CSS2C08

COMPUTER NETWORKS

MODULE 5
Security in Networks

1. **Principles of cryptography**
2. Integrity
3. Authentication
4. Key distribution and certification
5. Firewalls
6. Attacks and counter measures

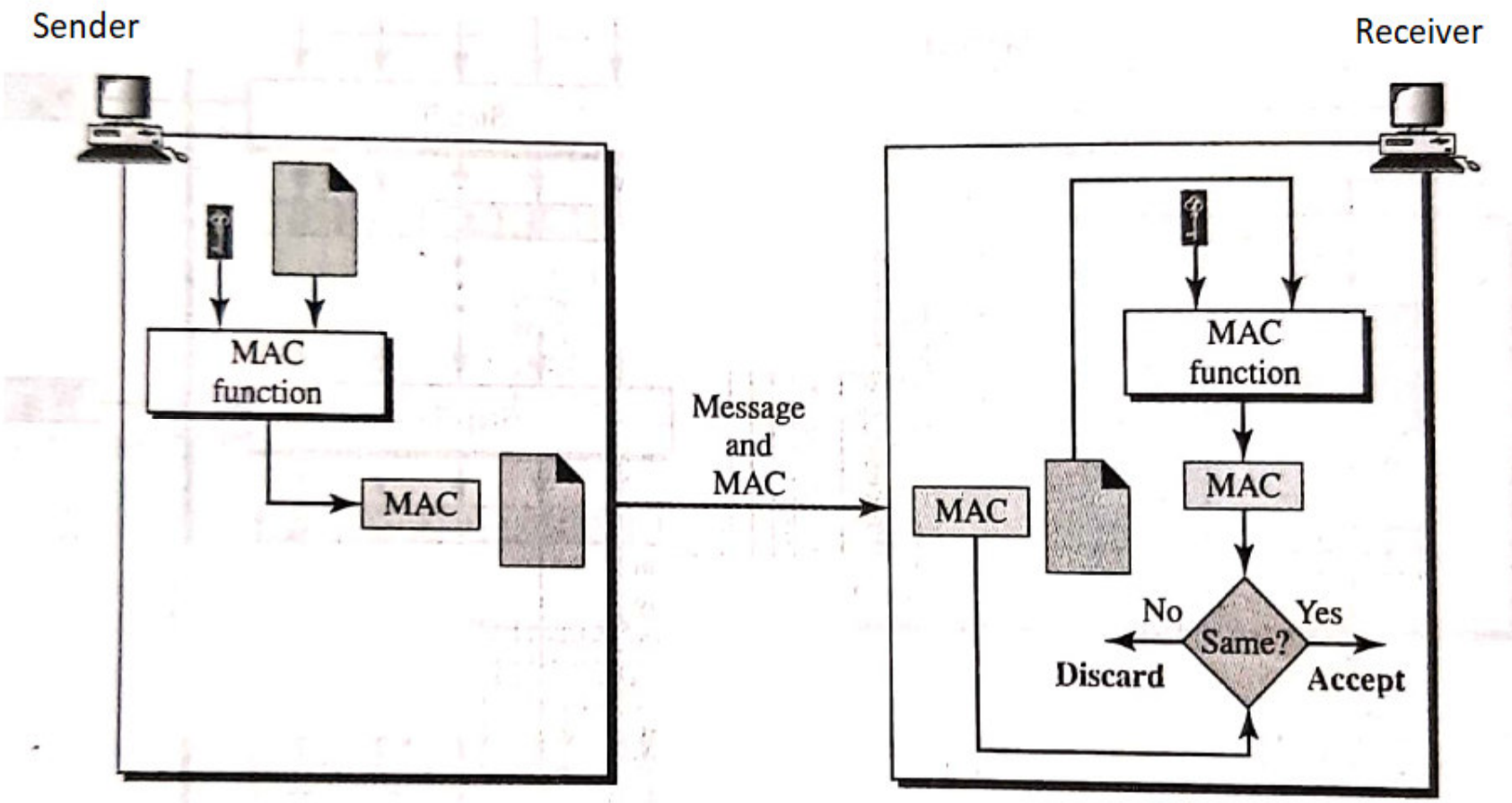
Authentication

- **Message Authentication:**
  - In message authentication the receiver needs to be sure of the sender's identity and that an imposter has not sent the message.
- **Entity Authentication:**
  - In entity authentication (or user identification) the entity or user is verified prior to access to the system resources.

