# IT Security


## Chapter 4: Authentication and Authorization

- Authentication
- Practical aspects of authentication
- Access control
- Access control procedures

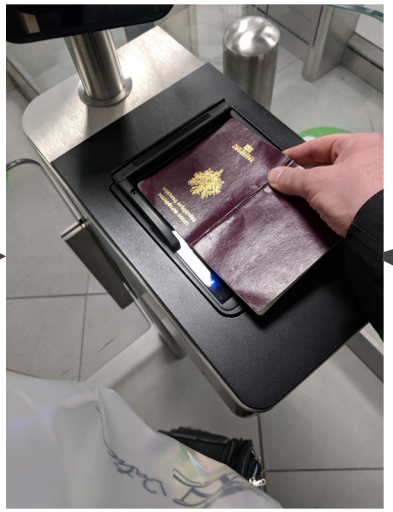
**Prof. Dr. Reiner Hüttl, TH Rosenheim, © 2023, 15.03.2023**

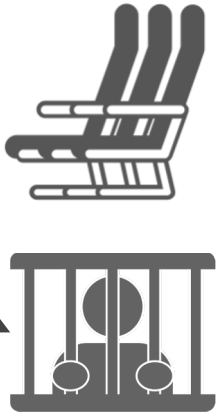
# What do we want to learn?

- What is the difference between authentication and authorization?
- What variants are there for authentication?
- What variants are there for authorization?
- What all needs to be taken into account during practical implementation?

# Identification, Authentication, Authentification and Authorization

This distinction is important!



*Identification*
provide identity

Authentication
assert identity

*Authentification*
validate the assertion

*Authorization*
execute an access policy

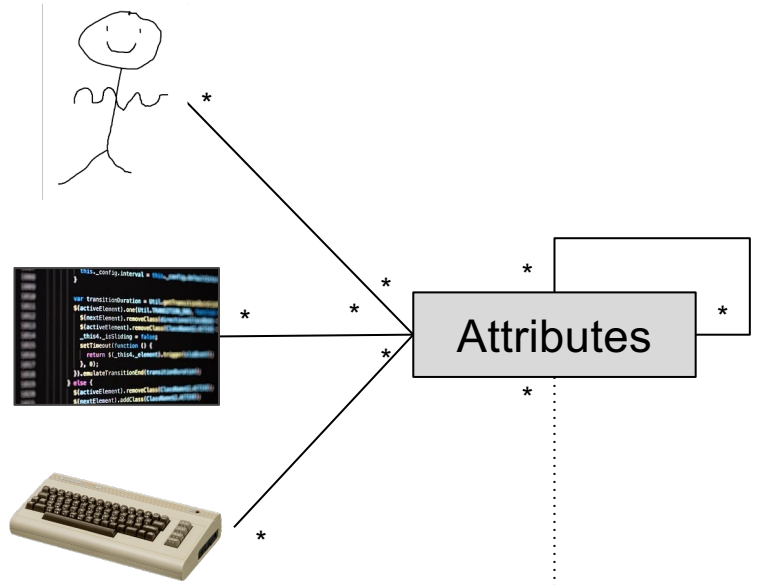
# Definitions for Authentication and Authorization

**Authentication:**
Authentication is the process of verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system. (NIST)

**Authorization:**
Authorization is the process of checking whether a person, IT component or application is authorized to perform a specific action or access a resource.

# Distinguish between identity and policy

Identity enables authentication




Policy enables authorization

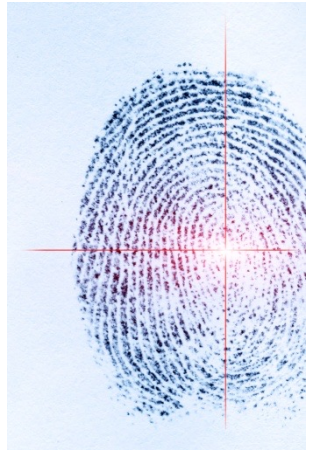
# Variants for authentication

- Authentication is possible through
  - knowledge
  - possession
  - personal characteristic
- Password procedure
- Biometry
- Challenge-Response method
- Certificates / signatures
  - based on a PKI

# Password authentication

- Widely used, easy to implement, mobile
- Problems
  - password cracking
  - if someone knows the password, he has unrestricted access
  - forgetting passwords → Secure recovery procedure is required
  - unencrypted transmission over networks
- Measures to increase security
  - minimum length, special characters, regular changes???
  - blocking/delaying the identifier after a small number of failed attempts
  - display of last login
  - storage as hash value
  - use of a salt (randomly selected bit sequence) in the calculation of the hash value
- Additional measures
  - One-time passwords, TAN (transaction numbers), e.g. online banking
  - NONCE, captcha, security question, time stamp

