CS255: Computer Security

Distributed Authentication and Authorization

Chengyu Song

Access Control

- Who (subject) gets to access what (object) and how (rights)
  - Identities: subject, object
- Implicit questions/challenges:
  - Naming: how to name identities?
  - Authentication: how to associate entity with identities?
  - Security threats?

Trusted Computing Base (TCB)

- TCB = what is assumed as trusted
  - Cannot be compromised by attackers, always correctly performs the expected functionality
- TCB in access control
  - Hardware reference monitor
  - OS reference monitor
  - Inlined reference monitor

Access Control in Distributed Env

- Consider a networked environment, how to perform access control?
  - Subject? Object?
- What is the TCB?
  - All machines are under strict control ==> nothing special
  - Semi-open environments ==> network address
  - Open environments ==> distributed authentication and authorization
    - Kerberos (symmetric encryption), PKI (asymmetric encryption)

Cryptography Primitives

Perfect Secrecy

- Claude Shannon: the father of information theory
- Basic idea: ciphertext **C** should provide no "information" about plaintext **M**
- Have several equivalent formulations:
  - The two random variables **M** and **C** are independent
  - Knowing what values **C/M** takes does not change what one believes the distribution **M/C** is
  - Encrypting two different messages `m0` and `m1` results in exactly the same distribution

Cryptography Primitives

One-time pad

- **K**: random n-bit key
- **M**: n-bit message (plaintext)
- **C**: n-bit ciphertext
- Encryption: C = M xor K
- Decryption: M = C xor K
- To satisfy perfect secrecy a key can only be used once -> Impractical!

Cryptography Primitives

Block Cipher

- Encrypt/Decrypt messages in fixed size blocks using the same secret key
  - k-bit secret key, n-bit plaintext/ciphertext
- Pseudo Random Permutation
  - How to make one-time pad practical? Use a small secret to generate large quantity of random bits
  - DES, AES
- Operation Modes: ECB (bad, don't use!), CBC, Counter

Cryptography Primitives

Communications in open environment

- Threat model: attackers can **observe**, **modify**, and **replay** messages
- Goals:
  - Confidentiality: $E_s\{m\}$
  - Integrity: $H_s\{m\}$
  - Authenticity: challenge and response
    - Example: send m, expect $D_s\{responds\} = m+1$
    - The secrecy of the key `s`

Cryptography Primitives

Public key cryptography

- Problems with symmetric key: need a secure channel to distribute keys
- Public key (asymmetric key) crypthoghic
  - Use a pair of keys {pk, sk}, where pk can be distributed in open channels
- Trapdoor function (TDF)
  - $E_{pk}\{m\}$ and $D_{sk}\{c\}$ is efficient
  - But given $c=E_{pk}\{m\}$ and pk, it is difficult to find m
- RSA (Rivest–Shamir–Adleman), DH (Diffie–Hellman), ECDH (Elliptic-curve Diffie–Hellman)