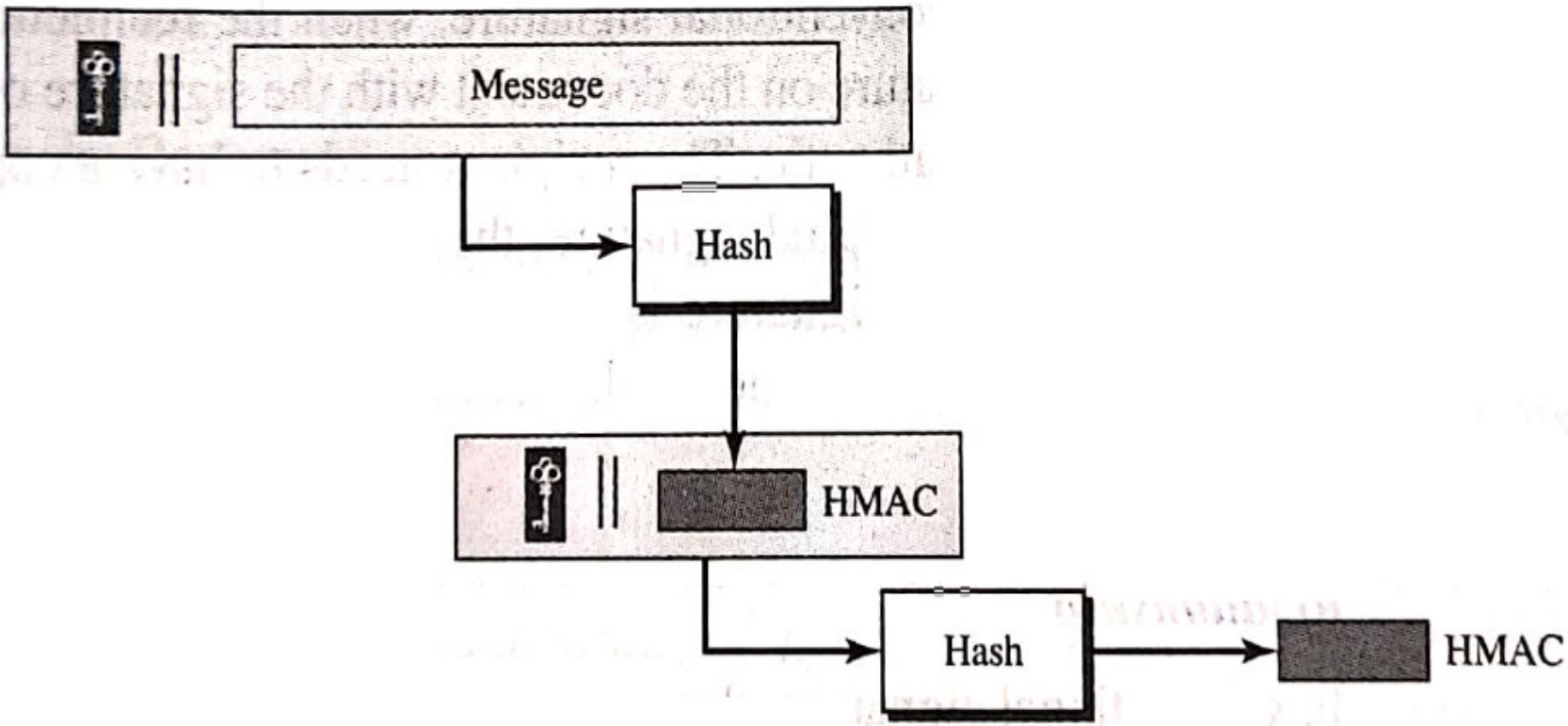
Message Authentication

- A hash function guarantees the integrity of a message. It guarantees that the message has not been changed. A hash function, does not authenticate the sender of the message.
- The digest created by a hash function is normally called a modification detection code (MDC). The code can detect any modification in the message.
- To provide message authentication, we need to change a modification detection code to a message authentication code (MAC).

- **MAC:**
  - An MDC uses a keyless hash function.
  - A MAC uses a keyed hash function.
  - A keyed hash function includes the symmetric key between the sender and receiver when creating the digest.


