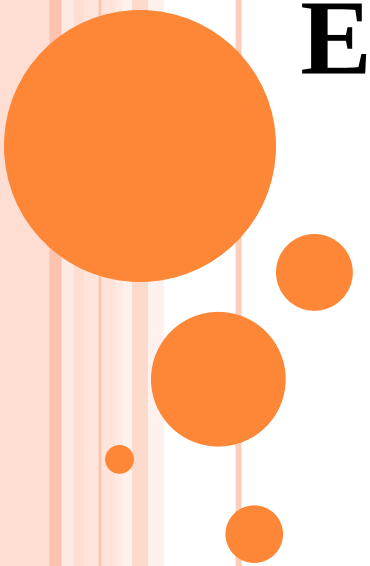
Entity Authentication



- 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.

- **Difference between message authentication and entity authentication**

  - Entity authentication happen in real time .

  - Message authentication authenticates one message but entity authentication authenticates the claimant for the entire duration of the session.

Verification Categories

- Something known
  - Eg : password, PIN, secret key
- Something possessed
  - Eg : passport,credit card
- Something inherent
  - Eg: Fingerprint, voice ,retinal pattern

- The simplest and oldest method is password based authentication.
- Whenever a user needs to access the system he uses a password which can be verified by the verifier.
- Passwords are of two types ,fixed and one time password.

P_A: Alice's stored password

Pass: Password sent by claimant

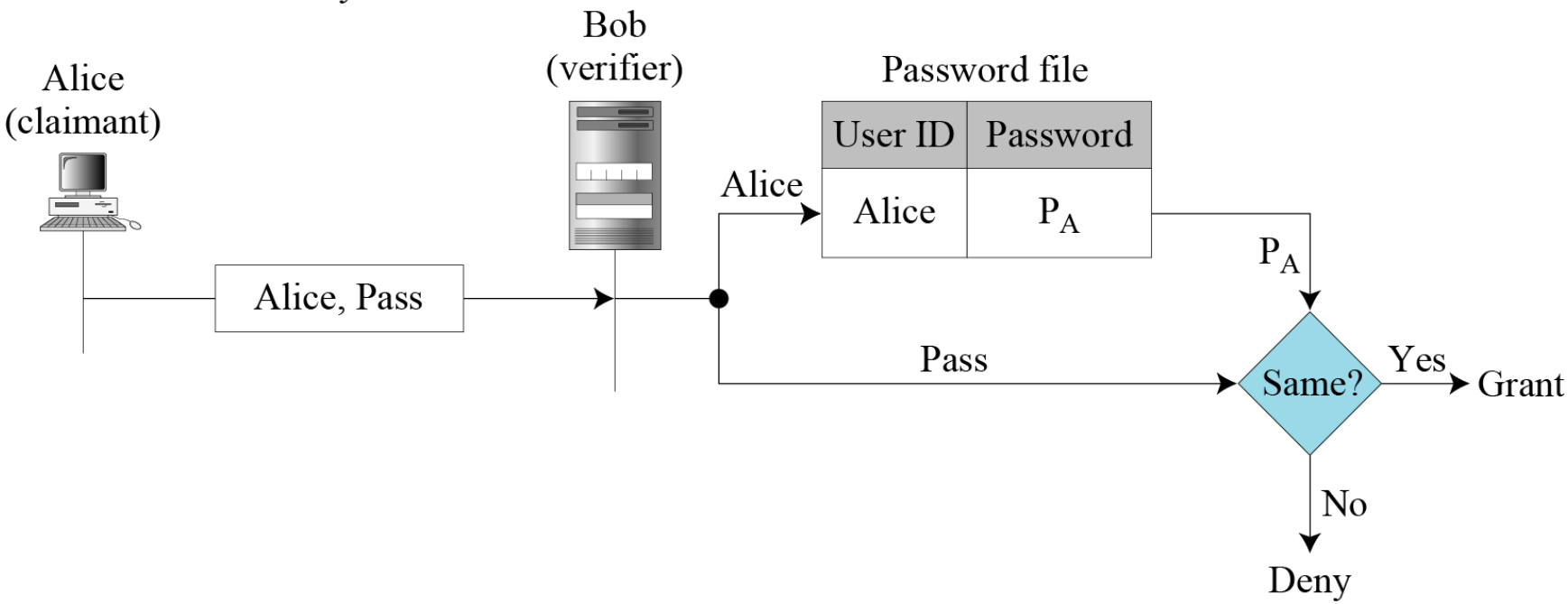


Fig : First approach using fixed password

- **Attacks on first approach**
  - Eavesdropping
  - Stealing a password
  - Accessing a password file
  - Guessing

