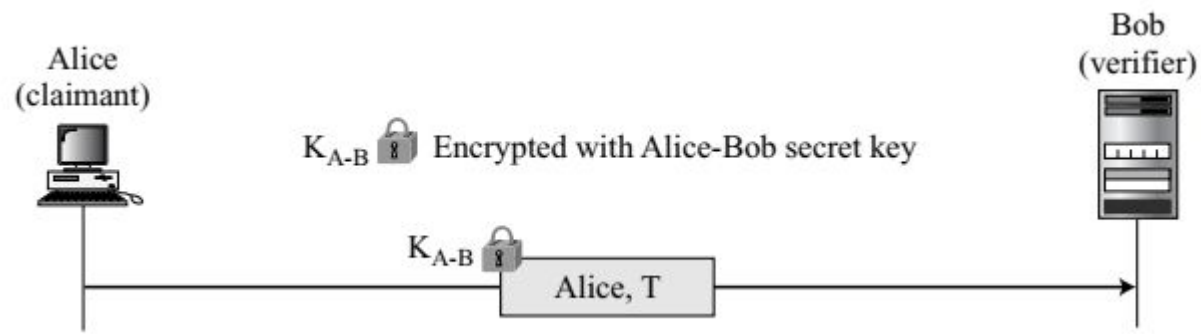
Using a Symmetric-Key Cipher

- In the first approach, the verifier sends a nonce, a random number used only once, to challenge the claimant.
- A nonce must be time-varying; every time it is created, it is different.
- The claimant responds to the challenge using the secret key shared between the claimant and the verifier.



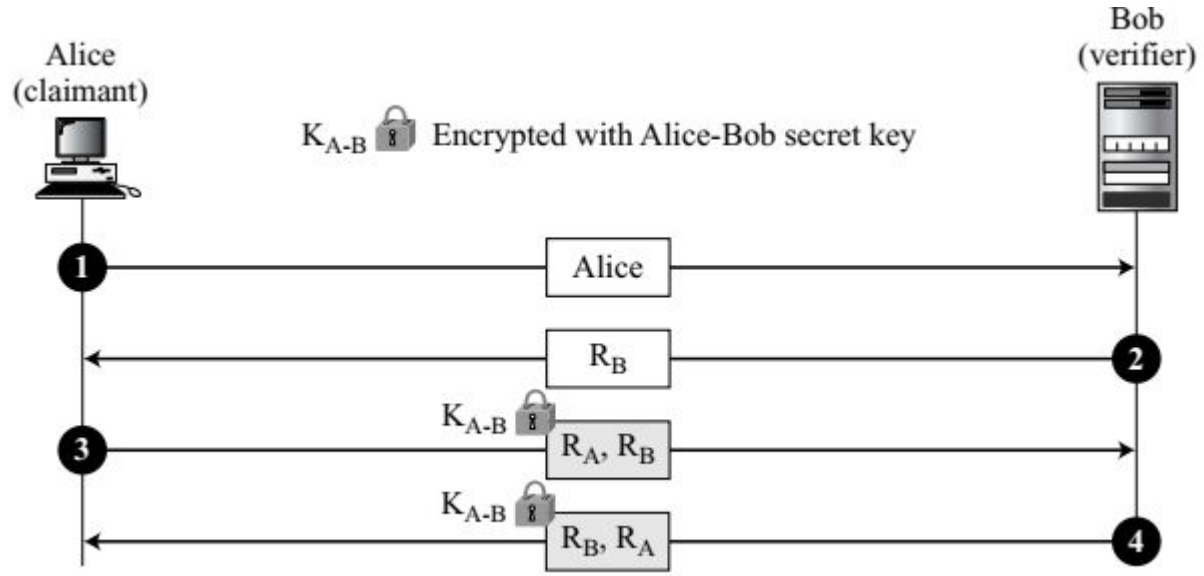
Second Approach

- In the second approach, the time-varying value is a timestamp, which obviously changes with time.
- In this approach the challenge message is the current time sent from the verifier to the claimant.
- However, this supposes that the client and the server clocks are synchronized; the claimant knows the current time.
- This means that there is no need for the challenge message.
- The first and third messages can be combined.
- The result is that authentication can be done using one message, the response to an implicit challenge, the current time.