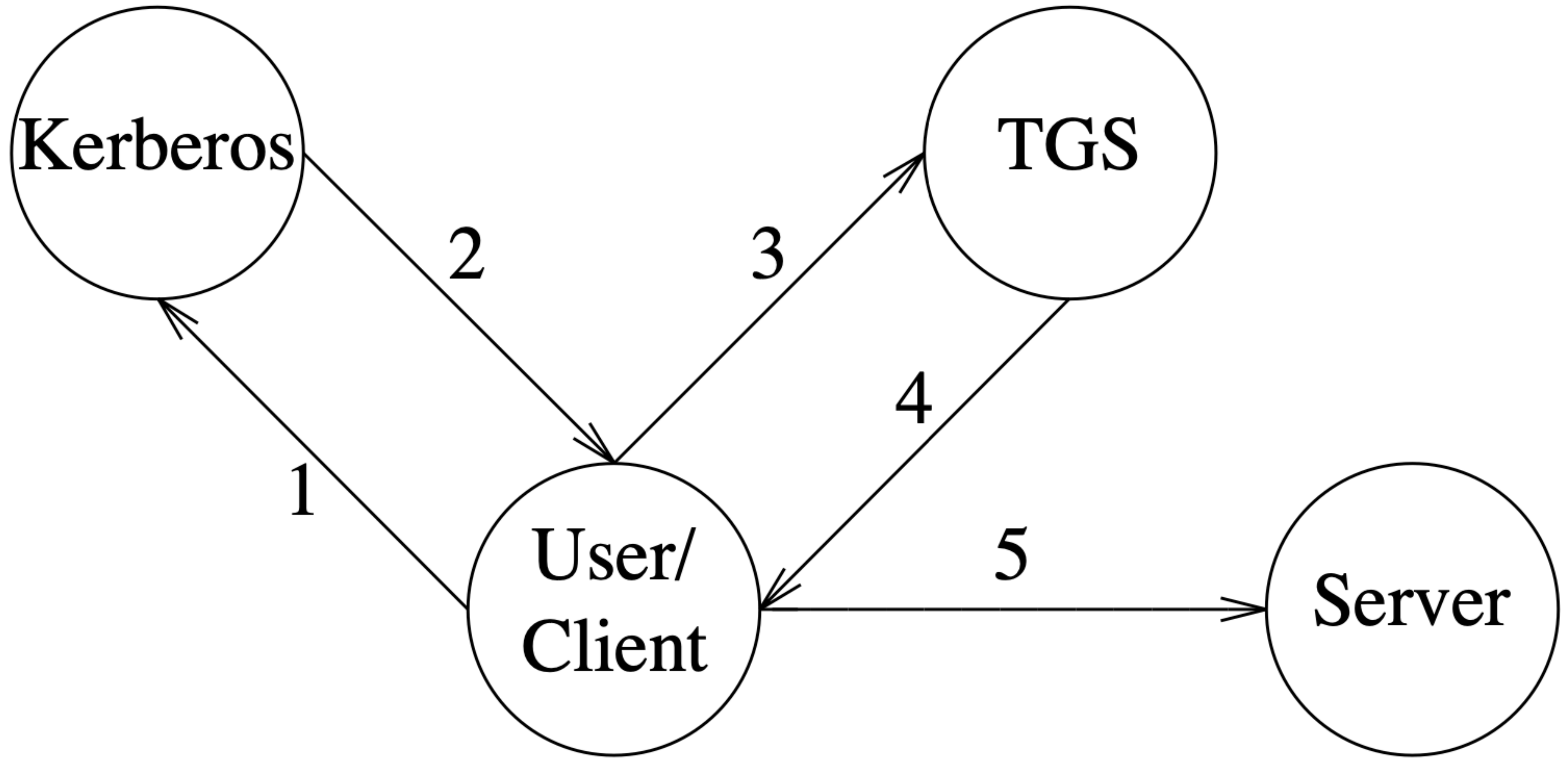
Kerberos

An Authentication Service for Open Network Systems

- Design goals
  - Secure: no impersonation
  - Reliable: no single point of failure
  - Transparent: user should not be aware
  - Scalable: no performance bottleneck
- Assumptions: Kerberos knows the secret key of all services

Kerberos

Overview



1. Request for TGS ticket
2. Ticket for TGS
3. Request for Server ticket
4. Ticket for Server
5. Request for service

Figure 9. Kerberos Authentication Protocols.

Kerberos

Tickets and Authenticators

- Purpose of the ticket
  - Authorization: a valid ticket means **c** can talk to **s**
- Threats?
  - Forgery, reuse
- How to prevent?