Sender
Receiver
MAC function
MAC
Message and MAC
MAC function
MAC
MAC
No
Yes
Same?
Discard
Accept

- **HMAC:**
  - Some MACs have been designed that are based on keyless hash functions such as SHA-l. This idea is a hashed MAC, called HMAC, that can use any standard keyless hash function such as SHA-l.
  - HMAC creates a nested MAC by applying a keyless hash function to the concatenation of the message and a symmetric key.


Message
Hash
HMAC
Hash
HMAC

- A copy of the symmetric key is prepended to the message.
- The combination is hashed using a keyless hash function, such as SHA-1.
- The result of this process is an intermediate HMAC which is again prepended with the key (the same key), and the result is again hashed using the same algorithm.
- The final result is an HMAC.
- The receiver receives this final HMAC and the message.
- The receiver creates its own HMAC from the received message and compares the two HMACs to validate the integrity of the message and authenticate the data origin.

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.

- There are two differences between message authentication and entity authentication:
  - First, message authentication may not happen in real time; entity authentication does.
  - 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.

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

  1. **Something known.** This is a secret known only by the claimant that can be checked by the verifier. Examples are a password, a PIN number, a secret key, and a private key.
  2. **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.
  3. **Something inherent.** This is an inherent characteristic of the claimant. Examples are conventional signature, fingerprints, voice, facial characteristics, retinal pattern, and handwriting.

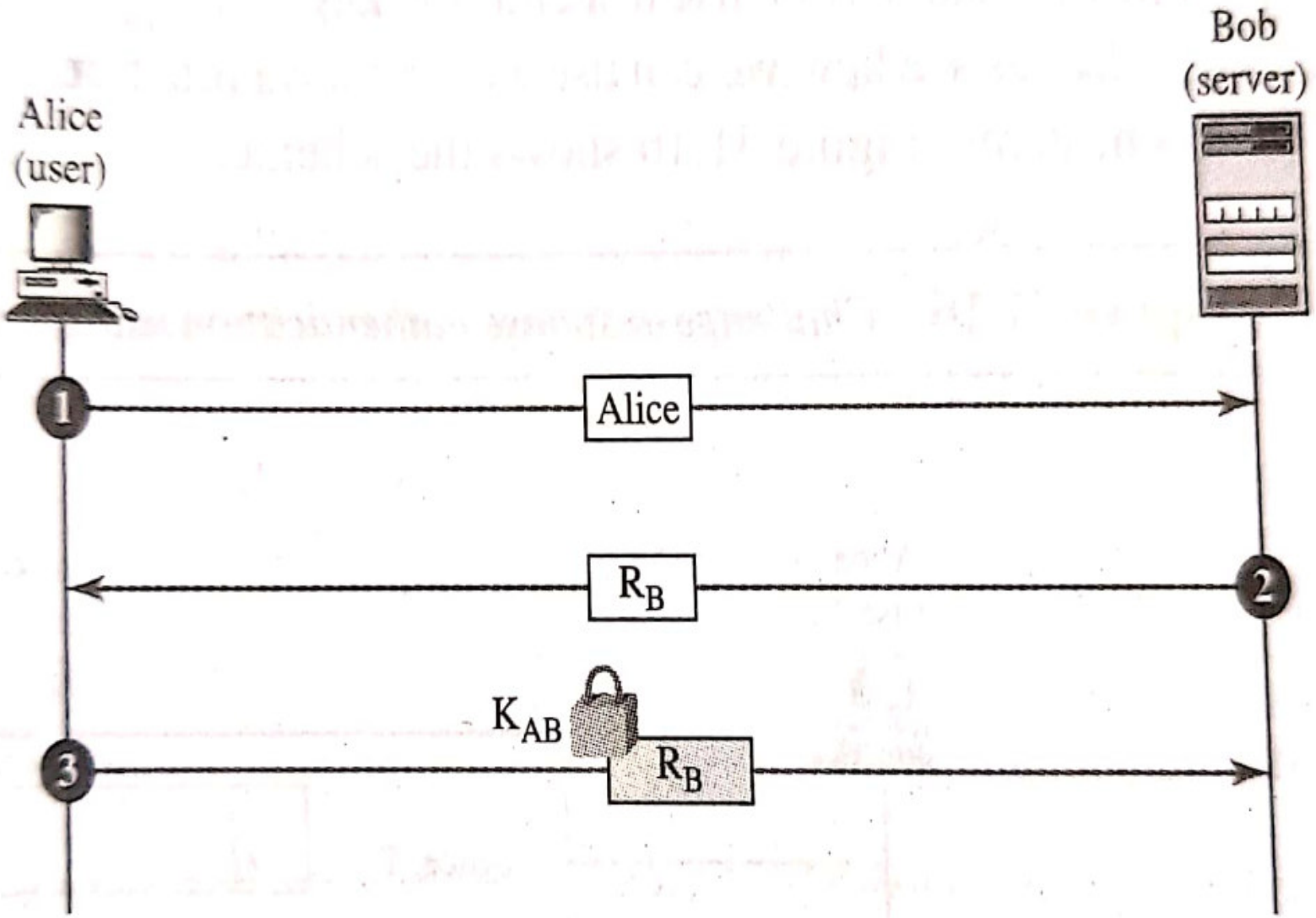
- **Passwords:**
  - The simplest and the oldest method of entity authentication is the password.
  - A password is used when a user needs to access a system to use the system's resources (log-in). Each user has a user identification that is public and a password that is private.
  - We can divide this authentication scheme into two separate groups:
    1. The fixed password and
    2. The one-time password.