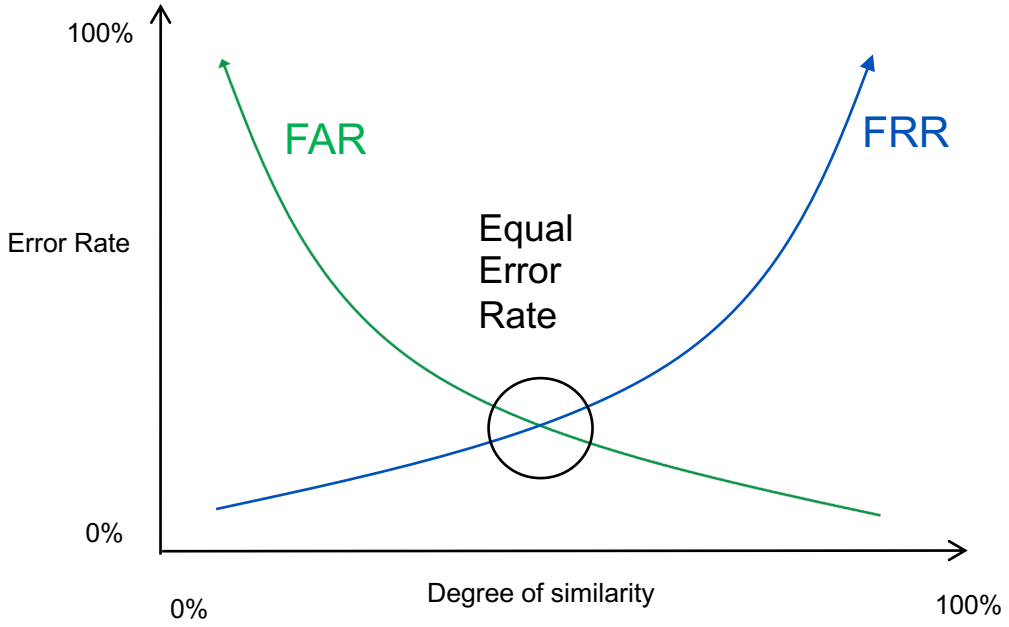
# Biometric techniques



- Fingerprint, face recognition, hand recognition, iris, retina, typing behavior, voice, signature recognition.
- Enable unique identification of individuals
- Problems
  - can be forged, intercepted, violently misused
  - import of biometric data bypassing the biometric sensor
  - some biometric characteristics cannot be exchanged
  - unnoticed collection and area-wide monitoring possible
  - security problem: reference database
  - pretend 100% security

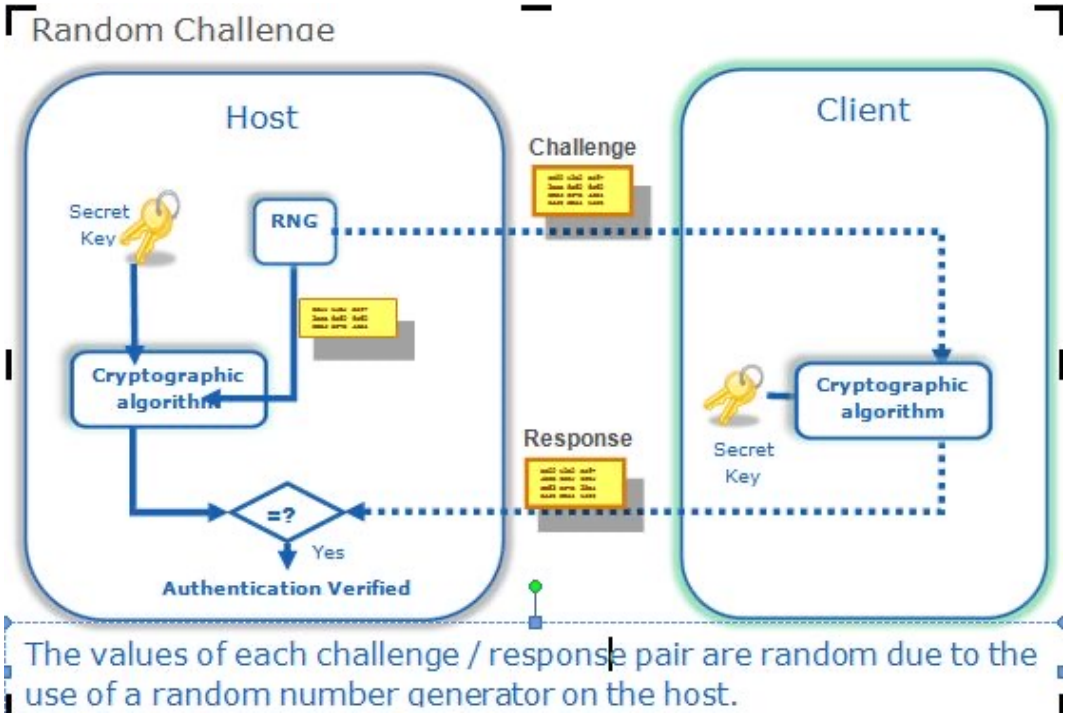
# There are two problems with the verification of biometric authentication

- **False Acceptance Rate FAR**
  - Falsely accepting an unauthorized person (false positive)
- **False Rejection Rate FRR**
  - Falsely rejecting an authorized person(false negative)



# Challenge Response Method

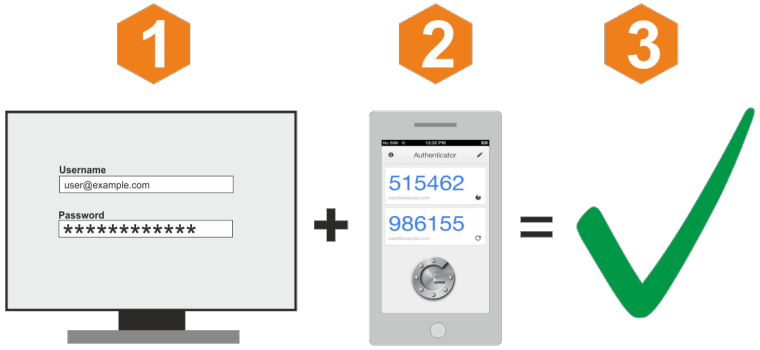
- Secure authentication method of a participant based on knowledge.
- One participant sets a task that the other must solve in order to prove that he knows a certain piece of information without transmitting this information himself.
- Example: Challenge response based on a shared private key
  - sender sends a random message to a receiver (challenge).
  - the receiver encrypts it with a secret key and sends it back (Response)
  - sender checks the result with its own encrypted version
  - use case: Authentication of smart cards against workplace computers




Source: https://atmelcorporation.wordpress.com/2013/04/01/random-challenge-response-authentication-in-plain-english/

