# CS484 Module 10 Part C - User Authentication - Introduction

Athens State University

**Outline**

## Contents



## 1 Some terms

**Terms: Authentication, Authorization, and Access Control**

- *Authentication*: the act of confirming the truth of an attribute of a single piece of data or entity.
- *Authorization*: the process of verifying that someone or something is permitted to perform some action or task
- *Access control*: the selective restriction of access to a place or other resource

**Terms: Authentication Factors**

- Four means of authenticating a user's identity based upon
  - what they know: password, PIN
  - what they have: key, token, smartcard
  - who they are (static biometrics): fingerprint, retina
  - what they do (dynamic biometrics): voice sign
- Can be used alone or combined, most info. sec. standards require that at least two for strong authentication

# 2 Authentication protocols

### Authentication protocols

- Use to convince participants of identity and to exchange session keys
- May be one-way or mutual
- Key issues:
  - Confidentiality - protection of session keys
  - Timeliness - prevention of replay attacks

### Replay attacks

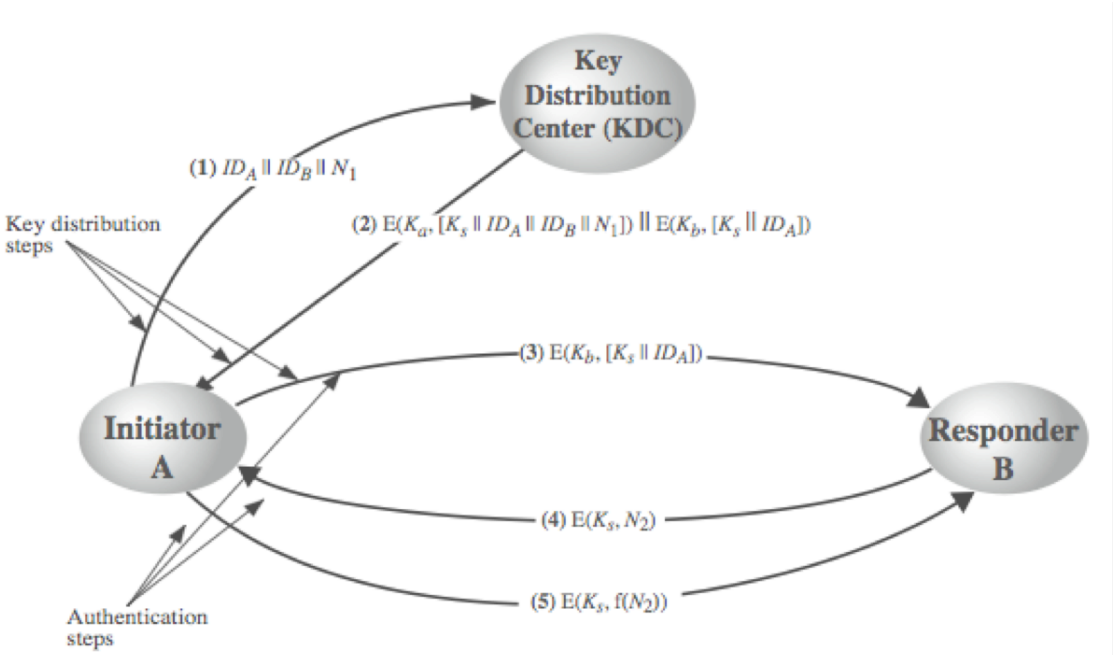
- Where a valid signed message is copied and later reset
  - simple replay
  - repetition that can be logged
  - repetition that cannot be detected
  - backward replay without modification
- Countermeasures
  - sequence numbers (impractical)
  - timestamps (require synchronized clocks)
  - Challenge/Response (using unique nonces)

# 3 Using Symmetric Encryption

### Using Symmetric Encryption

- Use a two-level hierarchy keys much like key distribution
- Assume a trusted KDC
  - Each party shares own master key with KDC
  - KDC generates session keys used for connections between parties
  - Master keys used to distribute session keys to parties

### Needham-Schroeder Protocol




**Needham-Schroeder Protocol**

- Vulnerable to a replay attack if an old session key has been compromised
- One fix can be to use timestamps
- Or one can add an additional nonce
- Shows that one must be very careful about designing security protocols

