Key Management

19CSE311 Computer Security

Jevitha KP

Department of CSE

Symmetric Key Distribution

- Symmetric-key cryptography is more efficient than asymmetric-key cryptography for enciphering large messages.

- Symmetric-key cryptography, however, needs a shared secret key between two parties

- Can be done in two ways:

  - Key Agreement Protocol

  - Key Distribution Center (KDC)

SYMMETRIC-KEY AGREEMENT

- Alice and Bob can create a session key between themselves and is referred to as the **symmetric-key agreement.**

- Diffie-Hellman Key agreement protocol is one such scheme

Math Recap

- What is a Group ?

- Abelian Group ?

Math Basics - Group

- A group (G) is a set of elements with a binary operation "•" that satisfies four properties (or axioms).
- A **commutative group**, also called an **abelian group**, is a group in which the operator satisfies the four properties for groups plus an extra property, commutativity.
- The four properties for groups plus commutativity are defined as follows:
  - **Closure**: If a and b are elements of G, then **c = a • b** is also an element of G (i.e) the result of applying the operation on any two elements in the set is another element in the set.
  - **Associativity**: If a, b, and c are elements of G, then **(a • b) • c = a • (b • c)**. It does not matter in which order we apply the operation on more than two elements.
  - **Existence of identity**: For all a in G, there exists an element e, called the identity element, such that **e • a = a • e = a.**
  - **Existence of inverse:** For each a in G, there exists an element a′, called the inverse of a, such that **a • a′ = a′ • a = e.**
  - **Commutativity**: For all a and b in G, we have **a • b = b • a.** Note that this property needs to be satisfied only for a commutative group.

Math Basics - Group

- A group is called a **finite group** if the set has a finite number of elements; otherwise, it is an **infinite group.**

- **The order of a group, |G|,** is the number of elements in the group.

- Some groups have an interesting property: **all the elements in the group can be obtained by repeatedly applying the group operation to a particular group element.**

- If a group has such a property, it is called a **cyclic group** and the particular group element is called a **generator**.

Math Basics - Generator

- Trivial example : In the group Zn, the additive group of integers modulo n, 1 is always a generator:
  - $1 \equiv 1 \bmod n$
  - $1+1 \equiv 2 \bmod n$
  - $1+1+1 \equiv 3 \bmod n$
  - ...
  - $1+1+1+\ldots+1 \equiv n \equiv 0 \bmod n$

Math Basics - Generator

- For example, Zn*, the multiplicative group modulo n, is cyclic if and only if n is 1 or 2 or 4 or p^k or $2*p^k$ for an odd prime number p and $k \geq 1$.
- So Z5* must be a cyclic group because 5 is a prime number.
- Actually all the elements in Z5*, {1,2,3,4} can be generated by 2:
  - $2^1 \equiv 2 \bmod 5$
  - $2^2 \equiv 4 \bmod 5$
  - $2^3 \equiv 8 \equiv 3 \bmod 5$
  - $2^4 \equiv 16 \equiv 1 \bmod 5$

Diffie-Hellman Key Agreement

- In the Diffie-Hellman protocol two parties create a symmetric session key without the need of a KDC.