# Which authentication method do I use?

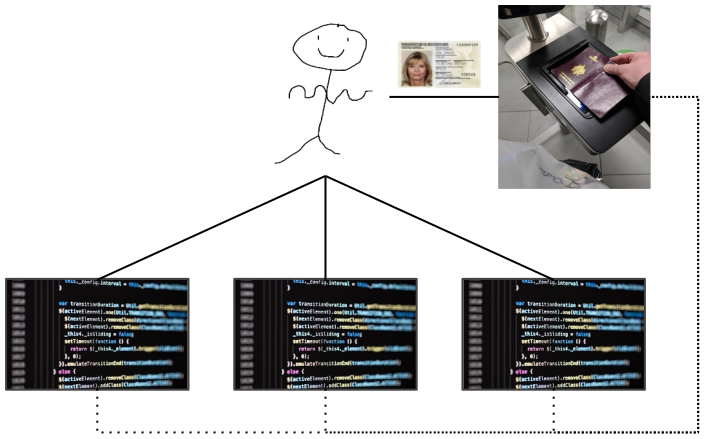
- Today, **Two-Factor Authentication** (**2FA**) is increasingly required
  - Main reason: Password method is too weak for today's computer power and password cracking techniques
  - Second factor is often a **One Time Token (OTT)** which is transmitted over another channel (**Two Channel Authentication**)
  - Security features of the second factor: unique, valid for a limited period of time
  - Examples: RSA Secure ID, Yubikey, TAN, SMS, special mobile apps (e.g. Google Authenticator)




Source: https://docs.opnsense.org/manual/two_factor.html

Source: https://www.yubico.com/de/product/yubikey-5-nfc/

# Single Sign On (SSO)



- After a one-time authentication, a user can access all computers and services for which he is authorized without having to log in each time.

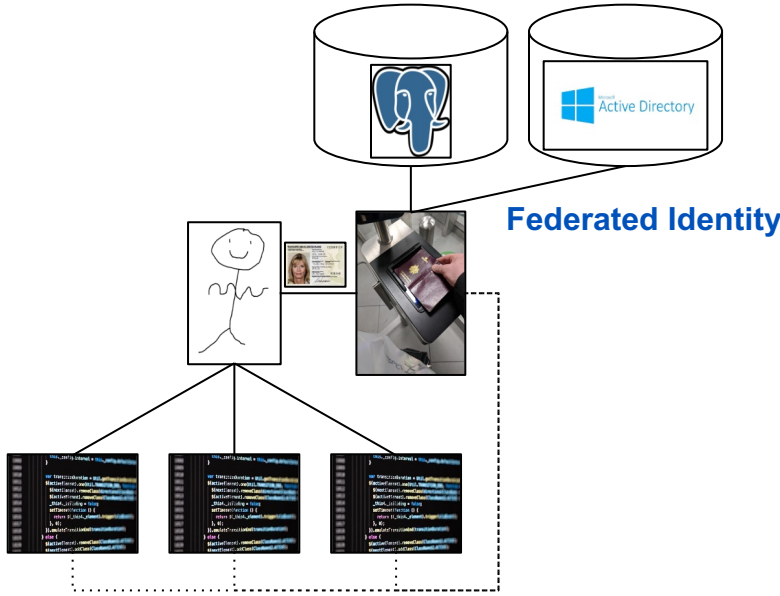
- After successful authentication, the user receives a digital token that is used as a digital ID for applications.

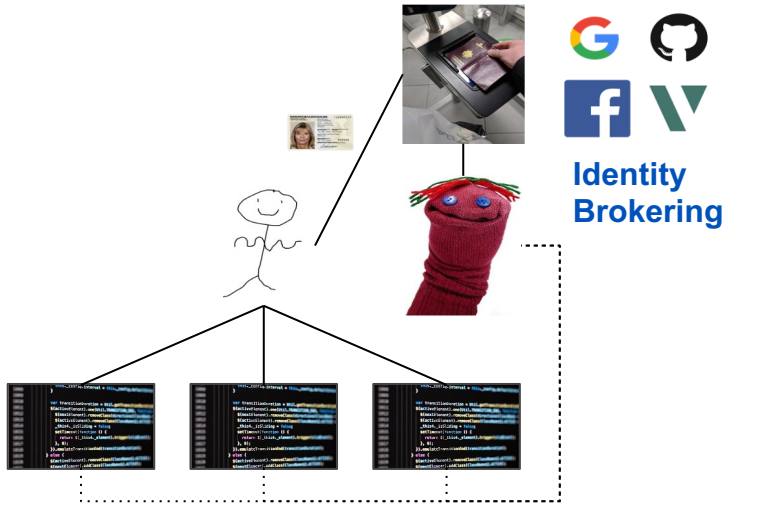
- Advantages
  - user only must log in once
  - password only must be transmitted once, user only needs one password
  - withdrawal/blocking of a user is possible at a central location

- Disadvantages
  - SSO server is weak point of the system
  - availability: in case of failure all services are blocked
  - if SSO identity is stolen, access to many systems is possible

# Other aspects of Single Sign On

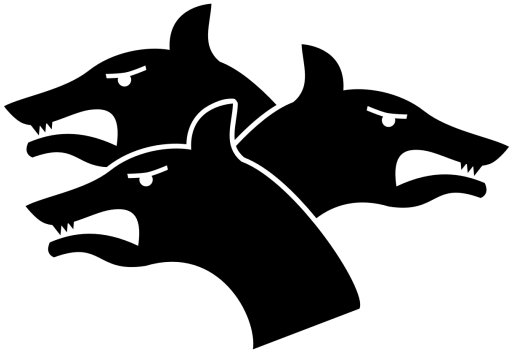
- **Federated SSO**: across company boundaries
- **Single Sign Out**: users are logged out of all other SSO services by logging out once.
- **Identity brokering**: trust between providers enables access by externally authenticated users
- Examples for SSO
  - Google (portal solution for various Google services)
  - Sibboleth (open source SSO based on SAML),
  - Windows Azure Active Directory
  - Kerberos
  - OpenID
  - SAML (Security Assertion Markup Language)




Source of the illustrations: QAware presentation by Christian Fritz, Andreas Zitzelsberger