1. **Fixed Password:** In this group, the password is fixed; the same password is used over and over for every access. This approach is subject to several attacks.
    - ❖ Eavesdropping.
    - ❖ Stealing a Password
    - ❖ Accessing a file
    - ❖ Guessing
2. **One-Time Password:** In this type of scheme, a password is used only once. It is called the one-time password. A one-time password makes eavesdropping and stealing useless.

- **Challenge-Response**
  - In challenge-response authentication, the claimant proves that she knows a secret without revealing it.
  - The challenge is a time-varying value such as a random number or a timestamp which 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.

❖ **Challenge/response authentication using a nonce:**

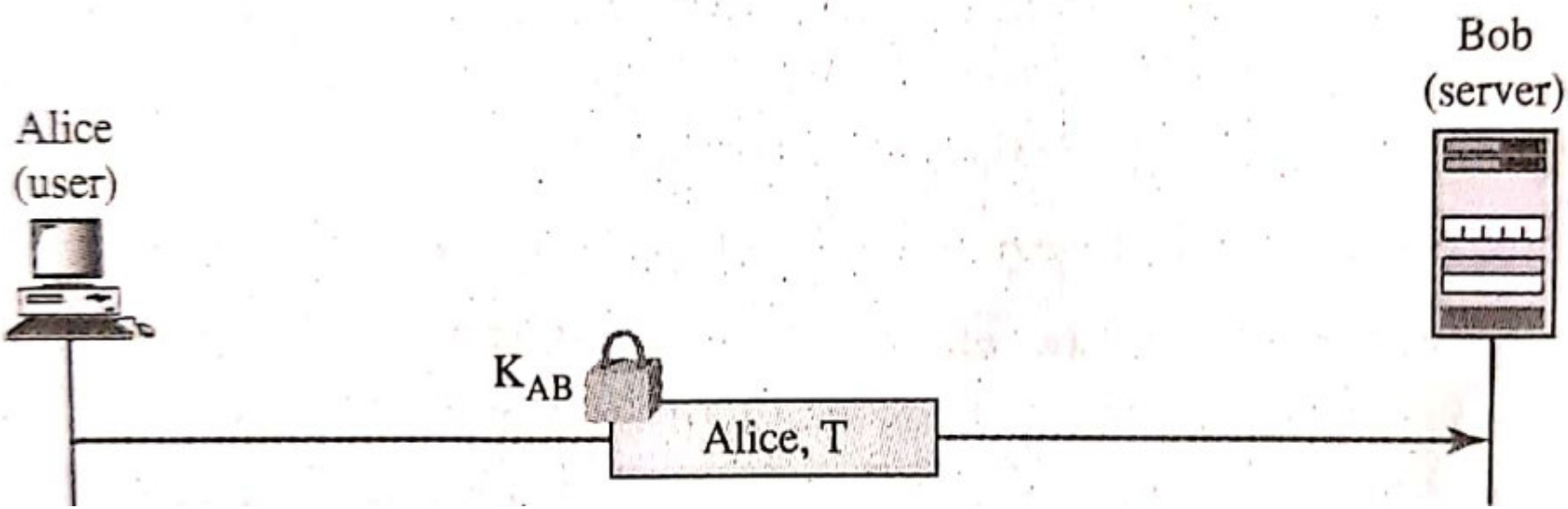
- In this category, the challenge-response authentication is achieved using symmetric key encryption.
- The secret here is the shared secret key, known by both the claimant and the verifier.
- The function is the encrypting algorithm applied on the challenge.