# 4 Kerberos

**Kerberos: History**

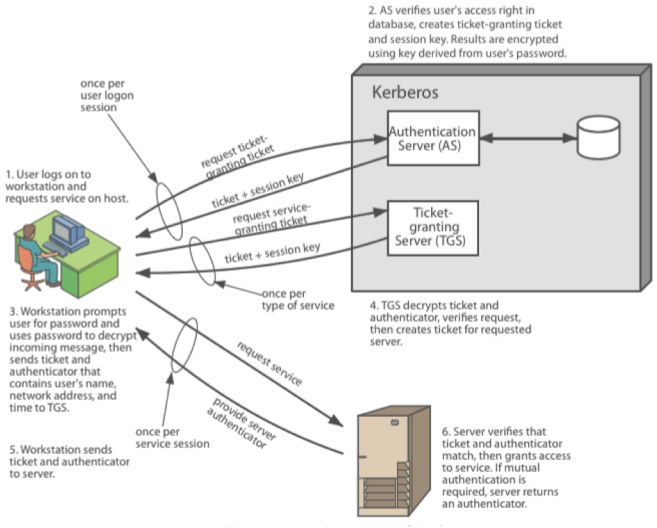
- Developed at MIT in the late 1980s as the security protocol and services for Project Athena (known for introducing thin-client computing, the XWindows graphics system, and other technologies commonly used today)
- Adopted by most modern operating systems
  - Basis of domain-level authentication in Windows 2000 and later
  - Linked with the LDAP directory services in most Linux distributions
  - Two versions in common use: v4 and v5

**Kerberos: Requirements**

- Start from the Needham-Schroder Protocol
- Must be

- secure: eavesdropper on network should not be able to obtain information required to impersonate a user
- reliable: lack of Kerberos for access control means services using it are not available
- transparent: user should not be aware that anything is happening beyond just entering a password
- scaleable: ust be capable of supporting large numbers of clients and servers

**Kerberos: v4: Overview**




**Kerberos: Realms**

- *Realm*: an environment consisting of
  - a Kerberos server
  - a number of clients, all registered with server
  - application severs, sharing keys with servers
- Multiple realms in a network imply that the Kerberos servers must share keys and trust

**Kerberos: v5: Overview**

- developed in the mid-1990s
- improvements over v4
  - switch from DES to AES
  - network protocol
  - resolve byte order issues
- technical issues
  - double encryption, session keys, password attacks, non-std mode of use

# 5 Case Study: Kerberos in ActiveDirectory

### Kerberos in a world of Windows

- Active Directory from Windows 2000 or later uses Kerberos 5 for authentication purposes
- Windows Integrated Authentication is a super-set of the Kerberos and NTLM protocols
- Active Directory hides the bulk of Kerberos within its infrastructure, and thus, is something rarely seen at the user level

### Active Directory Authentication

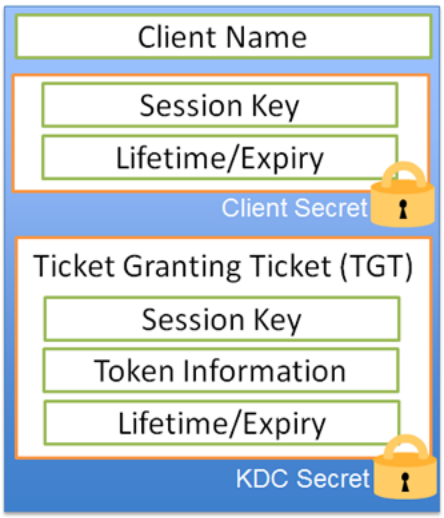

AS_REQ

- The AD DC serves as the KDC in an AD network
- Client log-on requests results in an Authentication Service Request (AS_REQ) containing credentials being sent to the DC
- The Client Time field is a nonce for the request created by using a hash of the user's password to encrypt the current time

### Authentication and clock skew

- The use of client time as the basis for a nonce in the auth request is the rationale behind why AD requires a skew tolerance of 5 minutes in AD networks
- The DC decrypts the encrypted timestamp using its local of the user's password hash
  - If this operations fails, then the request is rejected
  - If the decrypt succeeds and the DC's system time is within 5 minutes of the timestamp, then processing of the request continues

### AD Ticket Granting Tickets



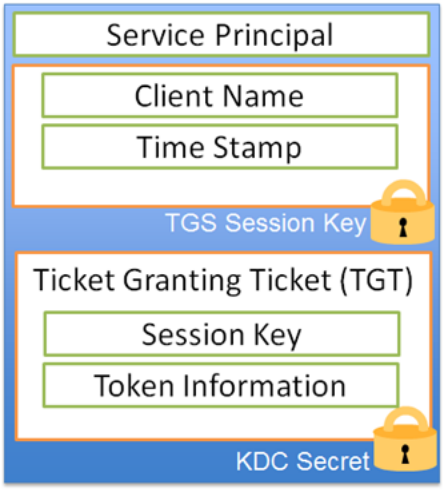
AS_REP

- The DC returns an Authentication Service Reply (AS_REP) back to the client
- The AS_REP includes an embedded TGT and session key
- The client caches the TGT and session key and disposes of the user's password