# Kerberos Authentication System

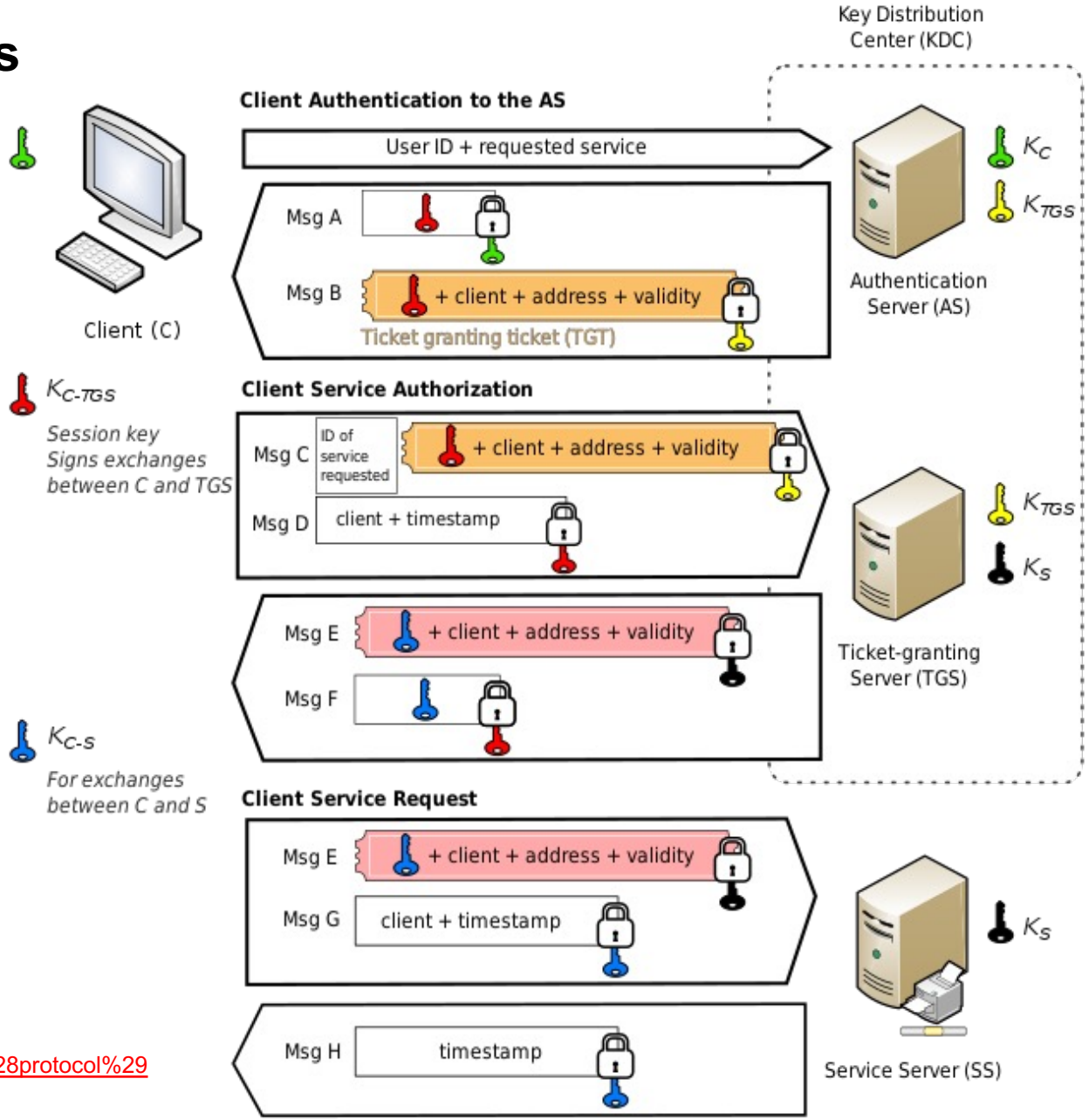


- Authentication protocol used with Windows
- Kerberos enables Single Sign On (SSO)
- No transmission of user passwords,
  no use of asymmetric procedures,
  uses symmetric procedures
- The authentication service is provided by a trusted server **Authentication Server (AS)**
- A **Ticket Granting Server (TGS)** grants tickets for access to services
- Problem: AS and TGS must always be available online
- Two tasks:
  - authenticate subjects (AS)
  - exchange session keys (TGS)

# Kerberos: How it works

## Three-steps:

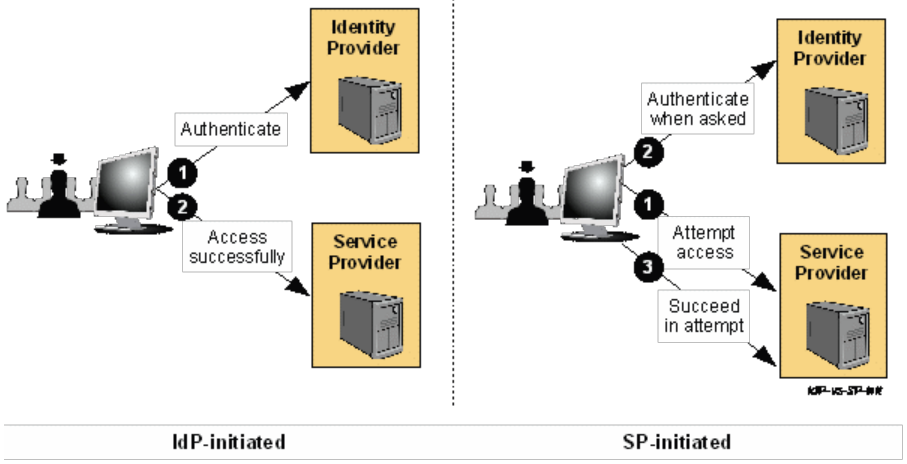
1) Message exchange with AS (Kerberos Server)
   →**key for TGS**
   once at login
2) Communication with Ticket Granting Server (TGS)
   → **tickets on demand**
3) Communicate with the service