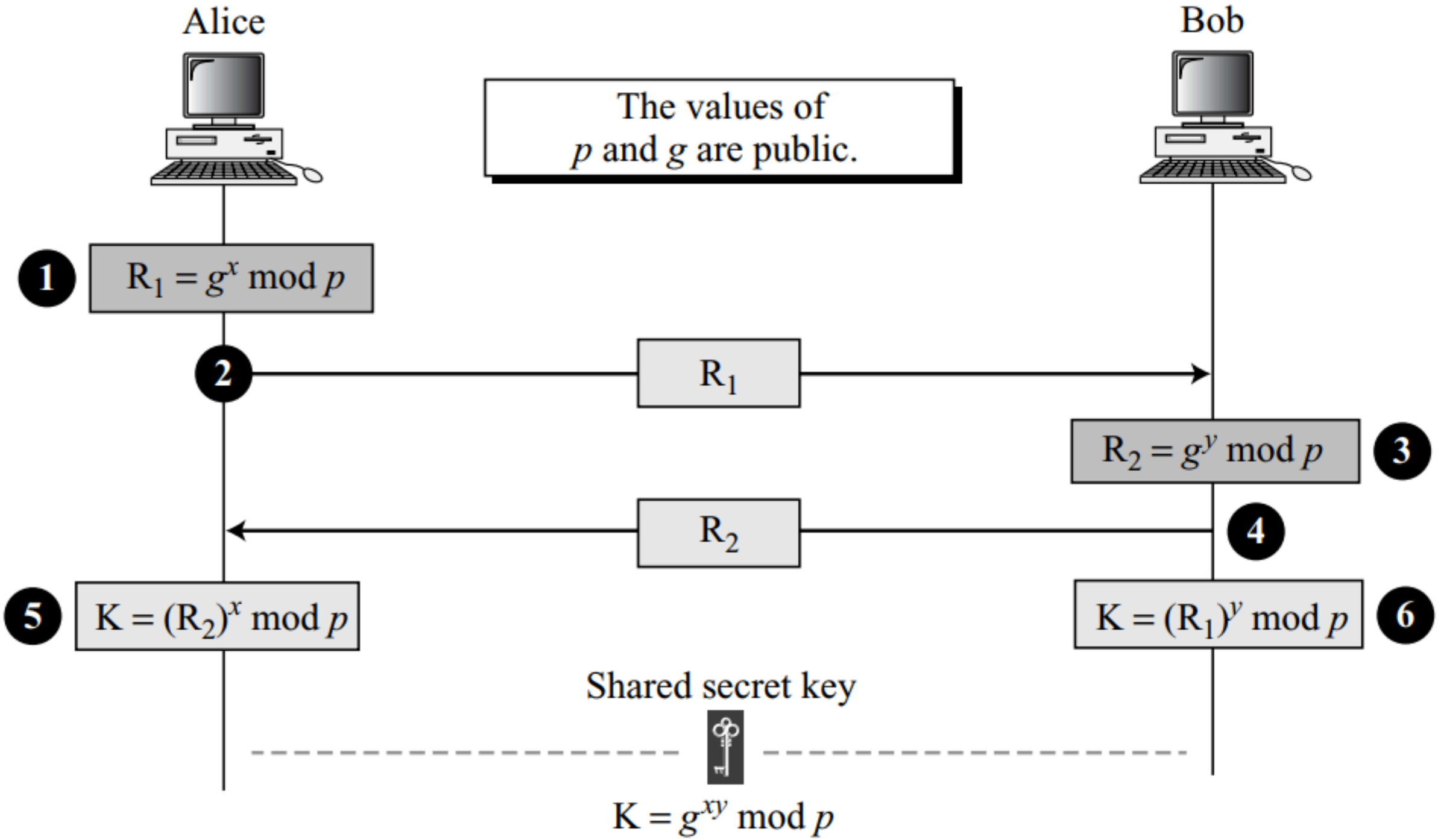
- Before establishing a symmetric key, the two parties need to **choose two numbers p and g.**

- The first number, p, is a large prime number on the order of 300 decimal digits (1024 bits).

- The **second number, g, is a generator of order p – 1** in the group <Zp*, ×>.

- These two (group and generator) do not need to be confidential.

- They can be sent through the Internet; they can be public.


Alice
Bob
The values of
p and g are public.
1
R1 = g^x mod p
2
R1
R2 = g^y mod p
3
R2
4
5
K = (R2)^x mod p
K = (R1)^y mod p
6
Shared secret key
K = g^xy mod p

Diffie-Hellman Key Agreement

- The steps are as follows:
- Alice chooses a large random number x such that $0 \leq x \leq p - 1$ and calculates **$R1 = g^x \bmod p$.**
- Bob chooses another large random number y such that $0 \leq y \leq p - 1$ and calculates **$R2 = g^y \bmod p$.**
- Alice sends R1 to Bob. Alice does not send the value of x; she sends only R1.
- Bob sends R2 to Alice. Bob does not send the value of y, he sends only R2.
- Alice calculates **$K = (R2)^x \bmod p = (R2)^x \bmod p = (g^y \bmod p)^x \bmod = g^{xy} \bmod p = K$**
- Bob calculates **$K = (R1)^y \bmod p = (R1)^y \bmod p = (g^x \bmod p)^y \bmod p = g^{xy} \bmod p = K$**
- **K is the symmetric key for the session**

$$K = (g^x \bmod p)^y \bmod p = (g^y \bmod p)^x \bmod p = g^{xy} \bmod p$$

Example

- Assume that g = 7 and p = 23.
- Compute the shared key using Diffie hellman key agreement protocol.
- Assume x = 3, y = 6