P_A: Alice's stored password

Pass: Password sent by claimant

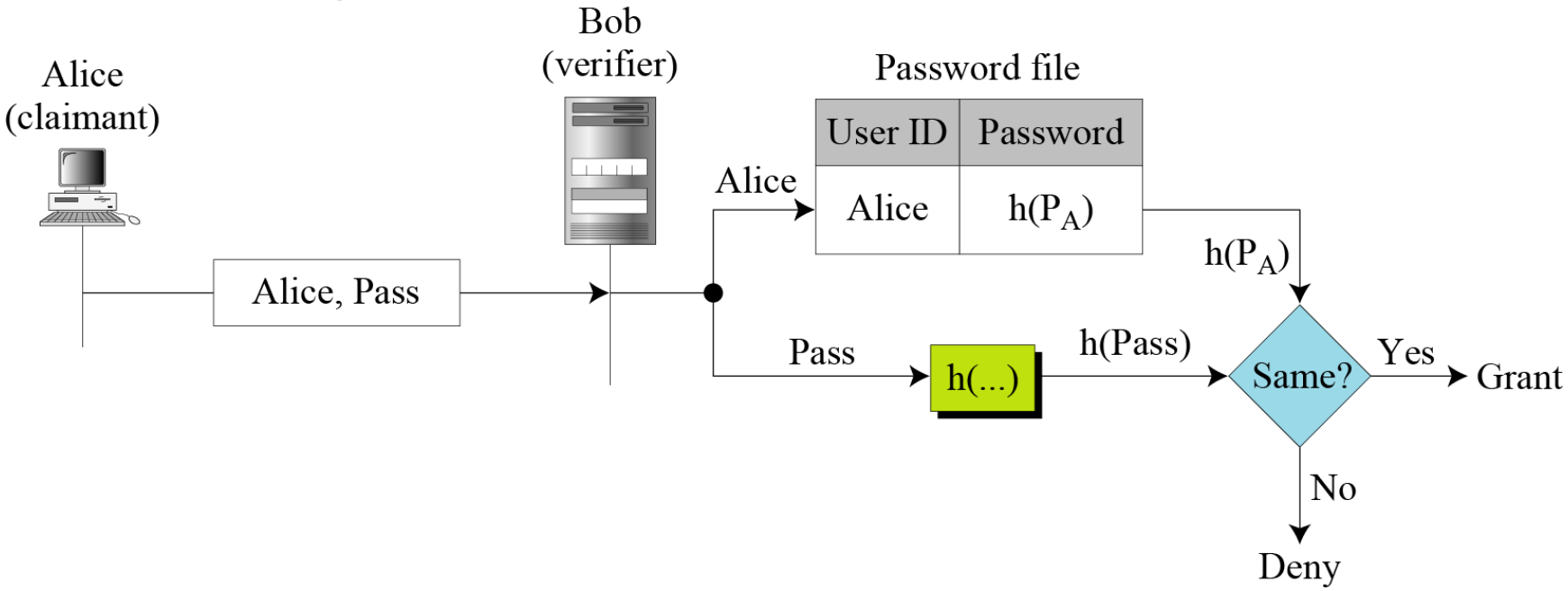


Fig : Hashing the password

- Second approach is vulnerable to dictionary attack.
- If he knows the password is six digits, eve can create a list of all six digit numbers.
- Find the hash value of all the numbers.
- Now eve can get the password file and try to obtain a match.

P_A: Alice's password

S_A: Alice's salt

Pass: Password sent by claimant

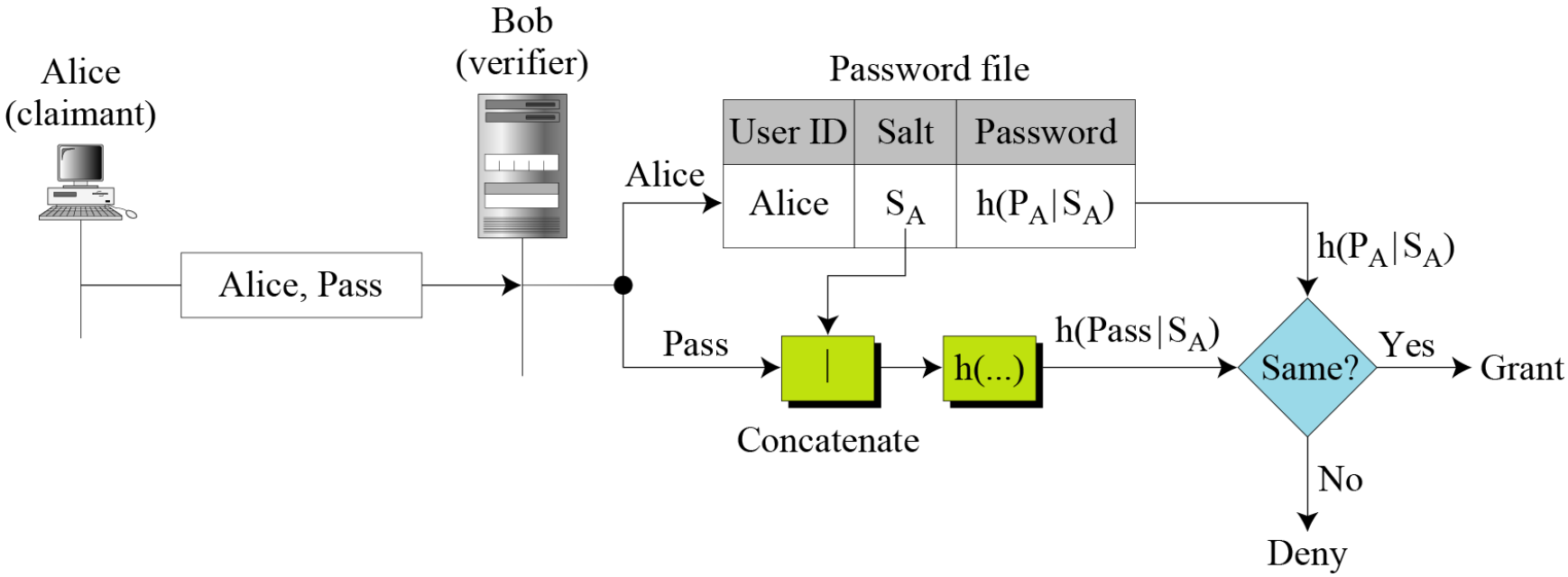


Fig : Salting the password

- One time password
- Password is used only once.
- It makes eavesdropping useless.