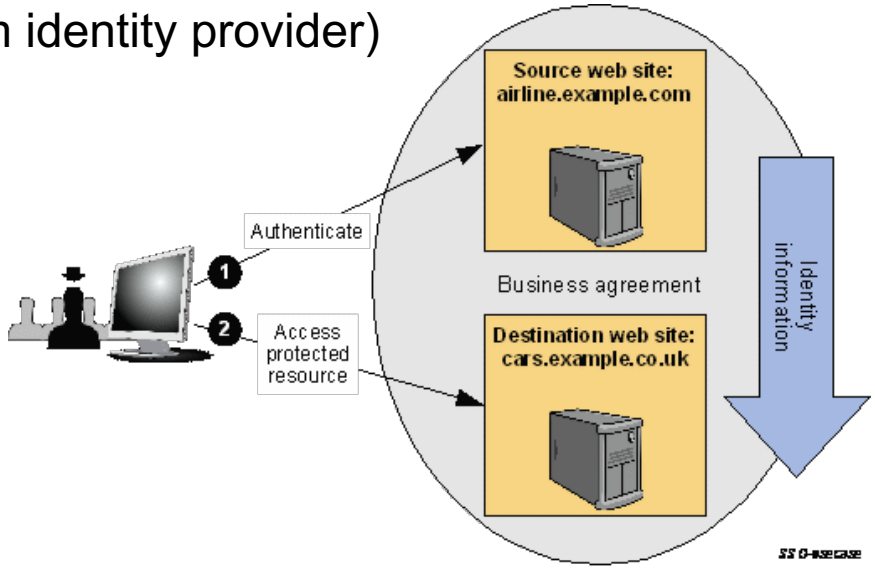

Source: https://en.wikipedia.org/wiki/Kerberos_%28protocol%29

# Security Assertion Markup Language (SAML)

- **SAML 2.0** is an XML framework for the exchange of authentication and authorization information (OASIS standard)
- Drivers of SAML: SSO, Federated Identity, Web services (SaaS)
- Two main types of SAML providers
  - identity provider (verifies end user)
  - service provider (requires authentication from identity provider)




*Differences in Initiation of Web Browser SSO*




*General Single Sign-On Use Case*

Quelle: https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html

# SAML Assertions

Quelle: https://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html

```
 1: <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
 2:   Version="2.0"
 3:   IssueInstant="2005-01-31T12:00:00Z">
 4:   <saml:Issuer Format=urn:oasis:names:SAML:2.0:nameid-format:entity>
 5:     http://idp.example.org
 6:   </saml:Issuer>
 7:   <saml:Subject>
 8:     <saml:NameID
 9:       Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
10:         j.doe@example.com
11:     </saml:NameID>
12:   </saml:Subject>
13:   <saml:Conditions
14:     NotBefore="2005-01-31T12:00:00Z"
15:     NotOnOrAfter="2005-01-31T12:10:00Z">
16:   </saml:Conditions>
17:   <saml:AuthnStatement
18:     AuthnInstant="2005-01-31T12:00:00Z" SessionIndex="67775277772">
19:     <saml:AuthnContext>
20:       <saml:AuthnContextClassRef>
21:         urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
22:       </saml:AuthnContextClassRef>
23:     </saml:AuthnContext>
24:   </saml:AuthnStatement>
25: </saml:Assertion>
```

*Figure 6: Assertion with Subject, Conditions, and Authentication Statement*

SAML assertions are statements about

**properties** (identity, attributes)

and **permissions** of a user

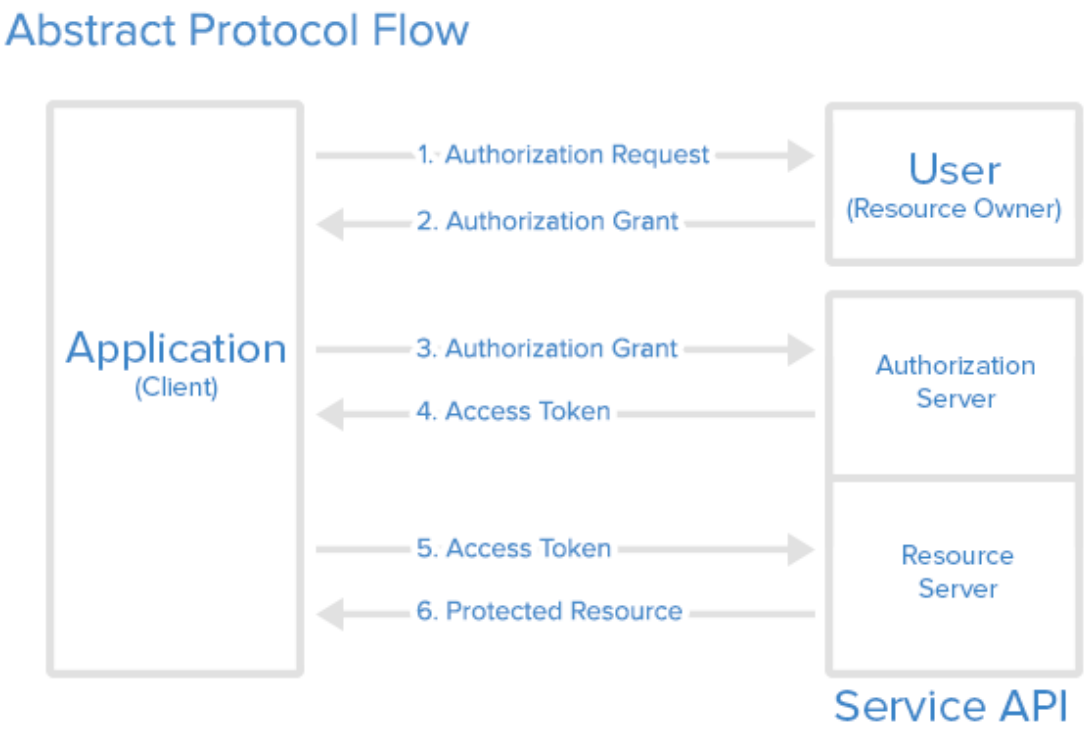
```
 1: <saml:AttributeStatement>
 2:   <saml:Attribute
 3:      xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
 4:      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
 5:      Name="urn:oid:2.5.4.42"
 6:      FriendlyName="givenName">
 7:      <saml:AttributeValue xsi:type="xs:string"
 8:        x500:Encoding="LDAP">John</saml:AttributeValue>
 9:   </saml:Attribute>
10:   <saml:Attribute
11:     NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
12:     Name="LastName">
13:     <saml:AttributeValue
14:       xsi:type="xs:string">Doe</saml:AttributeValue>
15:   </saml:Attribute>
16:   <saml:Attribute
17:     NameFormat="http://smithco.com/attr-formats"
18:     Name="CreditLimit">
19:     xmlns:smithco="http://www.smithco.com/smithco-schema.xsd"
20:     <saml:AttributeValue xsi:type="smithco:type">
21:       <smithco:amount currency="USD">500.00</smithco:amount>
22:     </saml:AttributeValue>
23:   </saml:Attribute>
24: </saml:AttributeStatement>
```