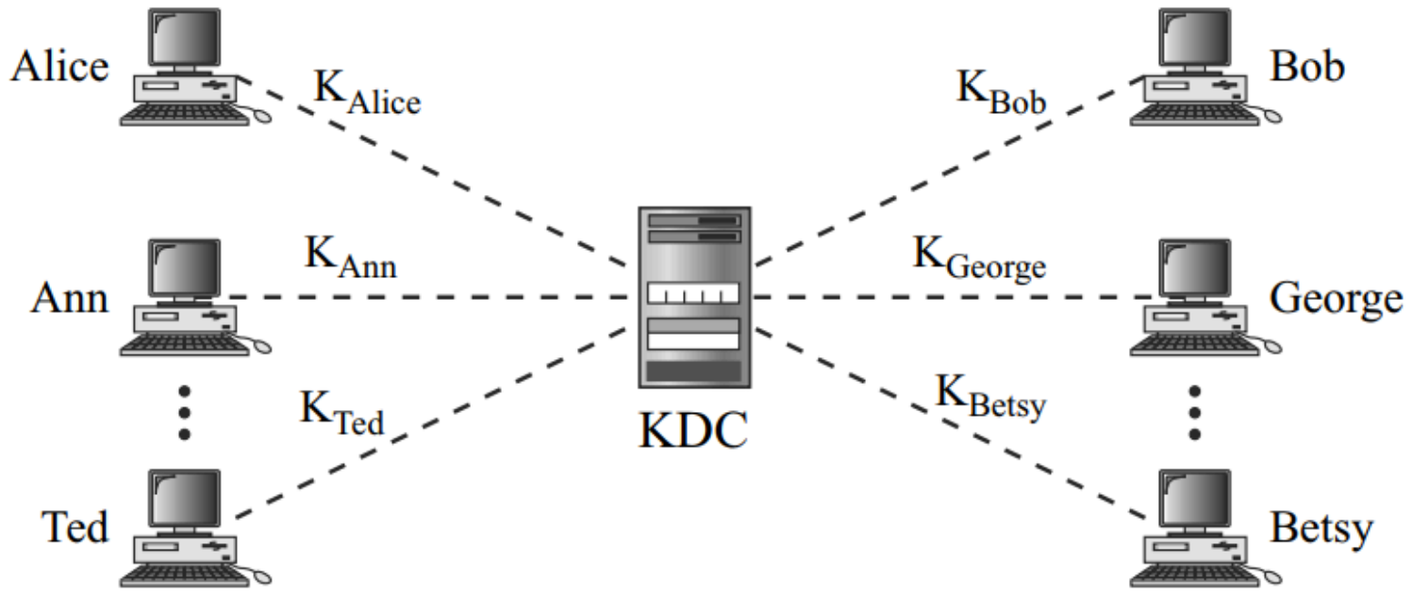
Example

- Assume that g = 7 and p = 23.
- Alice chooses x = 3 and calculates R1 = 7^3 mod 23 = 21.
- Bob chooses y = 6 and calculates R2 = 7^6 mod 23 = 4.
- Alice sends the number 21 to Bob.
- Bob sends the number 4 to Alice.
- Alice calculates the symmetric key K = 4^3 mod 23 = 18.
- Bob calculates the symmetric key K = 21^6 mod 23 = 18.
- The value of K is the same for both Alice and Bob; g^{xy} mod p = 7^{18} mod 35 = 18.

Key Distribution Center (KDC)

- A practical solution is the use of a trusted third party, referred to as a key-distribution center (KDC).

- To reduce the number of keys, each person establishes a shared secret key with the KDC



Key Distribution Center (KDC)

- A secret key is established between the KDC and each member.
- Alice has a secret key with the KDC, which we refer to as KAlice; Bob has a secret key with the KDC, which we refer to as KBob; and so on.
- **Communication Process**
  - Alice sends a request to the KDC stating that she needs a session (temporary) secret key between herself and Bob.
  - The KDC informs Bob about Alice’s request.
  - If Bob agrees, a session key is created between the two.
  - The secret key between Alice and Bob that is established with the KDC is used to authenticate Alice and Bob to the KDC and to prevent Eve from impersonating either of them

KERBEROS

- **Kerberos** is an authentication protocol, and at the same time a KDC, that has become very popular.

- Originally designed at MIT, it has gone through several versions. We discuss version 4, the most popular one here.

Servers

- **Three servers** are involved in the Kerberos protocol:
  - an authentication server (AS),
  - a ticket-granting server (TGS), and
  - a real (data) server that provides services to others.
- In figures, Bob is the real server and Alice is the user requesting service.

Authentication Server (AS)

- The authentication server (AS) is the **KDC** in the Kerberos protocol.

- Each user registers with the AS and is granted a **user identity and a password.**

- The AS has a **database** with these **identities and the corresponding passwords.**

- The AS verifies the user, issues a **session key** to be used between **Alice and the TGS**, and **sends a ticket for the TGS.**

Ticket-Granting Server (TGS)

- The **ticket-granting server (TGS)** issues a **ticket** for the **real server (Bob).**

- It also provides the **session key (KAB)** between **Alice and Bob.**

- Kerberos has separated user verification from the issuing of tickets.