Alice
(user)
Bob
(server)
1
Alice
R_B
2
K_{AB}
3
R_B

- The first message is not part of challenge-response, it only informs the verifier that the claimant wants to be challenged.
- The second message is the challenge. And RB is the nonce randomly chosen by the verifier to challenge the claimant.
- The claimant encrypts the nonce using the shared secret key known only to the claimant and the verifier and sends the result to the verifier.
- The verifier decrypts the message. If the nonce obtained from decryption is the same as the one sent by the verifier, Alice is granted access.

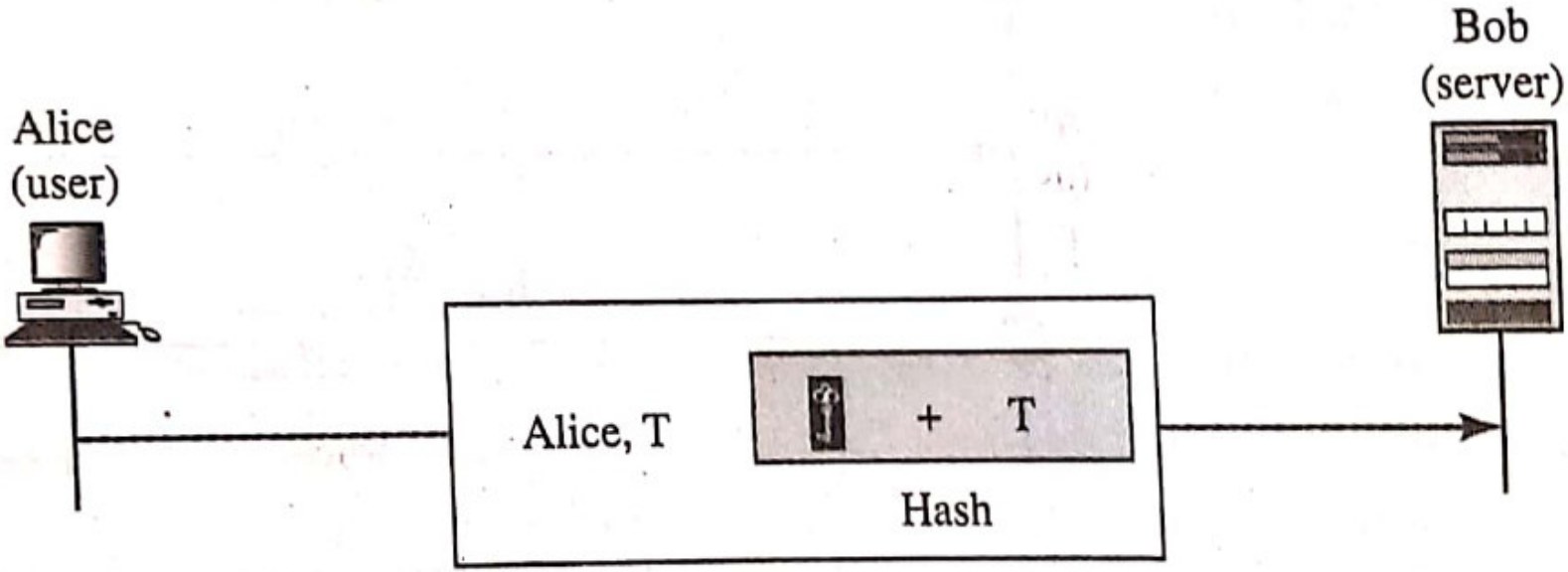
❖ **Challenge-response authentication using a timestamp**

- In this 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.
- 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
(user)
Bob
(server)
K_{AB}
Alice, T