*Figure 7: Attribute Statement*

# OAuth (Open Authorization)


- **OAuth 2.0** is a standard for the authorizing of API accesses on the web
- OAuth 2.0 enables delegation of authorization.
- An **access token** contains authorization for a client (e.g., customer) and a set of permissions.
- A **refresh token** can be used to obtain a new access token
- OAuth 2.0 is very well suited for server-side applications (e.g. cloud native services) where the access tokens can be stored securely
  - owner grants one-time permission to invoke a service
  - server stores access and refresh tokens

# OAuth 2.0 authorization protocol flow




Source: https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

Answer of authorization server
with **access token** and **refresh token**

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"example",
  "expires_in":3600,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
  "example_parameter":"example_value"
}
```

https://tools.ietf.org/html/rfc6749#section-1.5

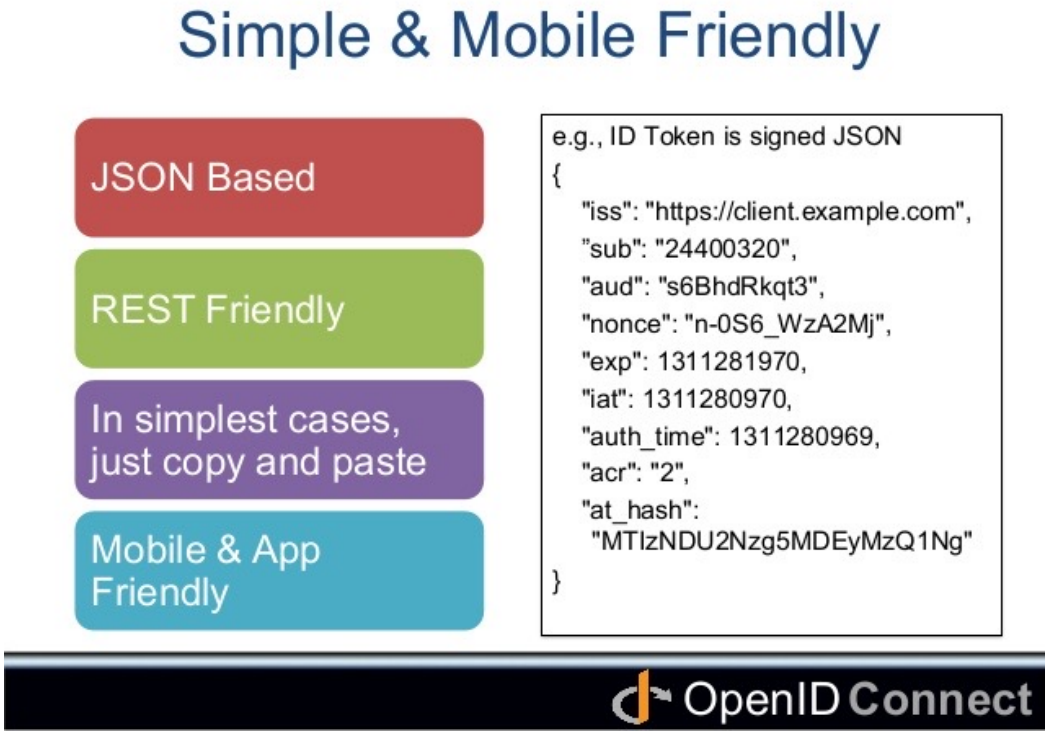
Authorization Grant = permission from owner to access resource

Access Token= token to access resource

# SSO in the Web



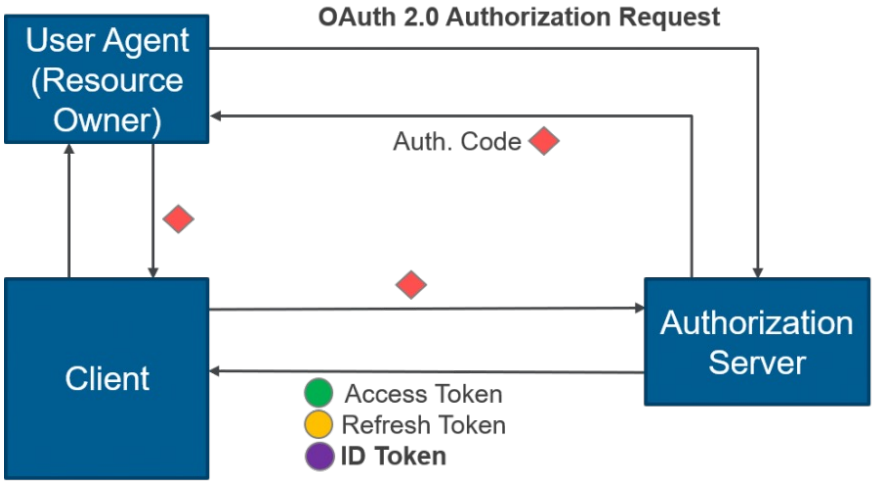
- **OpenID Connect** extends OAuth with all necessary functions for login and SSO
- OpenID supports decentralization: there are many OpenID providers, switching between them is possible
- Authentication information is stored in a signed **JSON Web Token JWT**
  - Claims (UserID, email, ...)
  - Metadata
  - Optional Encryption




Quelle:
http://de.slideshare.net/nat_sakimura/introduction-to-openid-connect/30

# The interaction of OAuth, OpenID Connect and JWT

- Resource Owner gives Client the **Authorization Code** via an OAuth Request
- Client fetches the tokens to access the resource with Authorization Code
- **Access Token**: allows access to the resource and the userinfo
- **Refresh Token**: is used to renew an expired Access Token
- **ID Token** and userinfo give access to the resource and are in the format JWT