- In this way, though Alice verifies her ID just once with the AS, she can **contact the TGS multiple times to obtain tickets for different real servers.**

Real Server

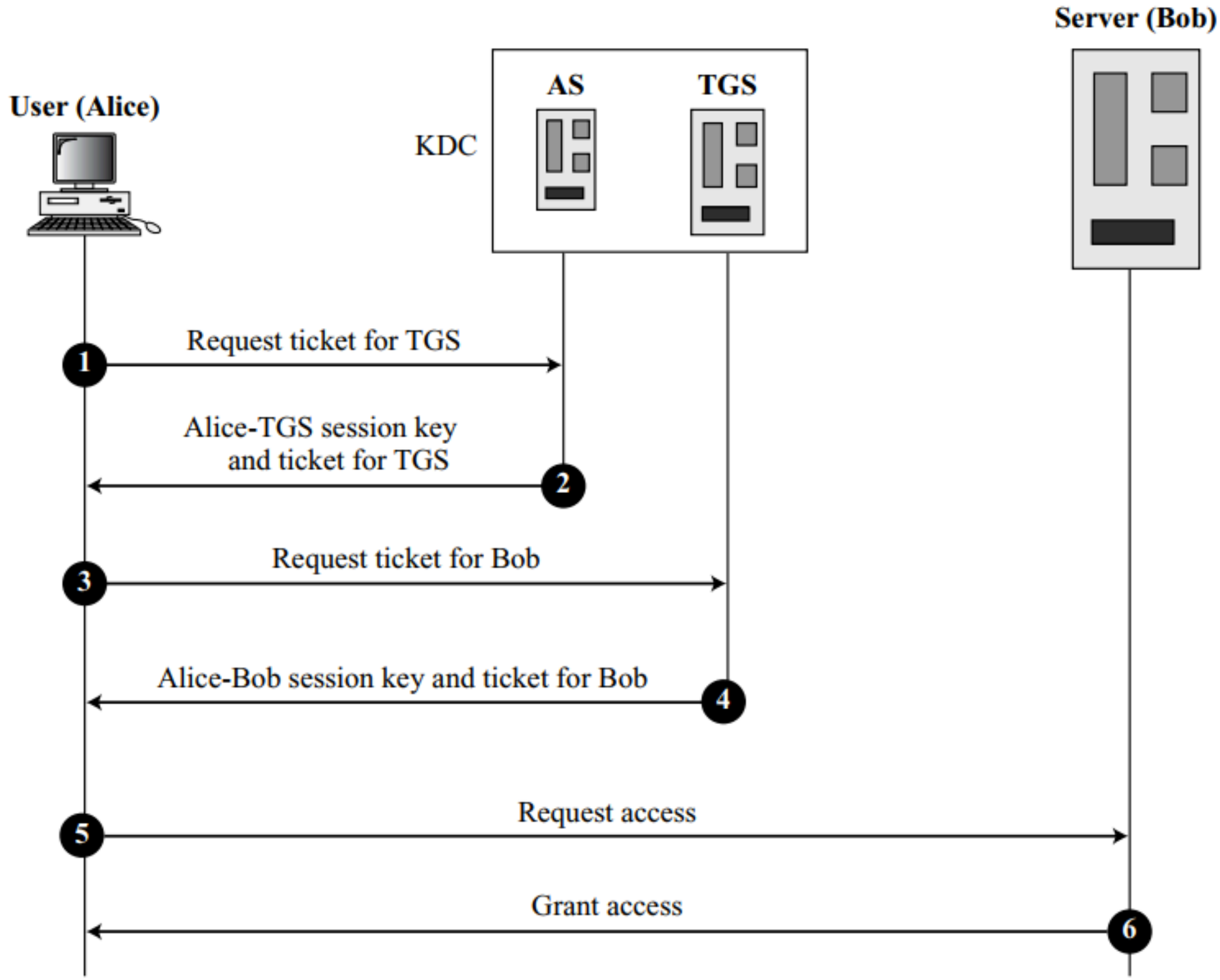
- The real server (Bob) provides services for the user (Alice).

- Kerberos is designed for a client-server program, such as FTP, in which a **user uses the client process to access the server process.**

- Kerberos is not used for **person-to-person authentication.**

AS: Authentication server

TGS: Ticket-granting server



Kerberos Operation

- A client process (Alice) can access a process running on the real server (Bob) in six steps, as shown
- Step 1 : Alice sends her request to the AS in plain text using her registered identity
- Step 2 :
  - The AS sends a message encrypted with **Alice's permanent symmetric key**, **$K_{A\text{-}AS}$.**
  - The message contains two items:
    - **a session key, $K_{A\text{-}TGS}$**, that is used by Alice to contact the TGS, and
    - a **ticket** for the TGS that is encrypted with the TGS symmetric key, **$K_{AS\text{-}TGS}$.**

Kerberos Operation

- Alice does not know $K_{A\text{-}AS}$, but when the message arrives, she types her **symmetric password.**