Alice
(claimant)
Bob
(verifier)
K_{A-B}
Encrypted with Alice-Bob secret key
K_{A-B}
Alice, T

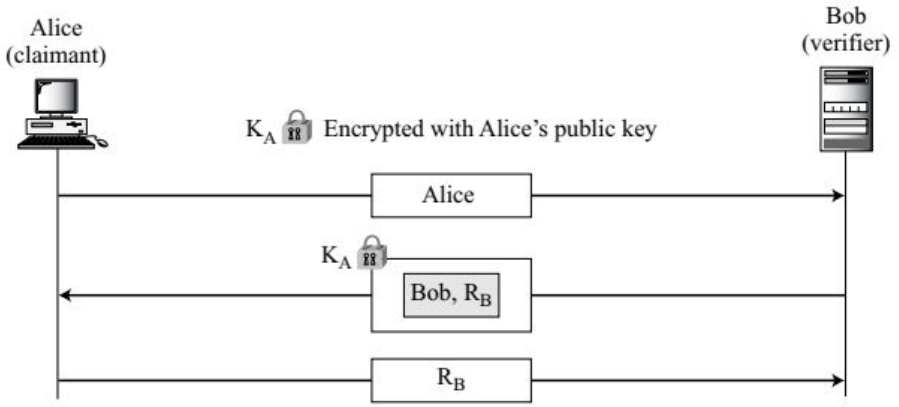
Third Approach

- The first and second approaches are for unidirectional authentication.
- Alice is authenticated to Bob, but not the other way around.
- If Alice also needs to be sure about Bob's identity, we need bidirectional authentication.