```
{
  "userinfo":
   {
    "given_name": {"essential": true},
    "nickname": null,
    "email": {"essential": true},
    "email_verified": {"essential": true},
    "picture": null,
    "http://example.info/claims/groups": null
   },
  "id_token":
   {
    "auth_time": {"essential": true},
    "acr": {"values": ["urn:mace:incommon:iap:silver"] }
   }
}
```

`/userinfo` soll diese Claims zusätzlich zum Scope liefern

`essential`: unverzichtbar für korrekte Funktion

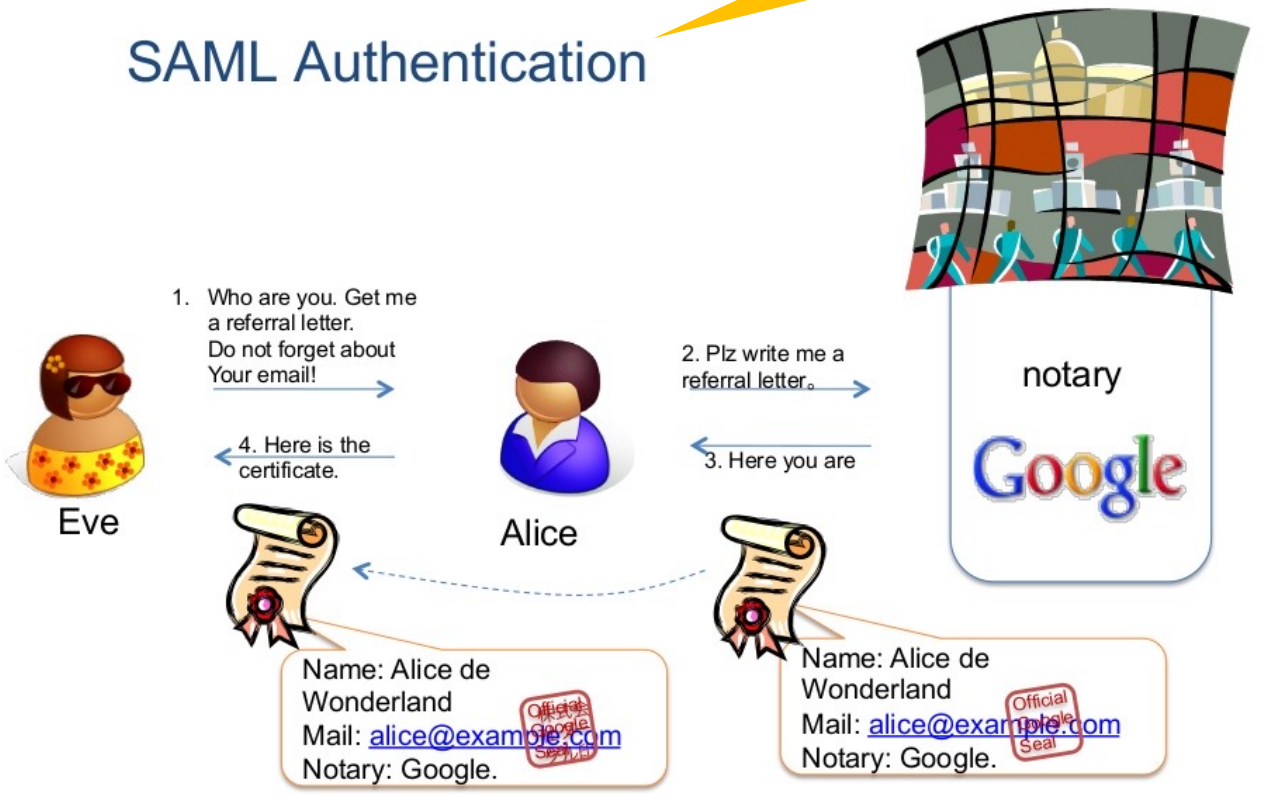
ID Token soll diese Claims zusätzlich zum Scope liefern

`values`: Werte sollen aus angegebenem Array stammen

Quelle https://www.oose.de/blogpost/oauth-openid-connect-und-jwt-wie-haengt-das-alles-zusammen-teil-2/