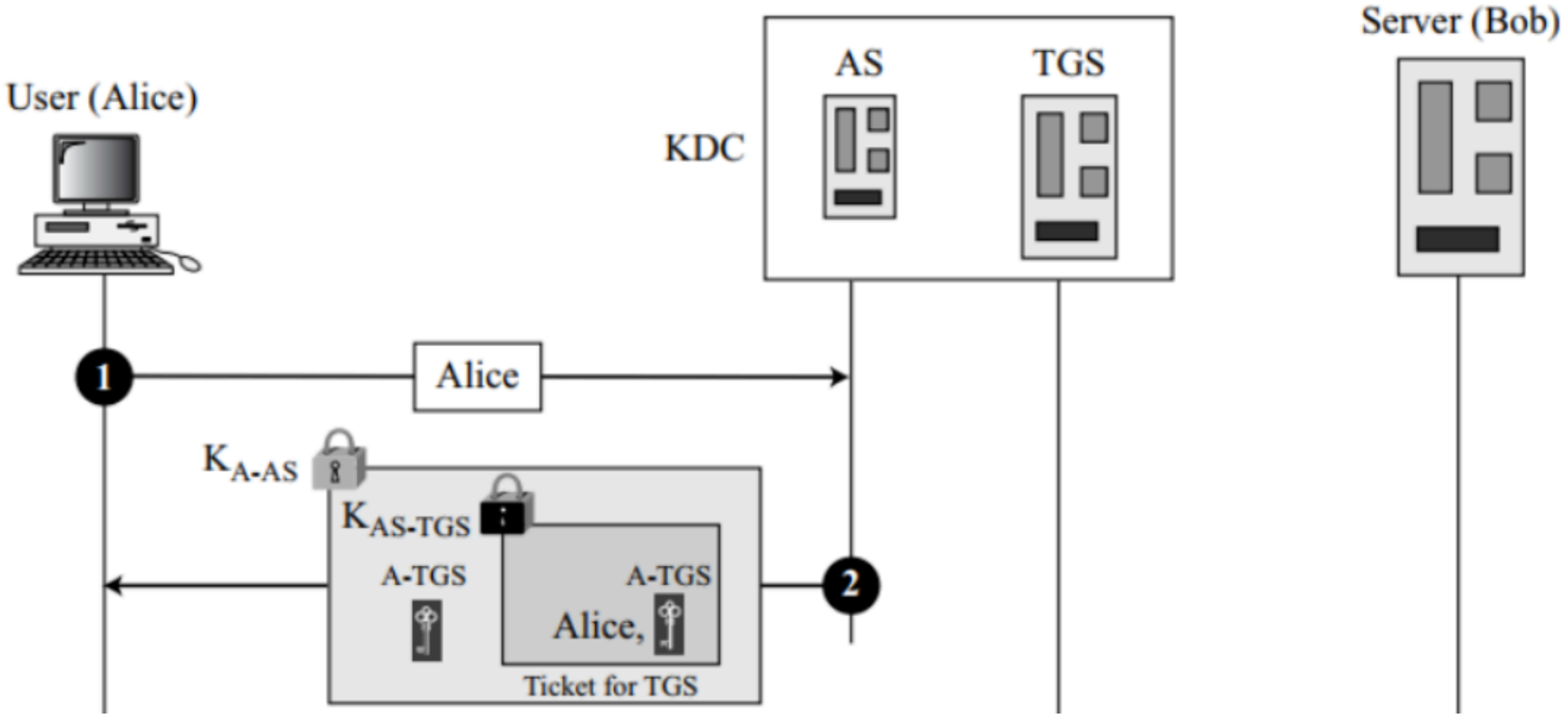
- The password and the appropriate algorithm together create **$K_{A\text{-}AS}$ if the password is correct.**

- The password is then immediately destroyed; it is not sent to the network and it does not stay in the terminal.

- It is used only for a moment to create $K_{A\text{-}AS}$.