Challenge response authentication

- In password authentication ,claimant proves her identity by revealing the secret.
- In challenge response authentication claimant proves that she knows a secret without sending it.
- The verifier first send a challenge which is a time varying value.
- The response by the claimant is the result of a function applied to the challenge.
- The response shows that the claimant knows the secret.

USING A SYMMETRIC KEY CIPHER

- The secret here is the shared secret key.
- The function is the encryption algorithm applied on the challenge.
- **First approach**
- The verifier sends a nonce to challenge the claimant.

First approach

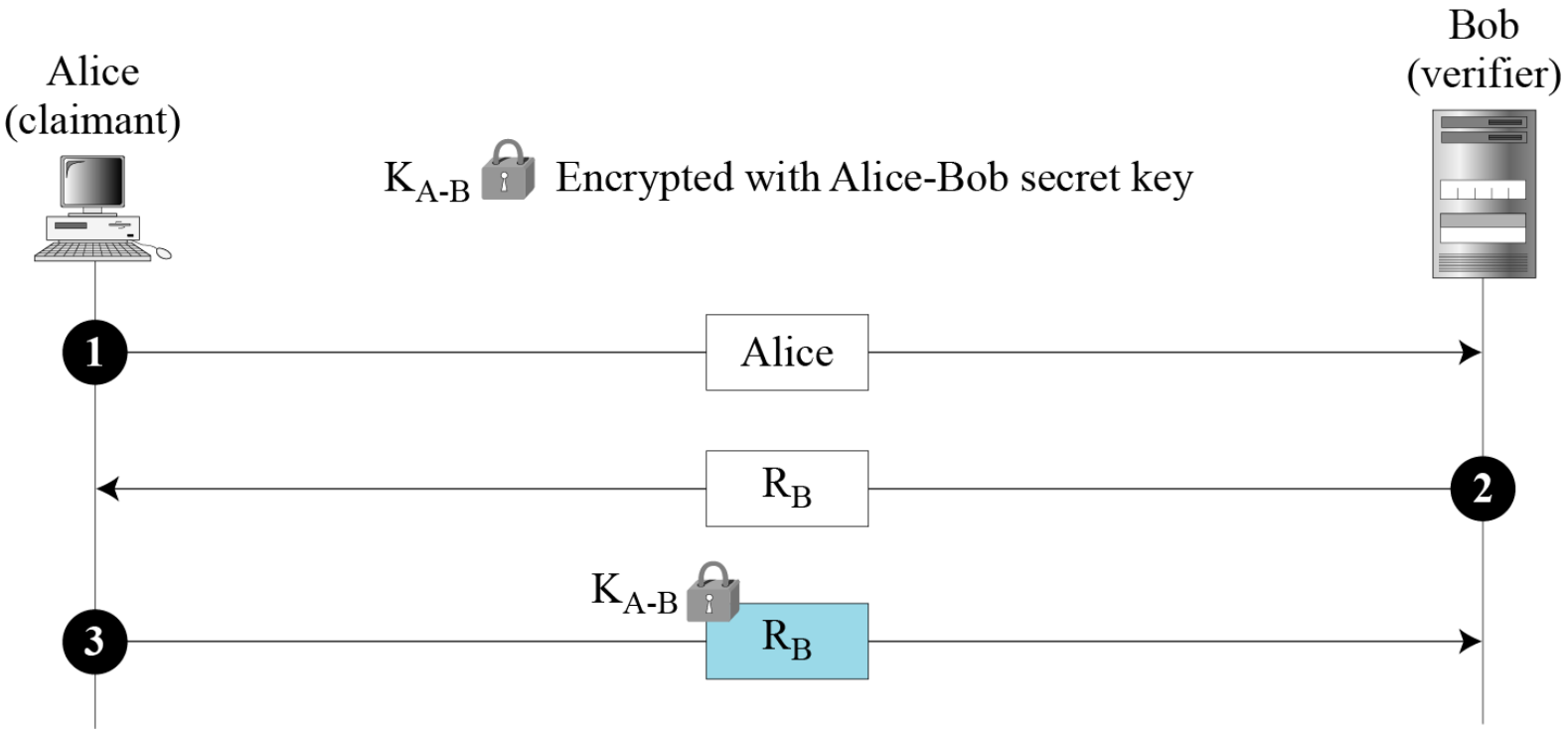


Fig : Nonce challenge

- The claimant and verifier should keep the value of symmetric key secret.
- The verifier should keep the value of nonce until the response is received.
- Use of nonce prevents the replay of third message.