**Looking closer at AS_REP contents**

- The first component of the reply deals with communication between the client and the DC
  - Data is encrypted using a hash of the user's password
  - The payload contains a session key and ticket expiry timestamp
- The second component contains the TGT
  - This is encrypted using the DC's secret key. This is stored on the DC as the password the DC's `krbtgt` local account
  - This account is created when the first DC in a domain is promoted

**Requesting Service Access**



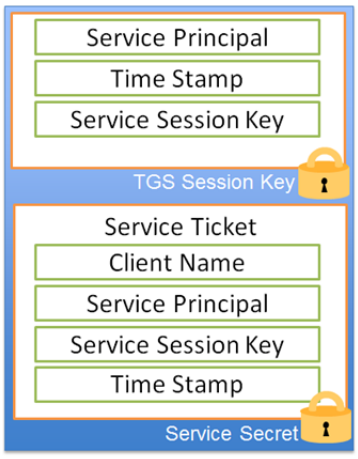
TGS_REQ

- The Service Principal Name (SPN) is name of the service to be accessed
- The client's name and a timestamp are encrypted by session key from the AS_REP message
- And a the encrypted copy of the TGT sent earlier

**Service Principal Names**

- AD maintains a forest-level mapping of default SPNs to every computer account in the directory
- This maps services hosted on a computer to the appropriate SPN, such as those used for CIFS and HTTP services
- One adds additional services to this node in the AD directory tree when needed

**Service Responses**

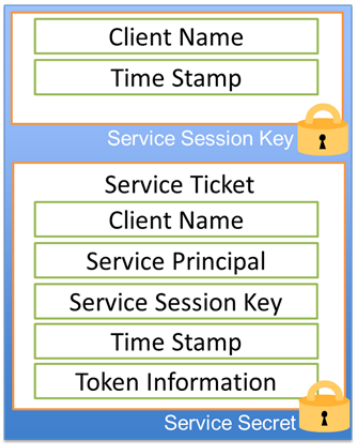


TGS_REP

- The response will include a second session key to use to communicate with the service

- A Service Ticket will be sent in the response that a client will cache and use when it needs to access the service

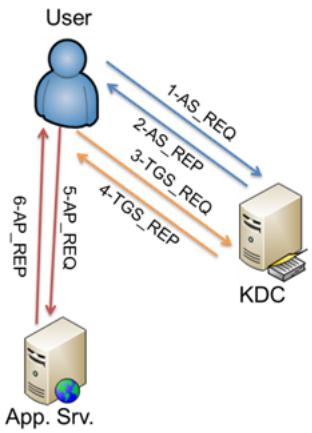
### Accessing Services




AP_REQ

- The AP_REP message is sent when a client needs to access a service
- The service decrypts the service ticket to get the session key
- Which is then used to decrypt the timestamp and client name fields
- Note that we have any authenticated at this point
  - It's up to service to authorize the access (and we go to LDAP)

### AD Authentication Process Overview




1. AS_REQ > KDC
   - AuthN Svc. Req.
2. AS_REP < User
   - AuthN Svc. Rep.
3. TGS_REQ > KDC
   - Tkt Gnt'ing Svc Req
4. TGS_REP > User
   - Tkt Gnt'ing Svc Rep
5. AP_REQ > App
   - App. Req.
6. AP_REP < User
   - App. Reply
   - *Optional*

# 6 Identity Management

**Identity Management: Definition**

- *Identity management*: management of individuals, the authentication, authorization, and privileges across system and enterprise boundaries
- Releated terms and services: Active Directory, OpenID, OAuth, and SAML

**Identity Management: Functions**

- The pure identity function: creation, management, and deletion of identities without regard to access or entitlement
- The user access function (log-on): example: use of data on a smart card to connect to service or services
- The service function: services, local or remote, require identity management services to control access to digital assets