❖ **Challenge-response authentication using Keyed-Hash Functions:**

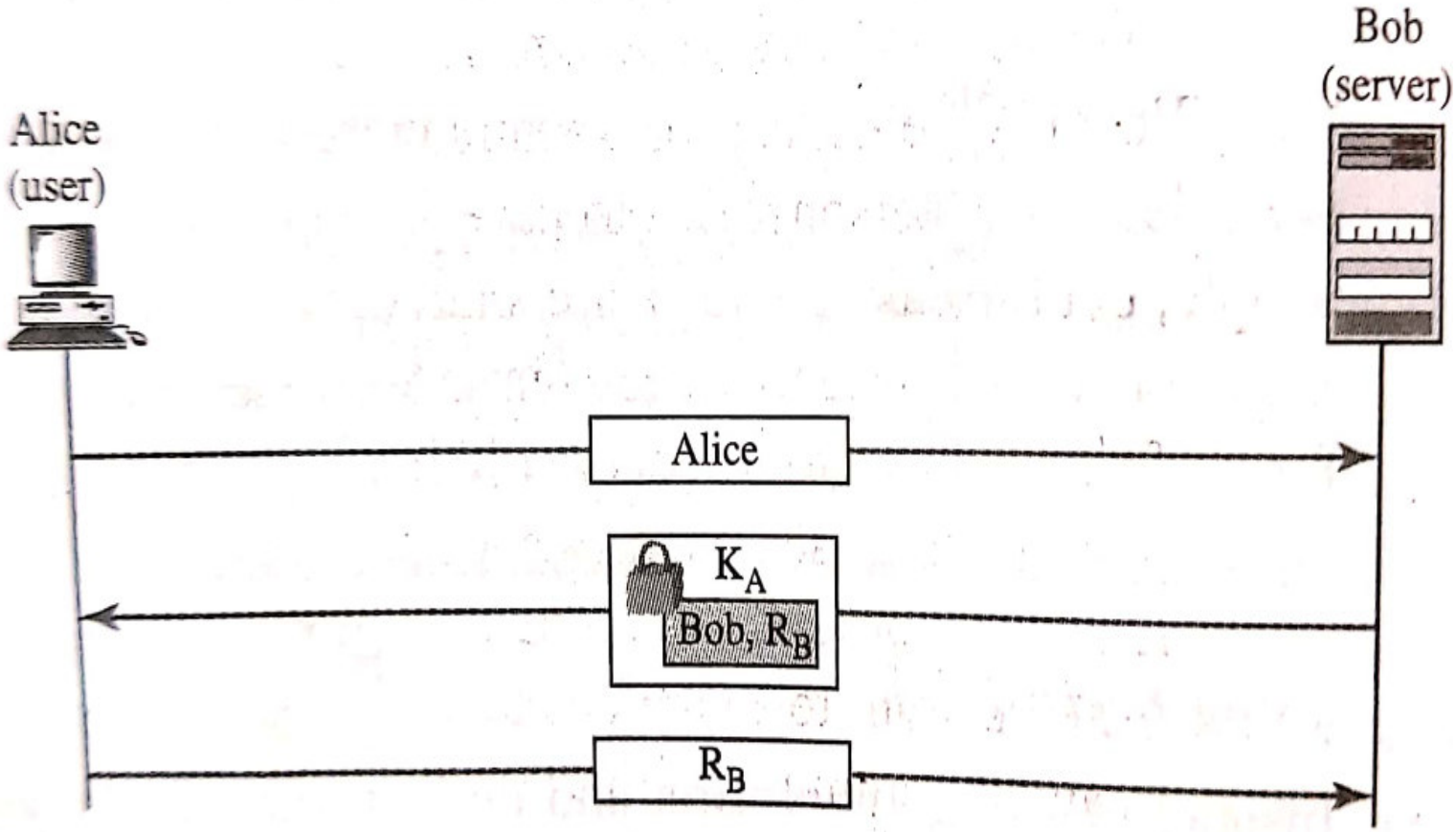
- Instead of using encryption and decryption for entity authentication, we can use a keyed-hash function (MAC).
- There are two advantages to this scheme.
  - ✓First, the encryption/decryption algorithm is not exportable to some countries.
  - ✓ Second, in using a keyed-hash function, we can preserve the integrity of challenge and response messages and at the same time use a secret, the key.


Bob
(server)
Alice
(user)
Alice, T
+
T
Hash

- Note that in this case, the timestamp is sent both as plaintext and as text scrambled by the keyed-hash function.
- When Bob receives the message, he takes the plaintext T, applies the keyed-hash function, and then compares his calculation with what he received to determine the authenticity of Alice.