Second approach

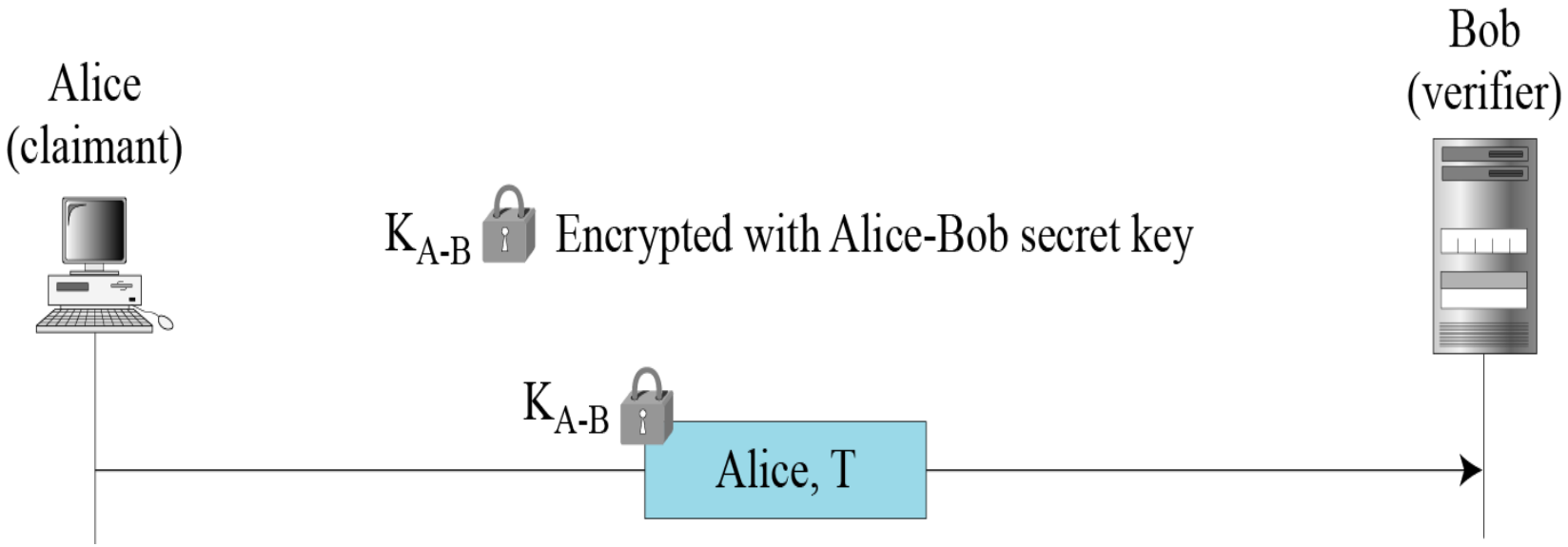


Fig : Timestamp challenge

- Challenge message is the current time sent by the verifier.
- The assumption is that client and server clocks are synchronized.
- Therefore the claimant knows the time.
- Therefore no need for challenge message.
- Authentication can be done using one message.