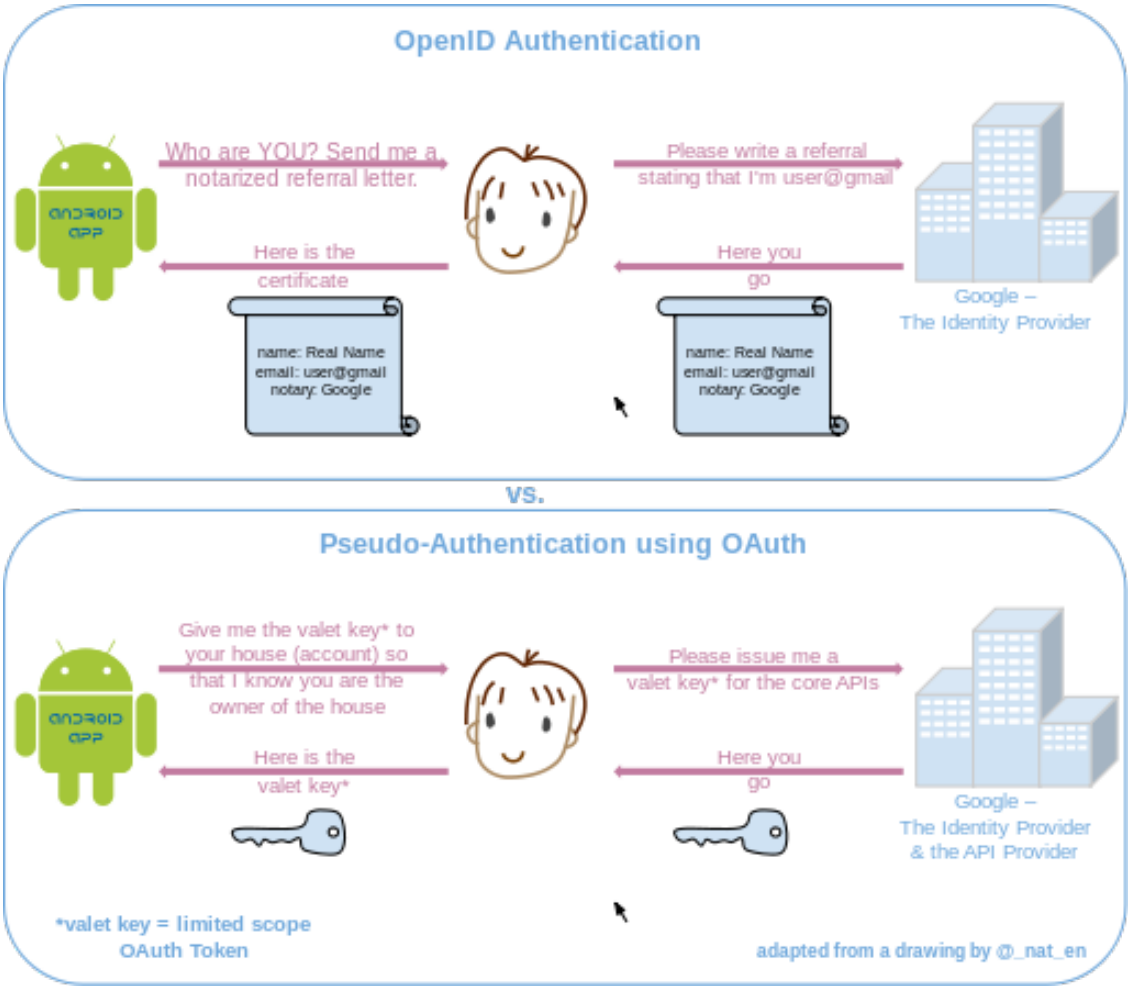
**Identity Management: Single sign-on**

- Property of access control of multiple related but independent software systems
- Traditional: Kerberos-based
  - Initial sign-on prompts for credentials, and gets a Kerberos TGT
  - The TGT is used to acquire service tickets to other services on the network (e-mail client, wiki, revision control)
  - Seen in both Windows and Linux environments