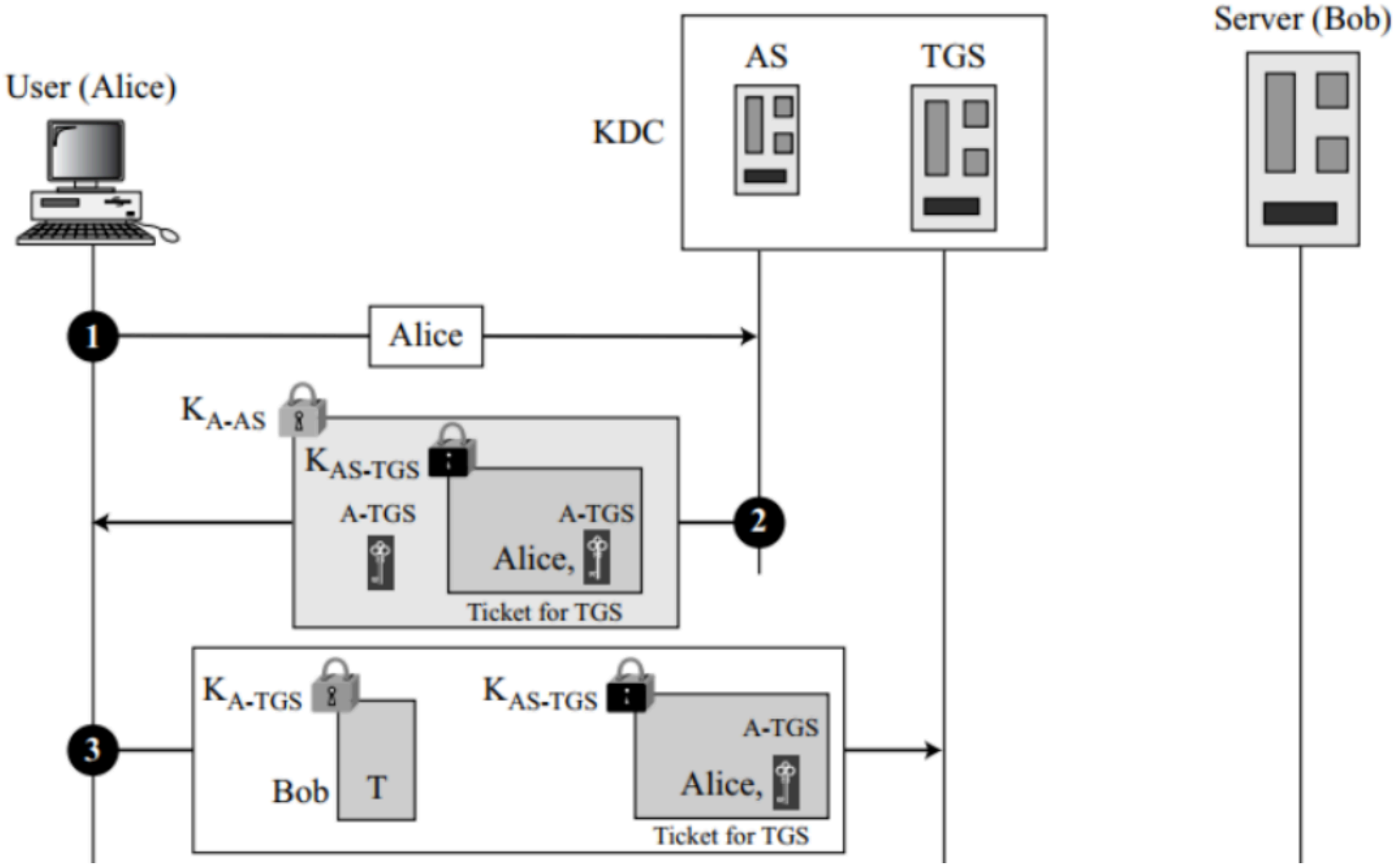
- The process now uses $K_{A\text{-}AS}$ to decrypt the message sent.

- **$K_{A\text{-}TGS}$ and the ticket are extracted.**


User (Alice)
KDC
AS
TGS
Server (Bob)
1
Alice
$K_{A\text{-}AS}$
$K_{AS\text{-}TGS}$
A-TGS
A-TGS
Alice,
Ticket for TGS
2