Third approach

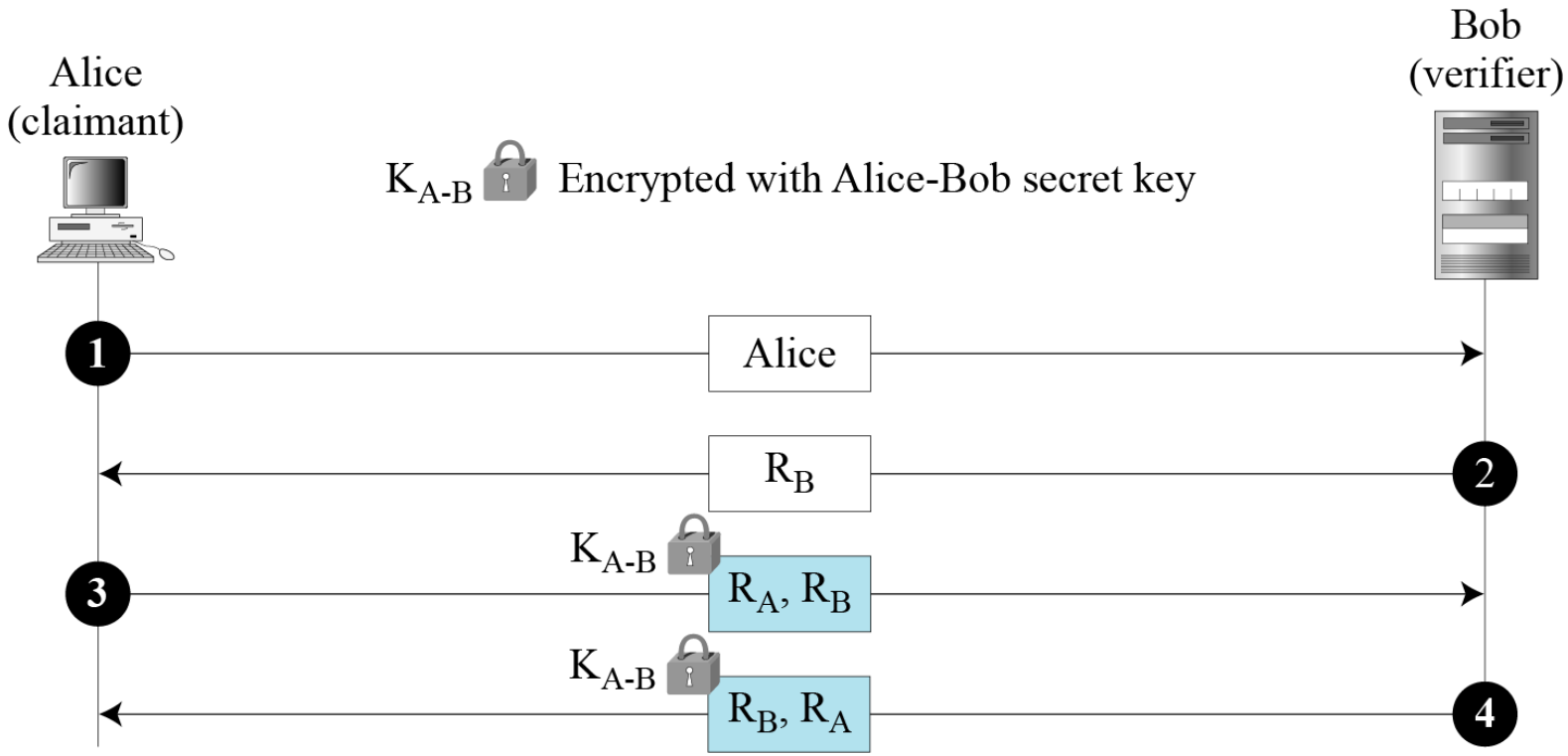


Fig : Bidirectional Authentication

USING A KEYED HASH FUNCTION

- The timestamp is sent both as plaintext and it's corresponding MAC.
- The receiver applies the keyed hash function and compares the calculated value with the received value.

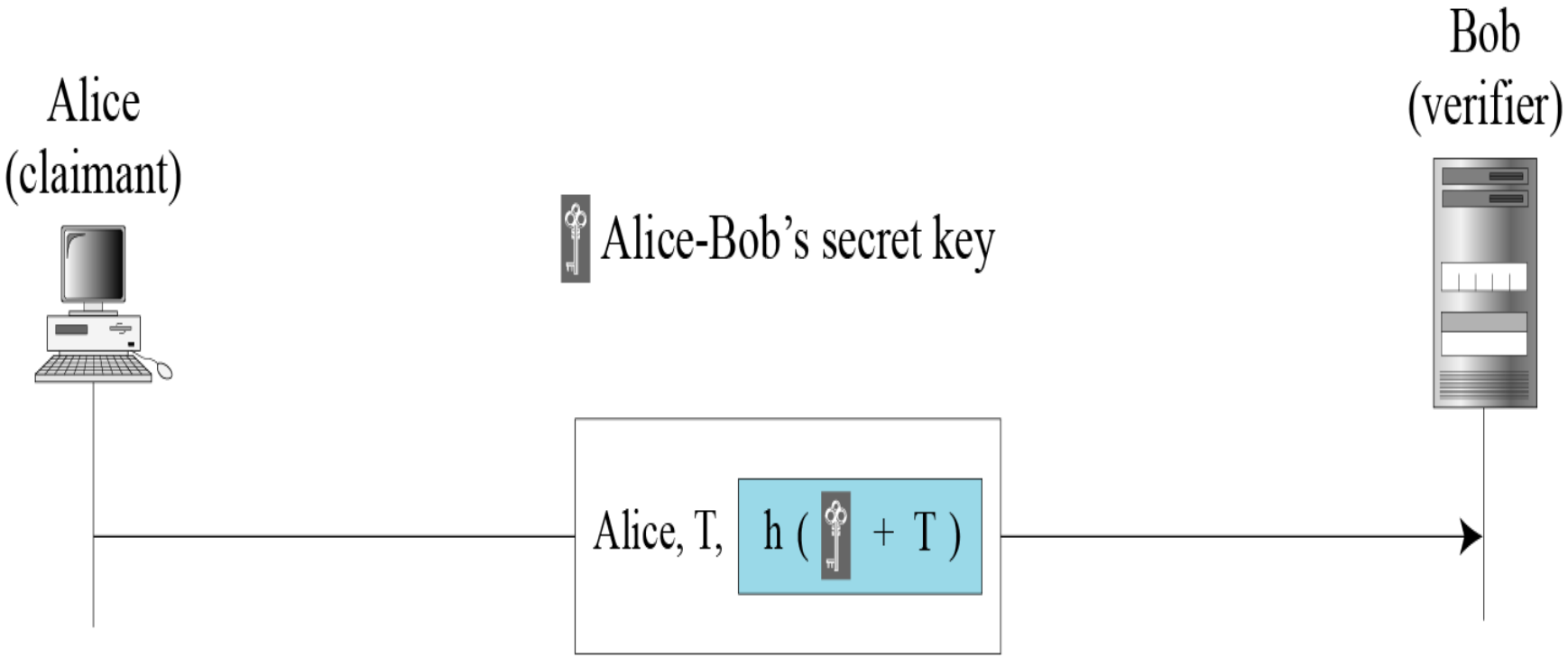


Fig: Keyed Hash Function

Using an asymmetric key cipher

- The secret element is the private key of claimant.
- The verifier encrypts the message using the public key of claimant.
- The claimant decrypts the message using the private key.
- The response to the challenge is the decrypted challenge.