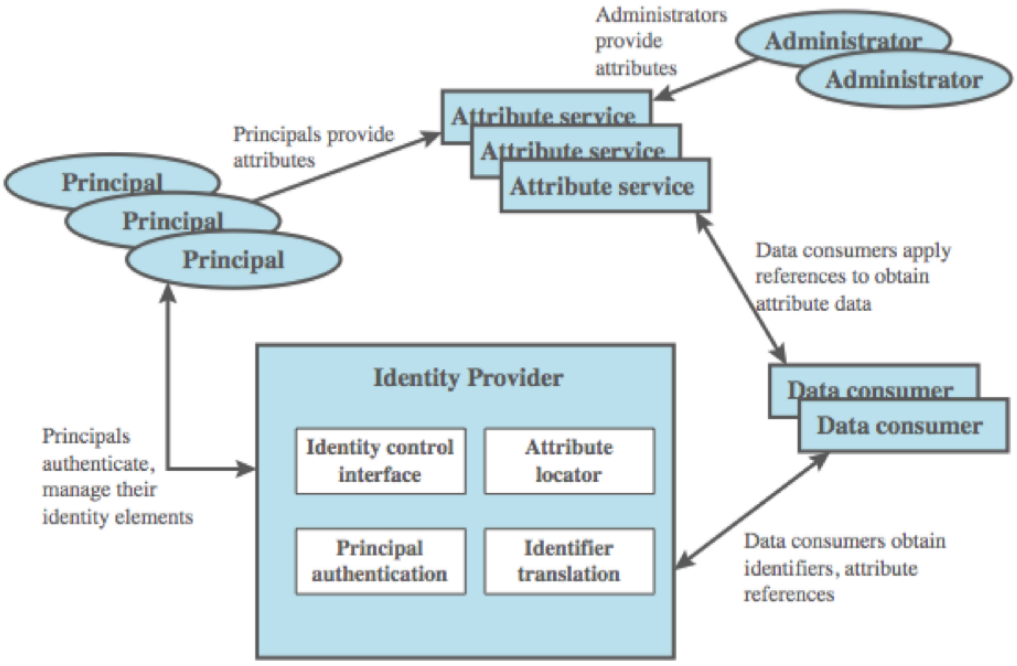
**Identity Management: Federated Identity Management**

- Single sign-on is the authentication subset of a federated identity management system
- Common set of policies, practices, and protocols to manage the identify and trust of users across enterprises
- Built on open standards: OAuth, OpenID, SAML
- Examples: Google ID, Windows Live ID, Facebook Connect (to an extent)

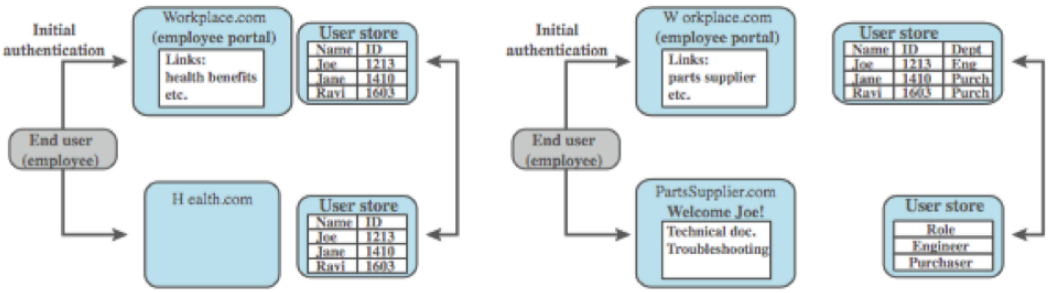
**Identity Management: Use cases from OpenID and OAuth**




**Identity Management: Data Flows**

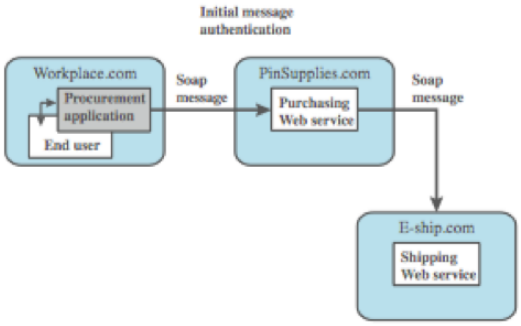



**Identity Management: Workflow**




(a) Federation based on account linking

(b) Federation based on roles



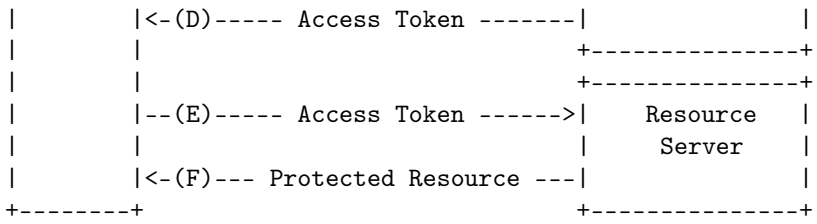

(b) Chained Web Services

### Identity Management: OAuth 2.0

- Four roles:
  - *resource owner*: entity capable of granting access to a protected resource
  - *resource server*: server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens
  - *client*: application making protected resource requests on behalf of a resource owner and within its authorization
  - *authorization server*: server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization

### Identity Management: OAuth 2.0

```
+--------+                                +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                                |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                                +---------------+
|        |                                +---------------+
|        |--(C)-- Authorization Grant -->| Authorization |
| Client |                                |     Server    |
```




# 7 Key Points

**Key Points**

- User authentication issues
- Using symmetric encryption
- The Kerberos trusted key server system
- Evolution of federated identity management