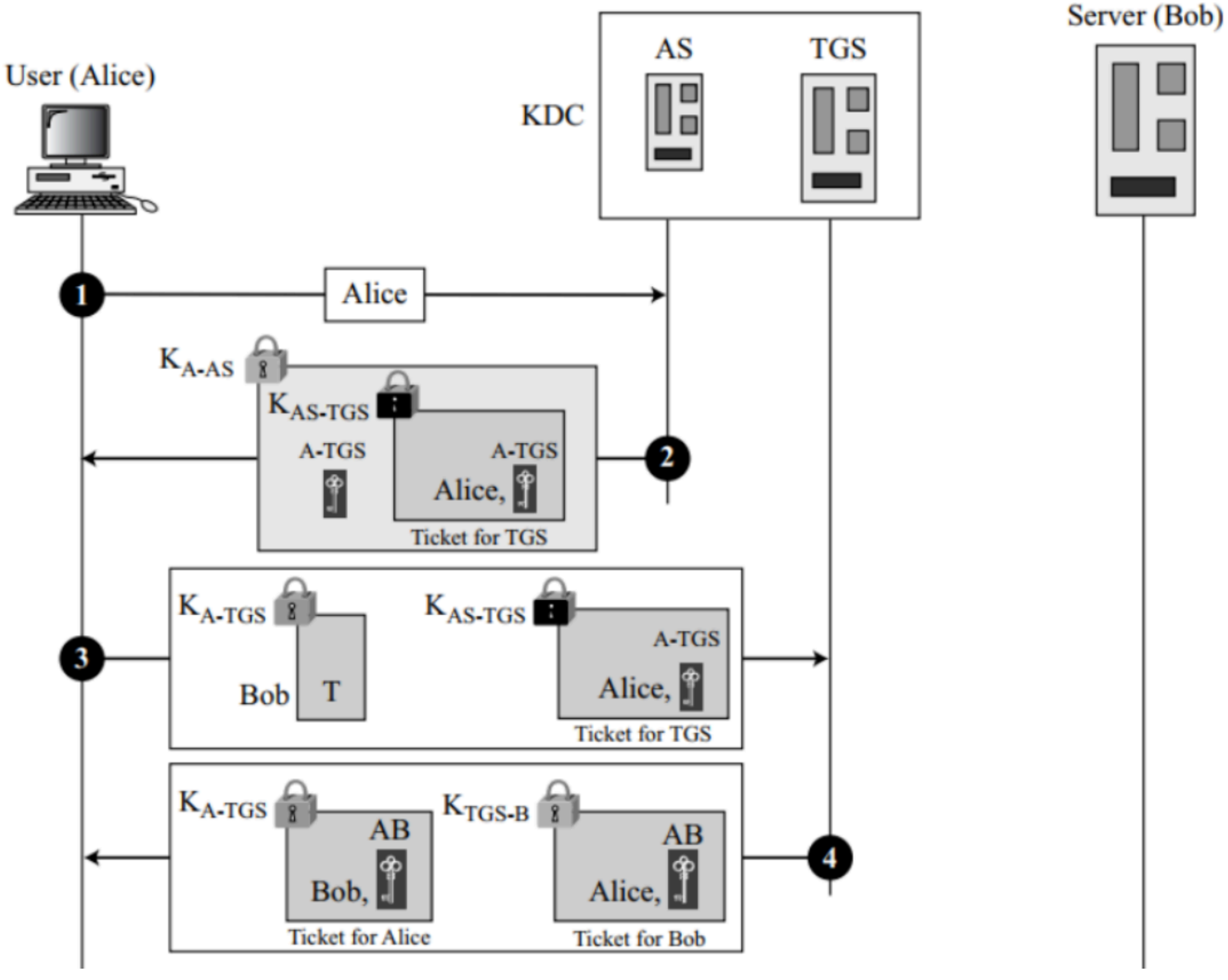
Kerberos Operation

- Step 3 :
  - Alice now sends three items to the TGS.
    - The first is the **ticket** received from the AS.
    - The second is the **name of the real server (Bob)**,
    - the third is a **timestamp** that is encrypted by **$K_{A\text{-}TGS}$.**
    - The timestamp **prevents a replay** by Eve.


User (Alice)
KDC
AS
TGS
Server (Bob)
1
Alice
$K_{A\text{-}AS}$
$K_{AS\text{-}TGS}$
A-TGS
A-TGS
Alice,
Ticket for TGS
2
3
$K_{A\text{-}TGS}$
Bob
T
$K_{AS\text{-}TGS}$
A-TGS
Alice,
Ticket for TGS