First Approach

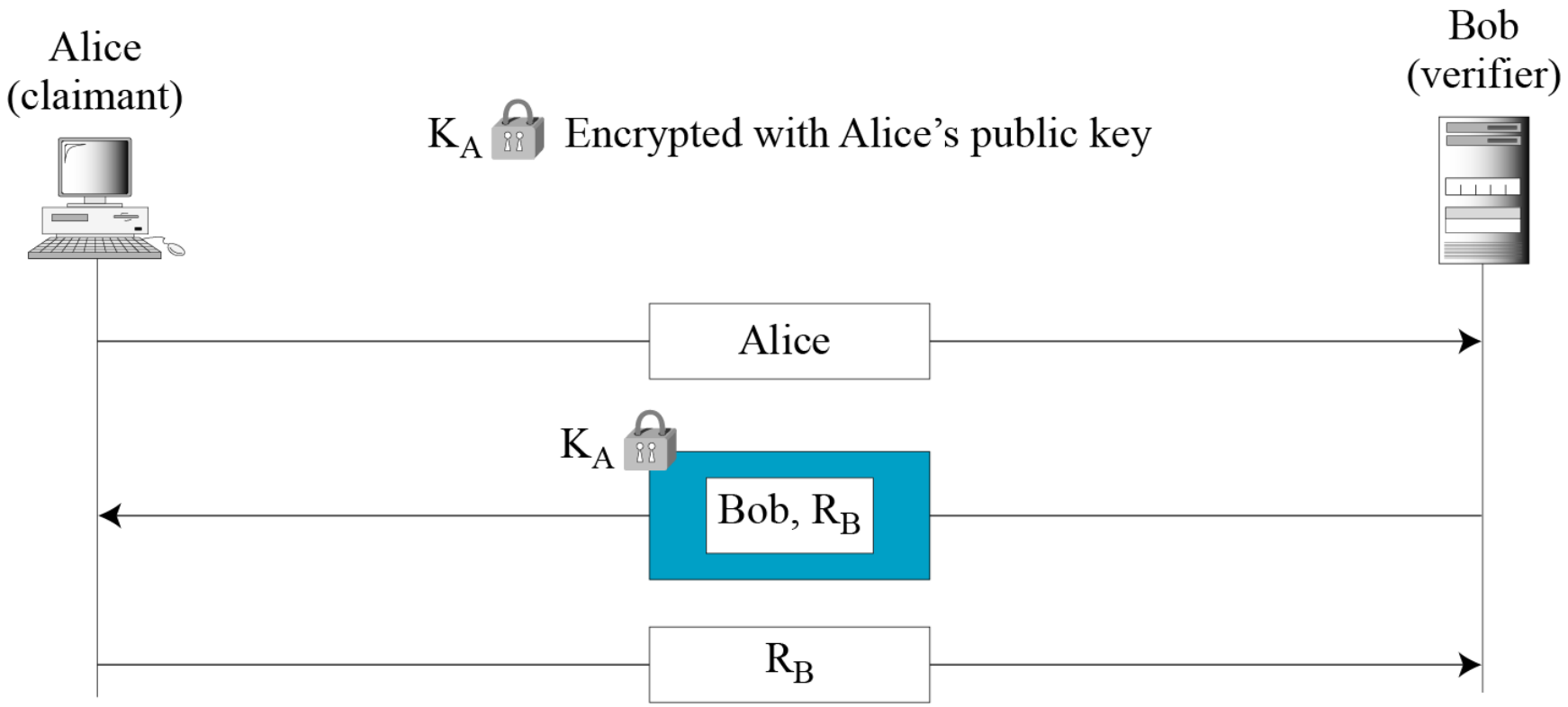


Fig :Unidirectional asymmetric key authentication

Second Approach

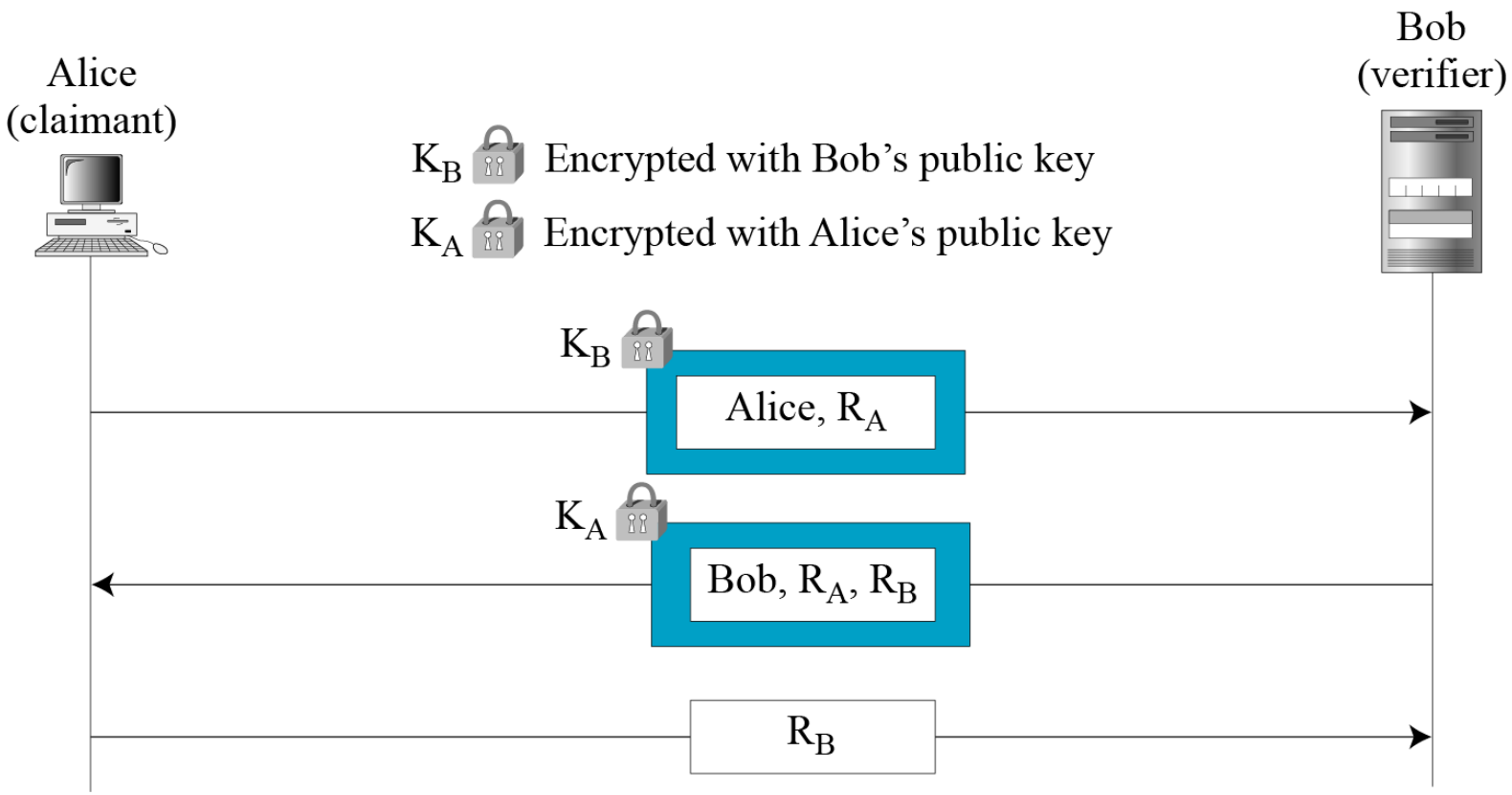


Fig: Bidirectional asymmetric-key authentication

- **Using digital signature**
- For entity authentication the claimant uses her private key for signing.