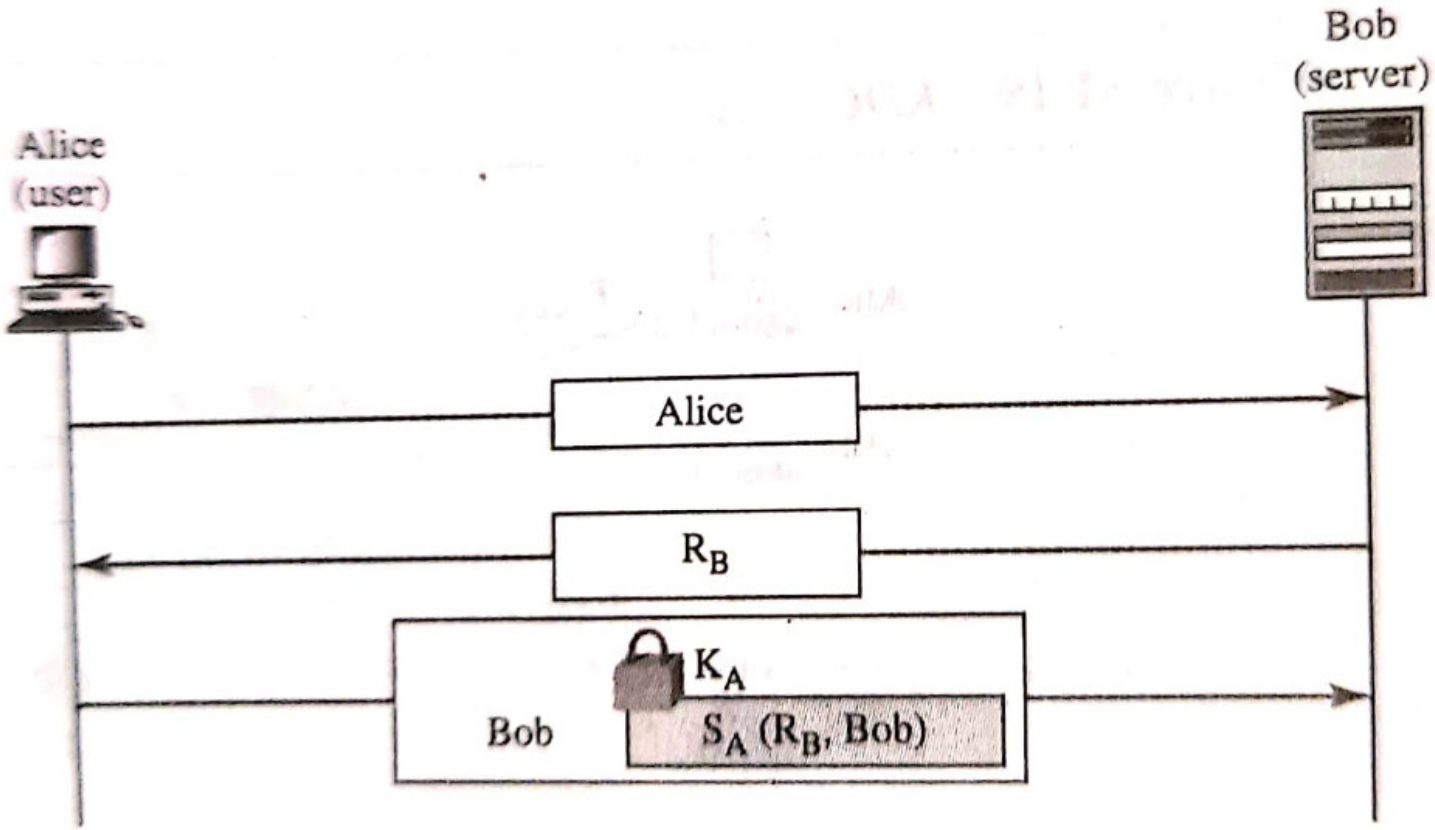
❖ **Challenge-response authentication using an Asymmetric-Key Cipher**

- Instead of a symmetric-key cipher, we can use an asymmetric-key cipher for entity authentication. Here the secret must be the private key of the claimant. The claimant must show that she owns the private key related to the public key that is available to everyone.
- This means that the verifier must encrypt the challenge using the public key of the claimant; the claimant then decrypts the message using her private key. The response to the challenge is the decrypted challenge. We show two approaches: one for unidirectional authentication and one for bidirectional authentication.
- In one approach, Bob encrypts the challenge using Alice's public key. Alice decrypts the message with her private key and sends the nonce to Bob.


Bob
(server)
Alice
(user)
Alice
K_A
Bob, R_B
R_B

❖ **Challenge-response authentication using Digital Signature**

- We can use digital signature for entity authentication.
- In this method, we let the claimant use her private key for signing instead of using it for decryption.



Bob uses a plaintext challenge. Alice signs the response.