$$\{s, c, addr, timestamp, life, K_{s,c}\}K_s$$

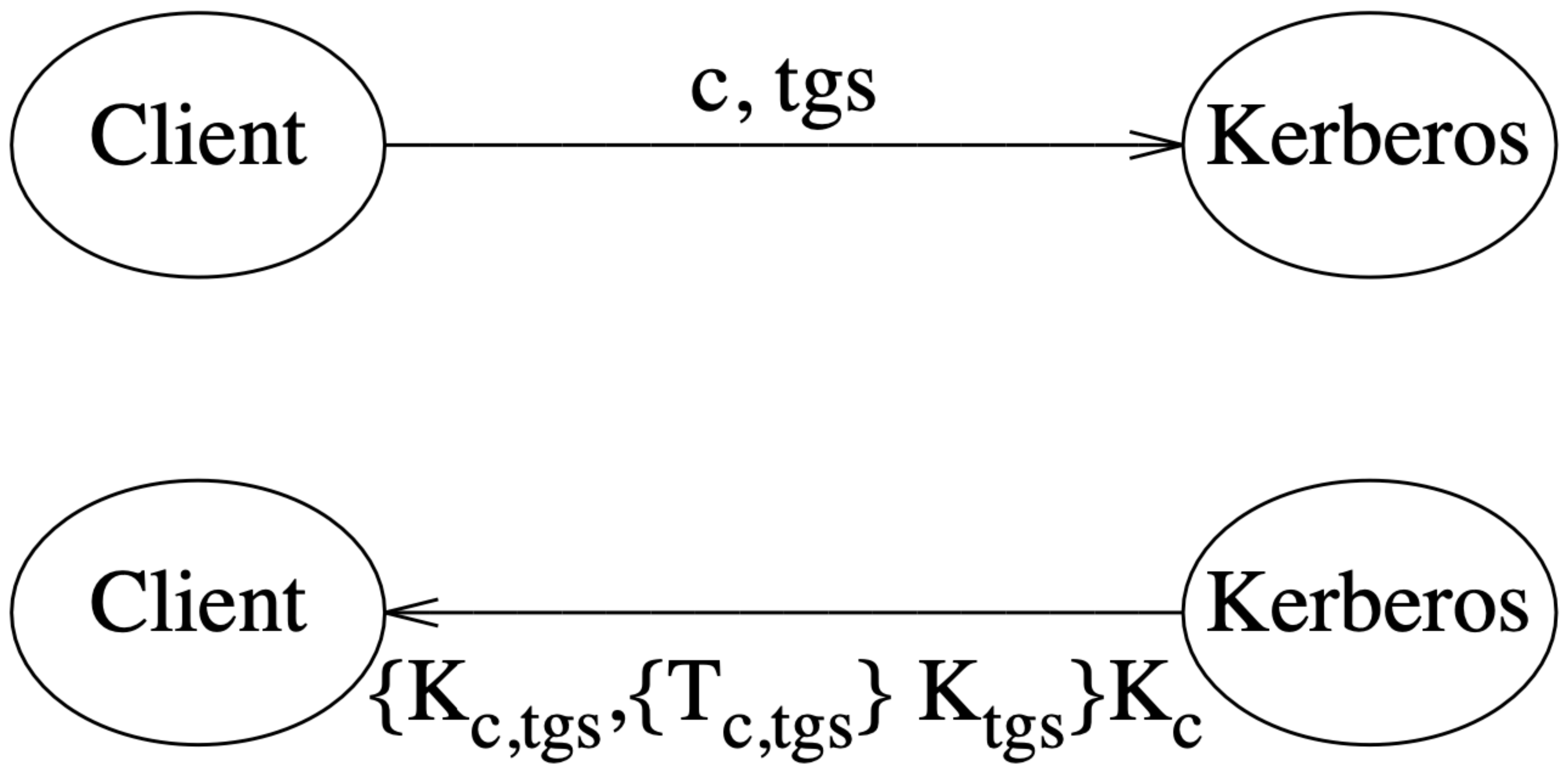
Figure 3. A *Kerberos* Ticket.

$$\{c, addr, timestamp\}K_{s,c}$$

Figure 4. A *Kerberos* Authenticator.