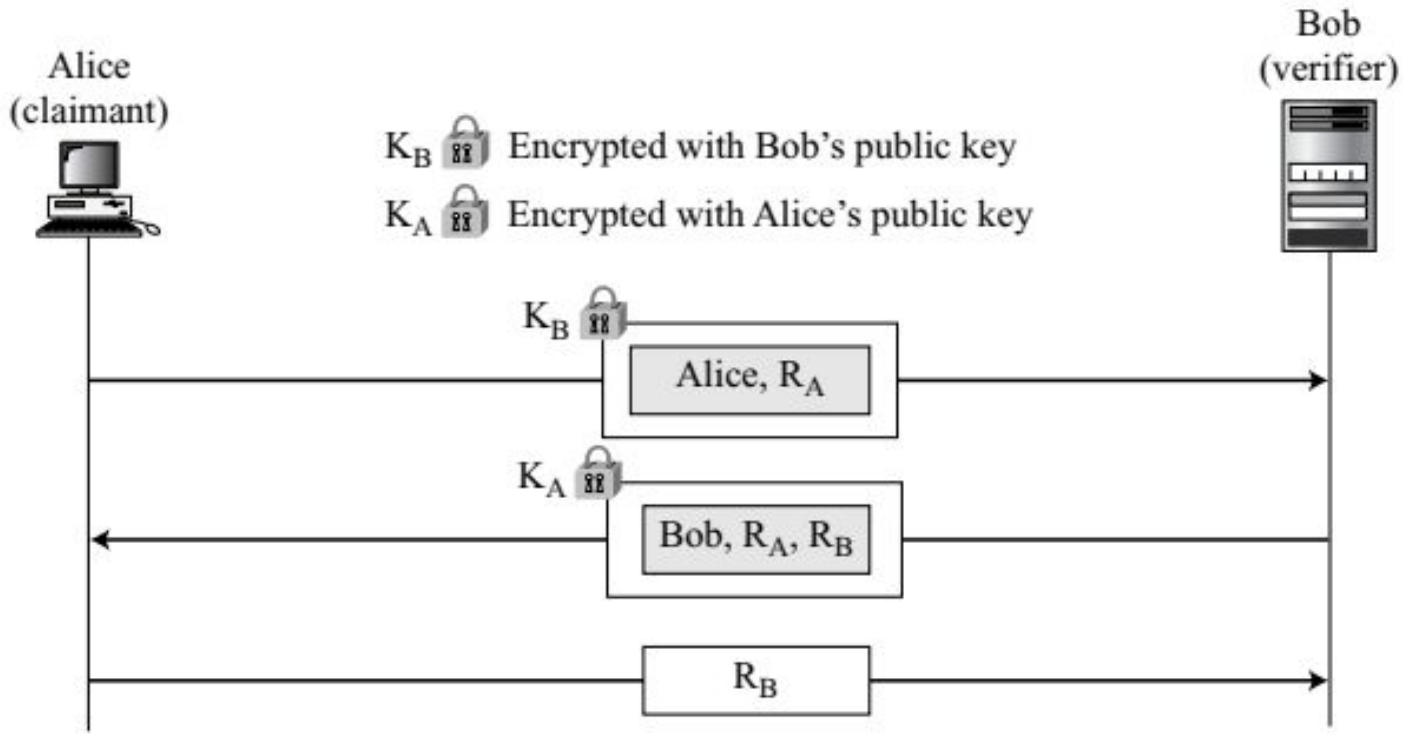
Using an Asymmetric-Key Cipher

- First Approach:
  - In the first approach, Bob encrypts the challenge using Alice's public key.
  - Alice decrypts the message with her private key and sends the nonce to Bob.

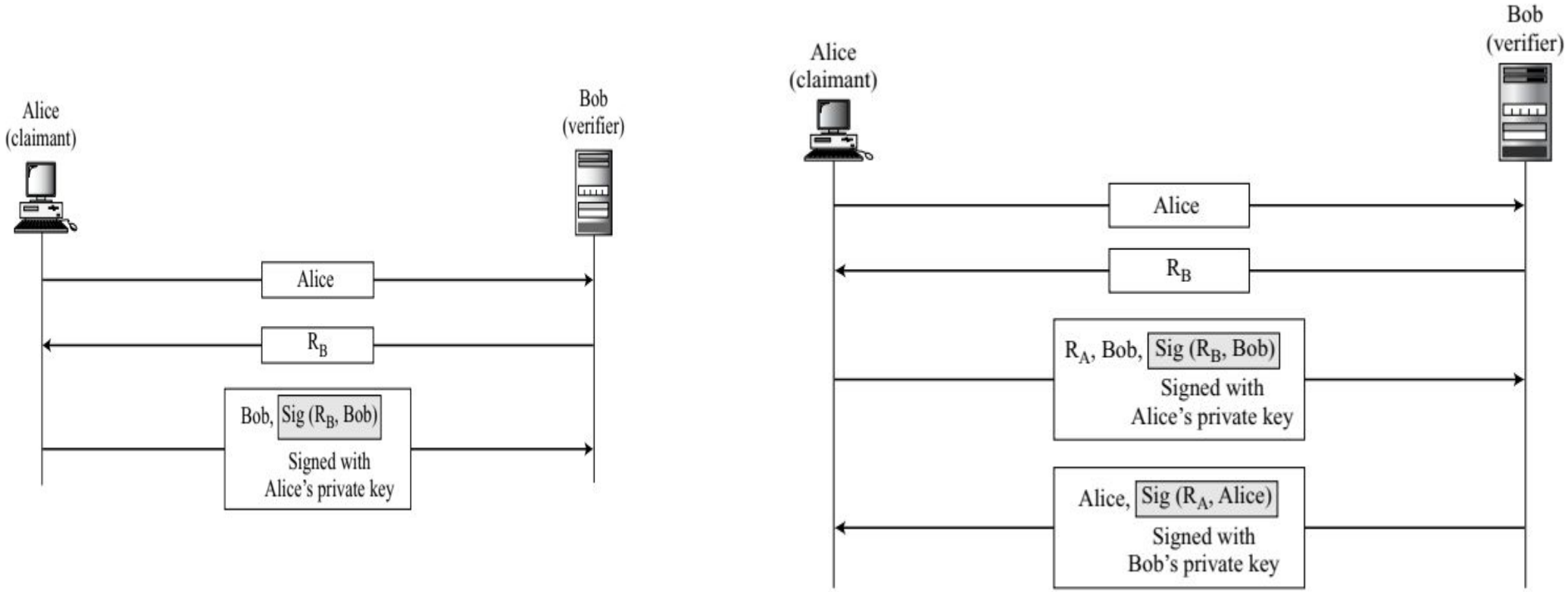


Second Approach

- In the second approach, two public keys are used, one in each direction.
- Alice sends her identity and nonce encrypted with Bob's public key.
- Bob responds with his nonce encrypted with Alice's public key.
- Finally, Alice, responds with Bob's decrypted nonce.