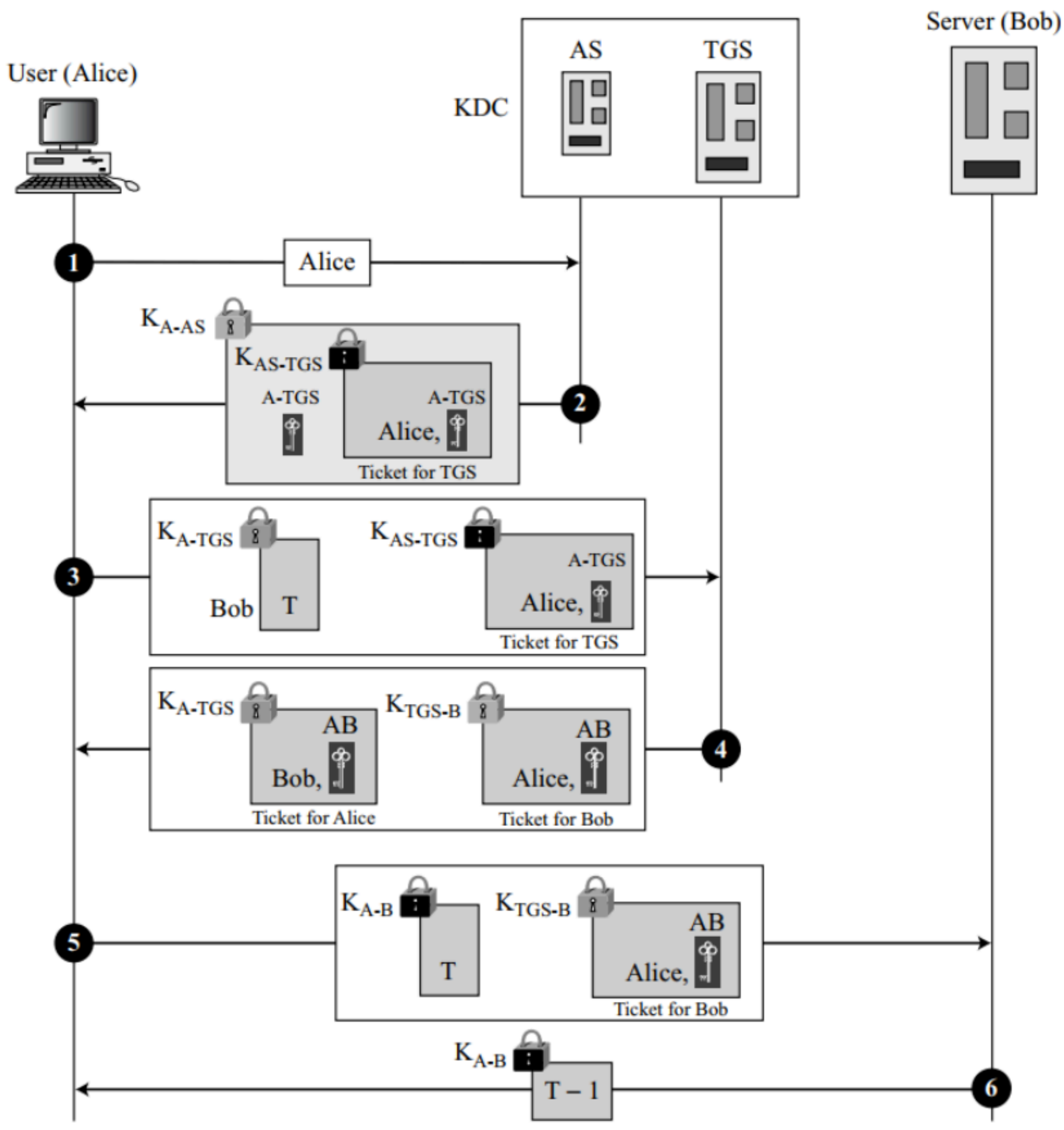
Kerberos Operation

- Step 4 :
  - Now, the TGS sends **two tickets**, each containing the session key between Alice and Bob, K_{A-B}**.**
  - The ticket for Alice is encrypted with $\mathbf{K_{A-TGS}}$; the ticket for Bob is encrypted with Bob's key, $\mathbf{K_{TGS-B}}$**.**
  - Eve cannot extract $\mathbf{K_{AB}}$ because Eve does not know $\mathbf{K_{A-TGS}}$ **or** $\mathbf{K_{TGS-B}}$**.**
  - She cannot replay step 3 because she **cannot replace the timestamp** with a new one (she does not know $\mathbf{K_{A-TGS}}$**).**
  - Even if she is very quick and sends the step 3 message before the timestamp has expired, she still receives the **same two tickets** that she **cannot decipher.**


User (Alice)
KDC
AS
TGS
Server (Bob)
1
Alice
$K_{A\text{-}AS}$
$K_{AS\text{-}TGS}$
A-TGS
A-TGS
Alice,
Ticket for TGS
2
3
$K_{A\text{-}TGS}$
Bob
T
$K_{AS\text{-}TGS}$
A-TGS
Alice,
Ticket for TGS
$K_{A\text{-}TGS}$
AB
Bob,
Ticket for Alice
$K_{TGS\text{-}B}$
AB
Alice,
Ticket for Bob
4

Kerberos Operation

- Step 5 : Alice sends **Bob's ticket** with the **timestamp** encrypted by $\mathbf{K_{A\text{-}B}}$.
- Step 6 :
  - Bob confirms the receipt by adding/subtracting 1 to the timestamp.
  - The message is encrypted with $\mathbf{K_{A\text{-}B}}$ and sent to Alice.


User (Alice)
KDC
AS
TGS
Server (Bob)
1
Alice
$K_{A\text{-}AS}$
$K_{AS\text{-}TGS}$
A-TGS
A-TGS
Alice,
Ticket for TGS
2
3
$K_{A\text{-}TGS}$
Bob
T
$K_{AS\text{-}TGS}$
A-TGS
Alice,
Ticket for TGS
4
$K_{A\text{-}TGS}$
AB
Bob,
Ticket for Alice
$K_{TGS\text{-}B}$
AB
Alice,
Ticket for Bob
5
$K_{A\text{-}B}$
T
$K_{TGS\text{-}B}$
AB
Alice,
Ticket for Bob
6
$K_{A\text{-}B}$
T – 1

Using Different Servers

- If Alice needs to receive services from different servers, she has to repeat only the last four steps.

- The first two steps have verified Alice’s identity and need not be repeated.

- Alice can ask TGS to issue tickets for multiple servers by repeating steps 3 to 6.

Realms

- Kerberos allows the global distribution of ASs and TGSs, with each system called a **realm**.

- A user may get a ticket for a local server or a remote server.

- In the second case, for example, Alice may ask her local TGS to issue a ticket that is accepted by a remote TGS.

- The local TGS can issue this ticket if the remote TGS is registered with the local one.

- Then Alice can use the remote TGS to access the remote real server.