Kerberos

Getting the Initial Ticket



Kerberos

Getting service tickets

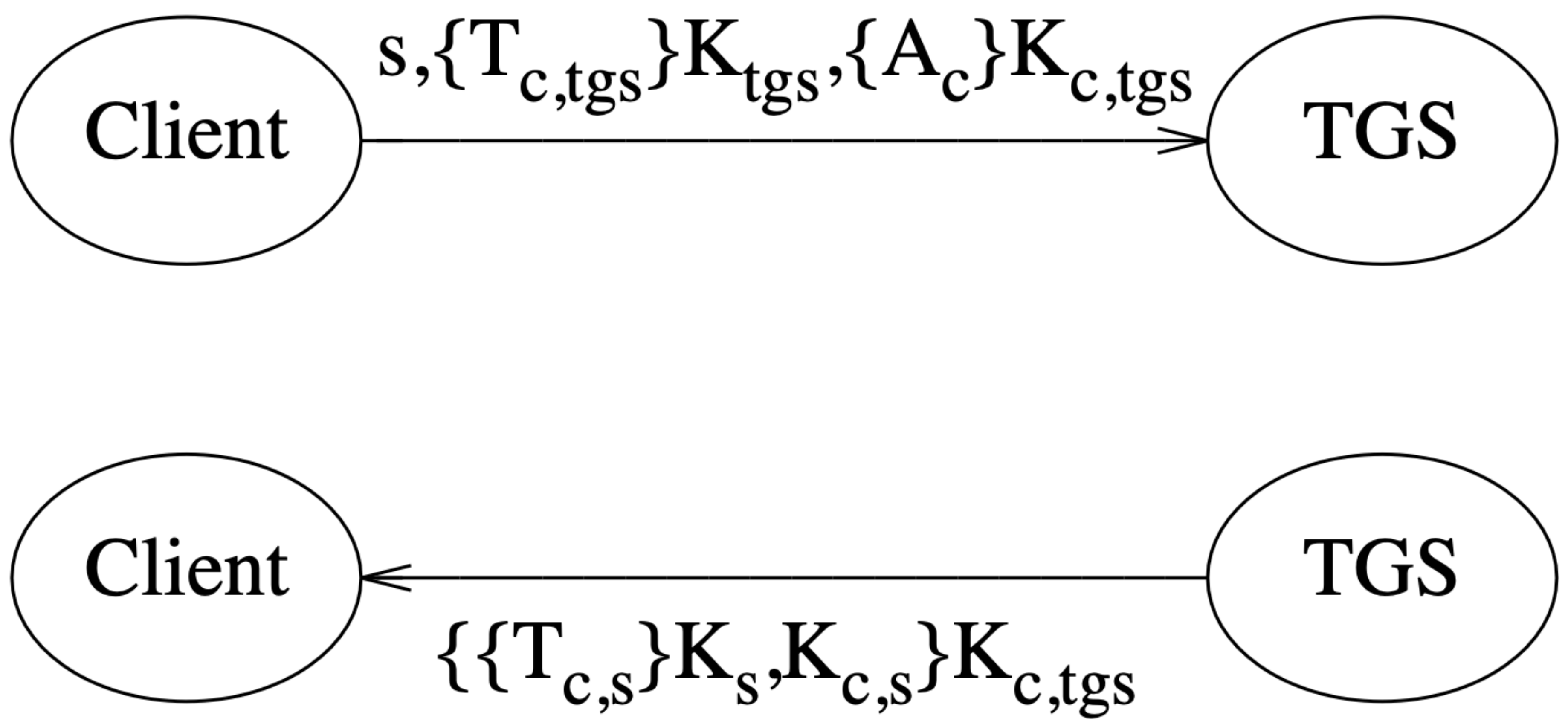


Figure 8. Getting a Server Ticket.

Kerberos

Requesting a service

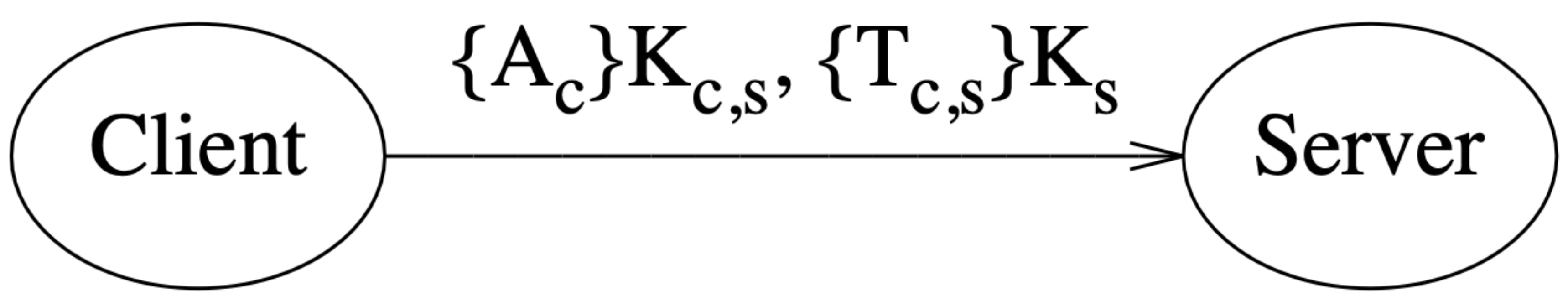


Figure 6. Requesting a Service.

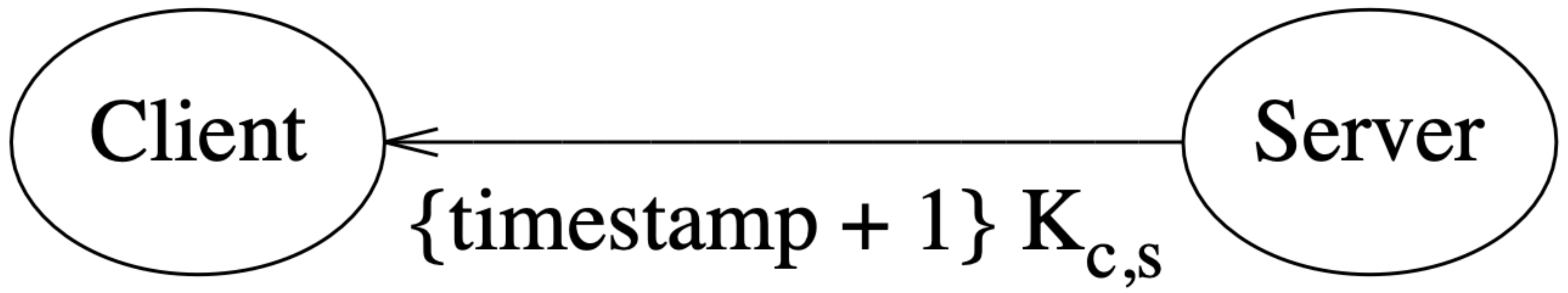


Figure 7. Mutual Authentication.