Using Digital Signature

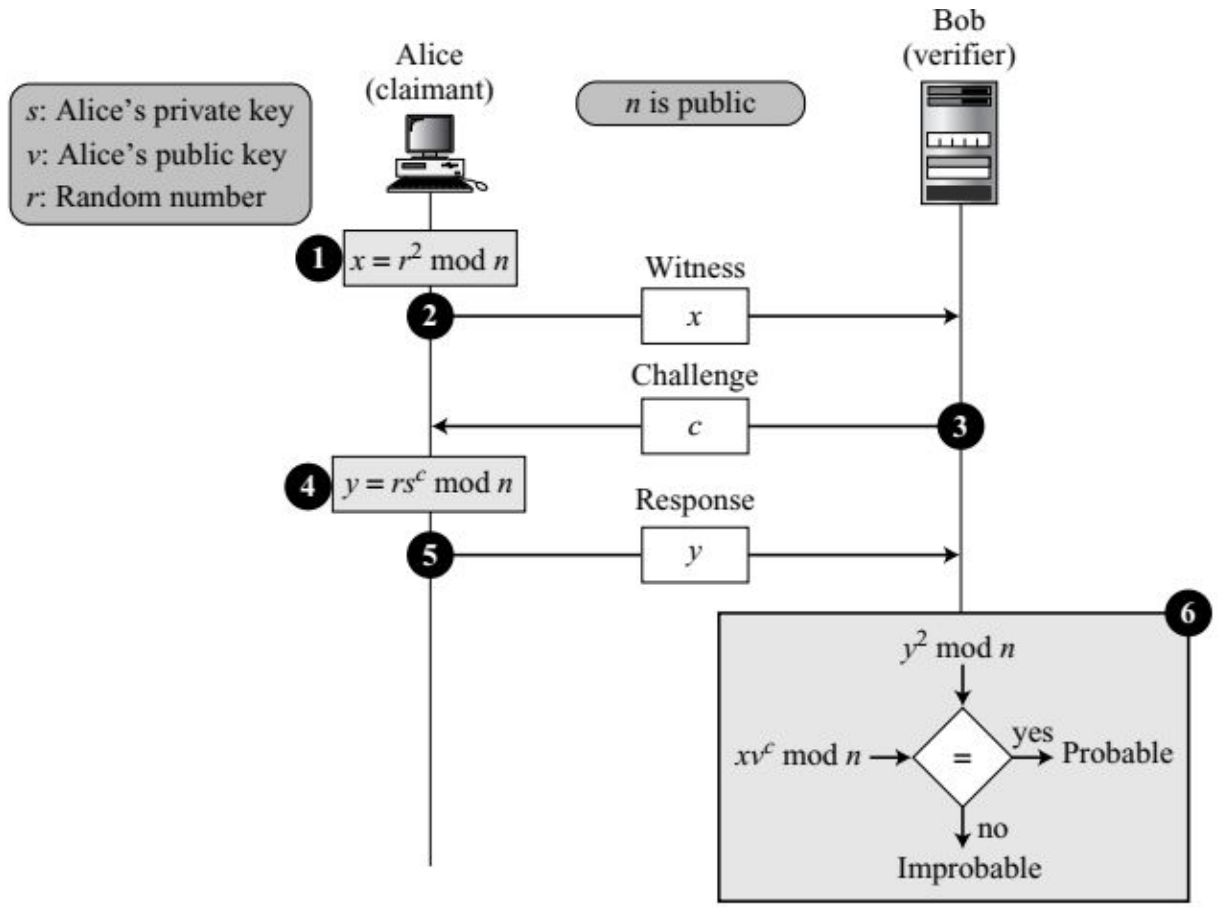
- 

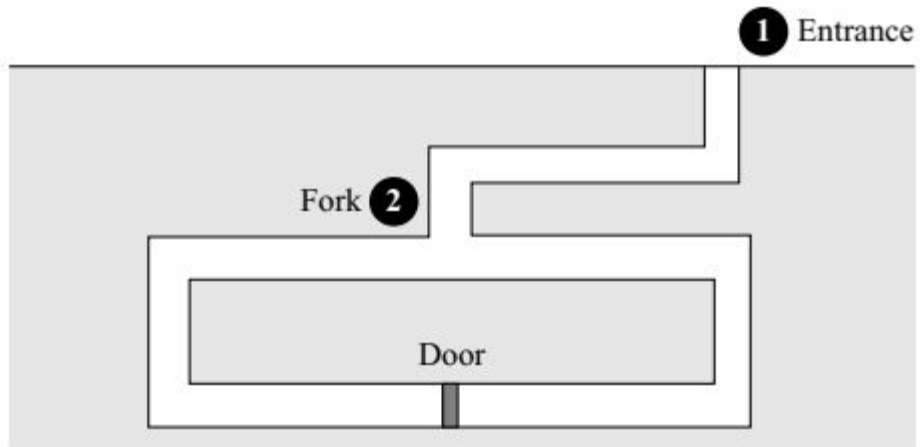
ZERO-KNOWLEDGE

- 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.

Fiat-Shamir Protocol

- In the Fiat-Shamir protocol, a trusted third party chooses two large prime numbers p and q to calculate the value of n = p × q.
- The value of n is announced to the public; the values of p and q are kept secret.
- Alice, the claimant, chooses a secret number s between 1 and n - 1 (exclusive).
- She calculates v = s2 mod n.
- She keeps s as her private key and registers v as her public key with the third party.


Alice
(claimant)
Bob
(verifier)
s: Alice's private key
v: Alice's public key
r: Random number
n is public
1
x = r^2 mod n
Witness
x
2
Challenge
c
3
4
y = rs^c mod n
Response
y
5
6
y^2 mod n
xv^c mod n
=
yes
Probable
no
Improbable


1 Entrance
Fork 2
Door

Feige-Fiat-Shamir Protocol

- The **Feige-Fiat-Shamir protocol** is similar to the first approach except that it uses a vector of private keys $[s_1, s_2, \ldots, s_k]$, a vector of public keys $[v_1, v_2, \ldots, v_k]$, and a vector of challenges $(c_1, c_2, \ldots, c_k)$. The private keys are chosen randomly, but they must be relatively prime to n. The public keys are chosen such that $v_i = (s_i^2)^{-1} \bmod n$. The three steps in the process are shown in Figure 14.15.

  We can prove that $y^2 v_1^{c_1} v_2^{c_2} \ldots v_k^{c_k}$ is the same as x:

$$
\begin{aligned}
y^2 v_1^{c_1} v_2^{c_2} \ldots v_k^{c_k} &= r^2 (s_1^{c_1})^2 (s_2^{c_2})^2 \ldots (s_k^{c_k})^2 \, v_1^{c_1} v_2^{c_2} \ldots v_k^{c_k} \\
&= x \, (s_1^2)^{c_1} (v_1^{c_1}) \, (s_2^2)^{c_2} (v_2^{c_2}) \ldots (s_2^2)^{c_2} (v_2^{c_k}) \\
&= x \, (s_1^2 v_1)^{c_1} \, (s_2^2 v_2)^{c_2} \ldots (s_k^2 v_k)^{c_k} = x \, (1)^{c_1} \, (1)^{c_2} \ldots (1)^{c_k} = x
\end{aligned}
$$

  The three exchanges constitute a round; verification is repeated several times with the value of c's equal to 0 or 1 (chosen randomly). The claimant must pass the test in each round to be verified. If she fails a single round, the process is aborted and she is not authenticated.

Guillou-Quisquater Protocol

-