# Authentication with certificate

SAML supports authentication and authorization

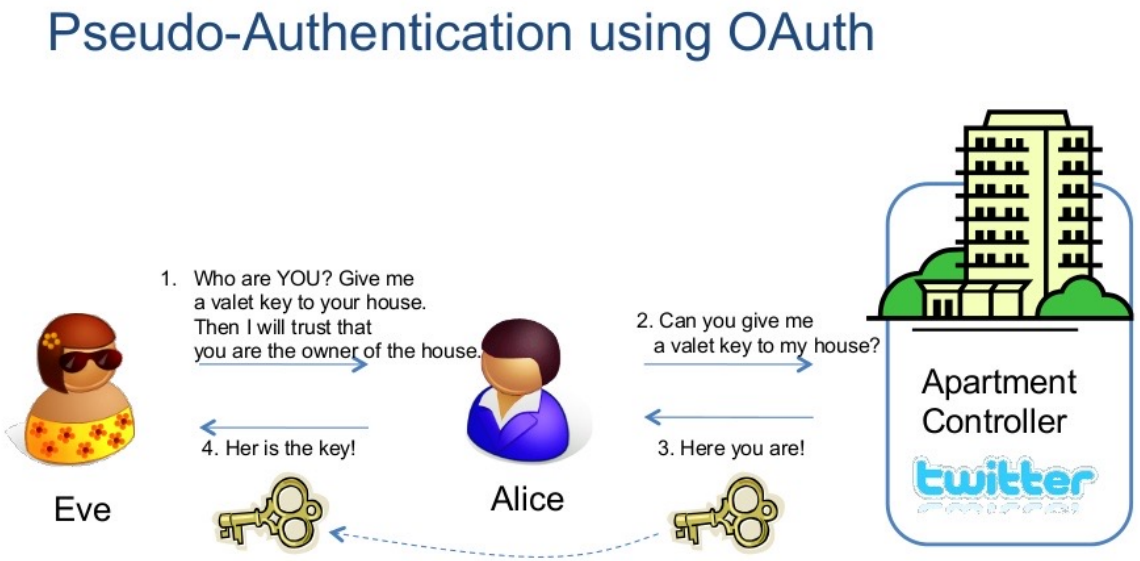



OpenID Connect

Quelle: http://de.slideshare.net/nat_sakimura/introduction-to-openid-connect/37

# Authentication with access token

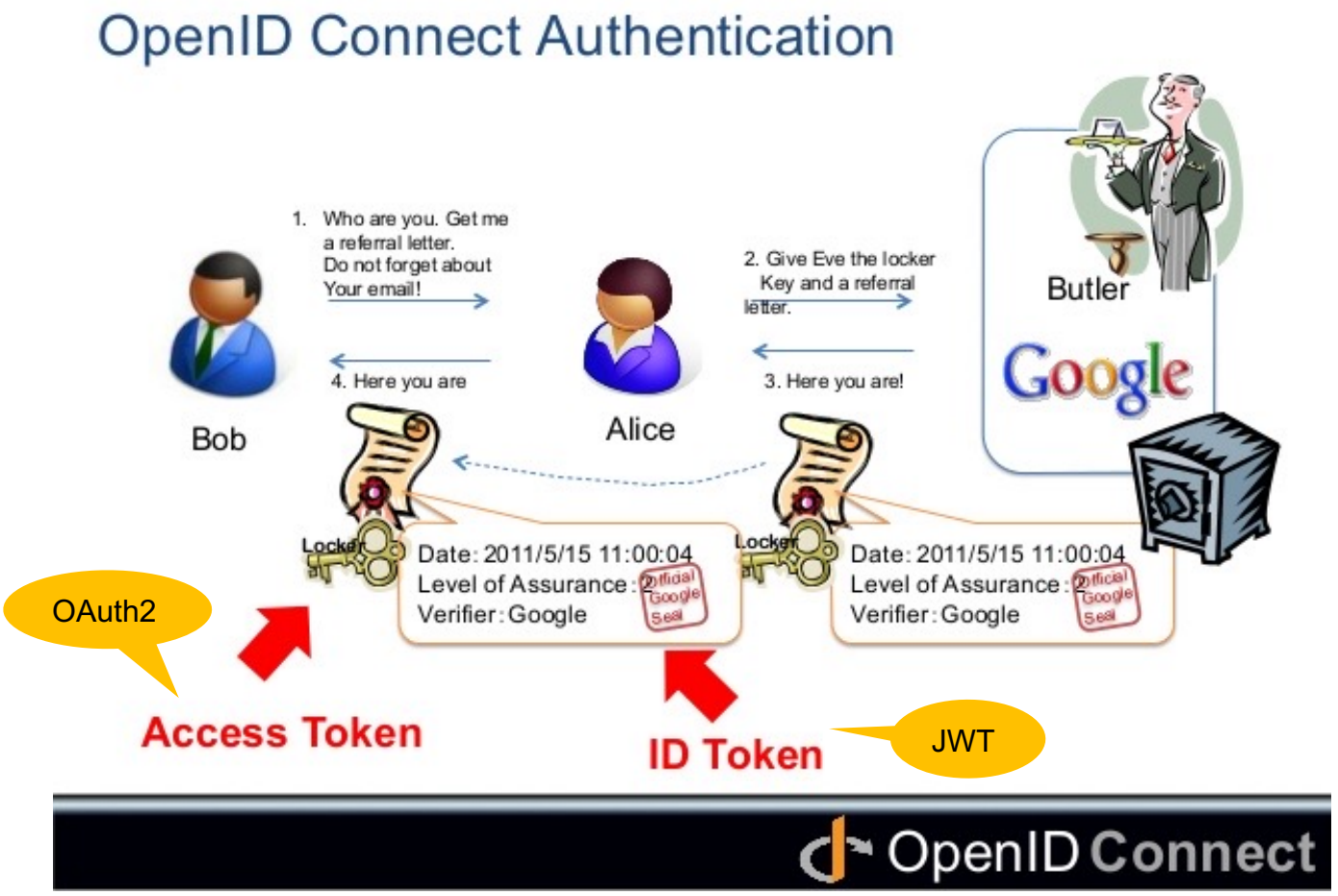
OAuth supports autorization




OpenID Connect

Quelle: http://de.slideshare.net/nat_sakimura/introduction-to-openid-connect/37

# The interaction between OpenID and OAuth




OpenID supports authentication


OAuth supports authorization

Quelle: http://de.slideshare.net/nat_sakimura/introduction-to-openid-connect/37

Video: https://www.youtube.com/watch?feature=player_embedded&v=Kb56GzQ2pSk