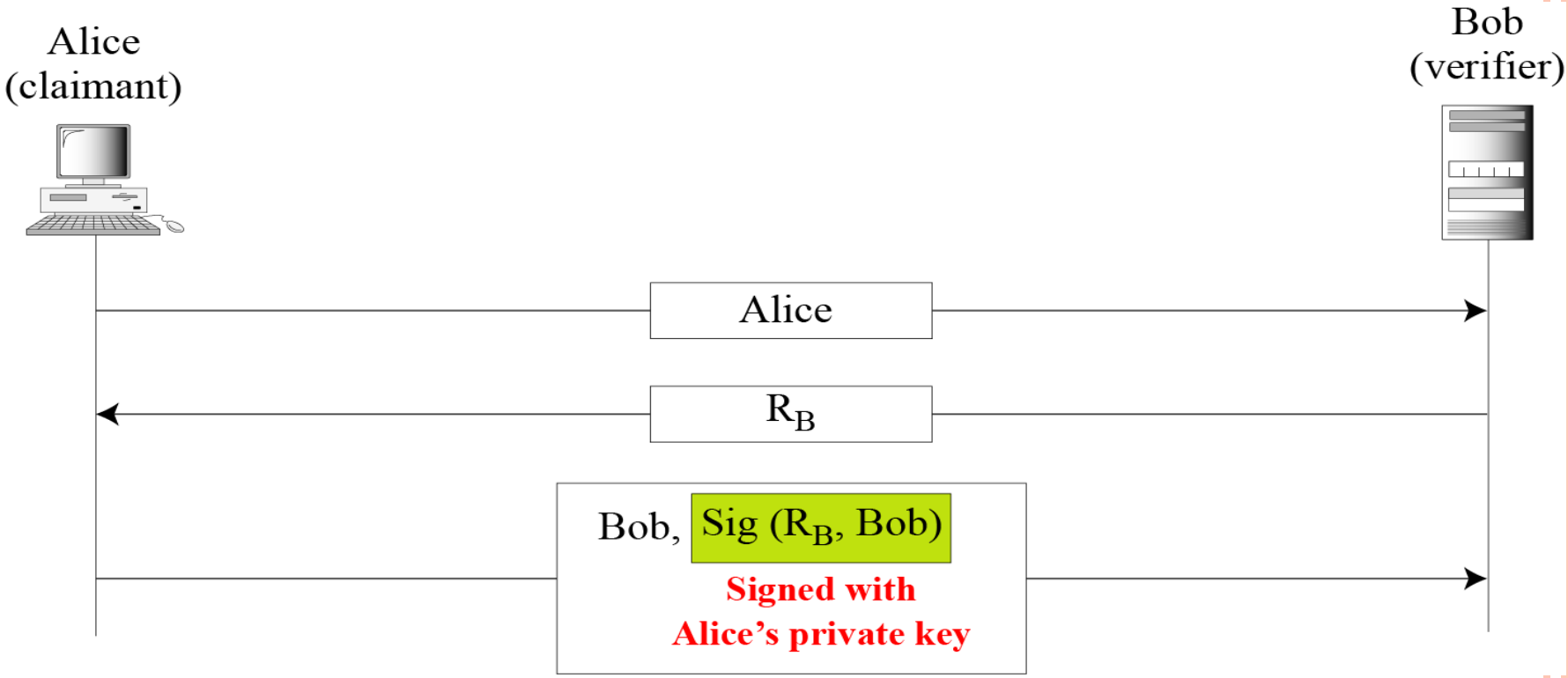


Fig : Unidirectional authentication

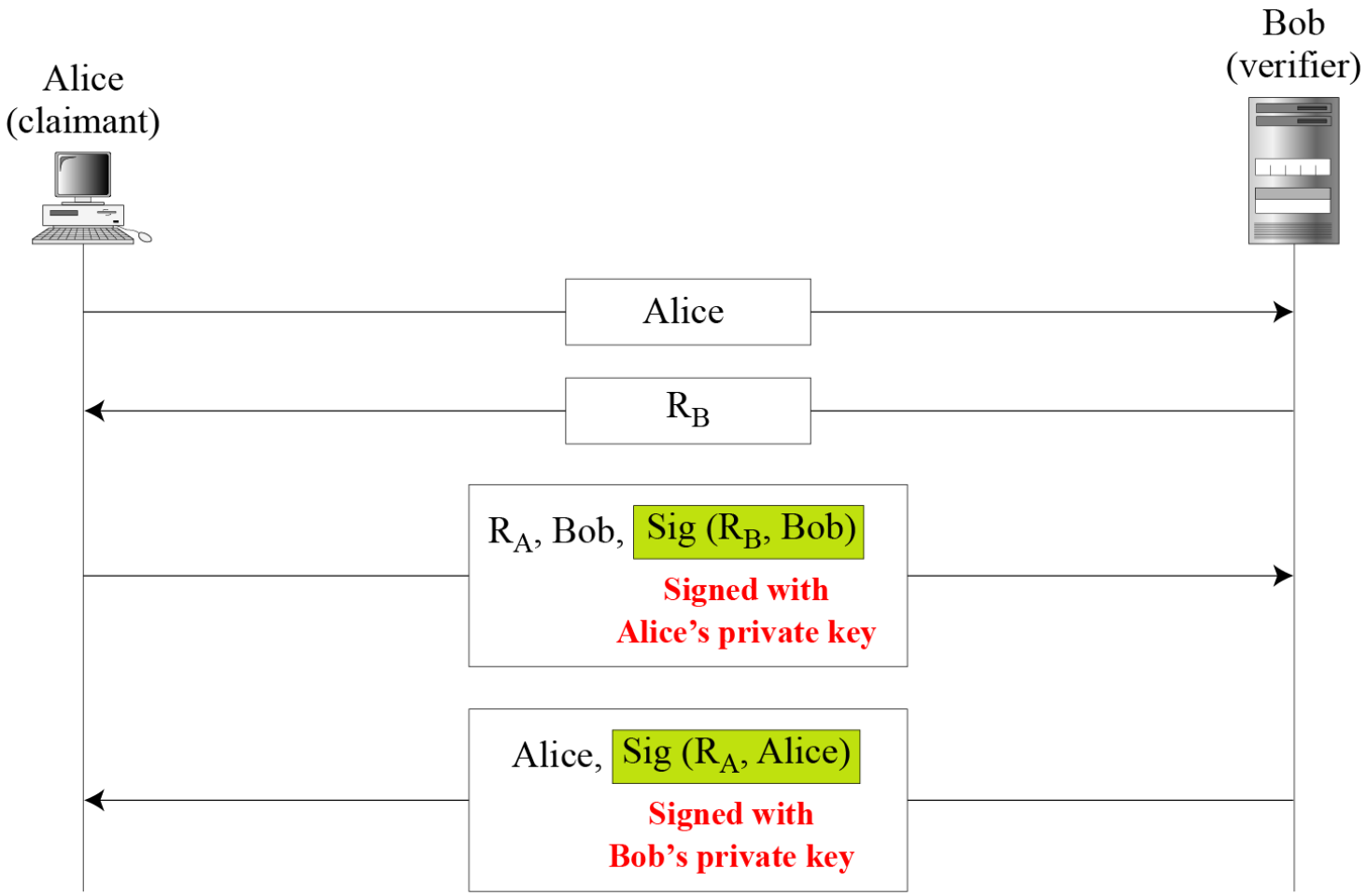


Fig : Bidirectional authentication