

IT-Sicherheit



Kapitel 4: Authentifizierung und Autorisierung

- ▶ Zugangskontrolle
- ▶ Authentifizierungsverfahren
- ▶ Praktische Aspekte bei der Authentifizierung
- ▶ Zugriffskontrolle
- ▶ Verfahren der Zugriffskontrolle





Worum geht es?



- ▶ Was ist der Unterschied zwischen Authentifizierung und Autorisierung?
- ▶ Welche Varianten gibt es für die Authentifizierung?
- ▶ Welche Varianten gibt es für die Autorisierung?
- ▶ Was muss bei der praktischen Umsetzung alles beachtet werden?



Identification, Authentication and Authorization

Diese Unterscheidung ist wichtig!



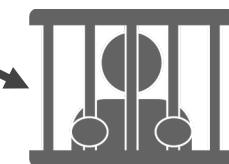
Identification
provide identity



Authentication
assert identity



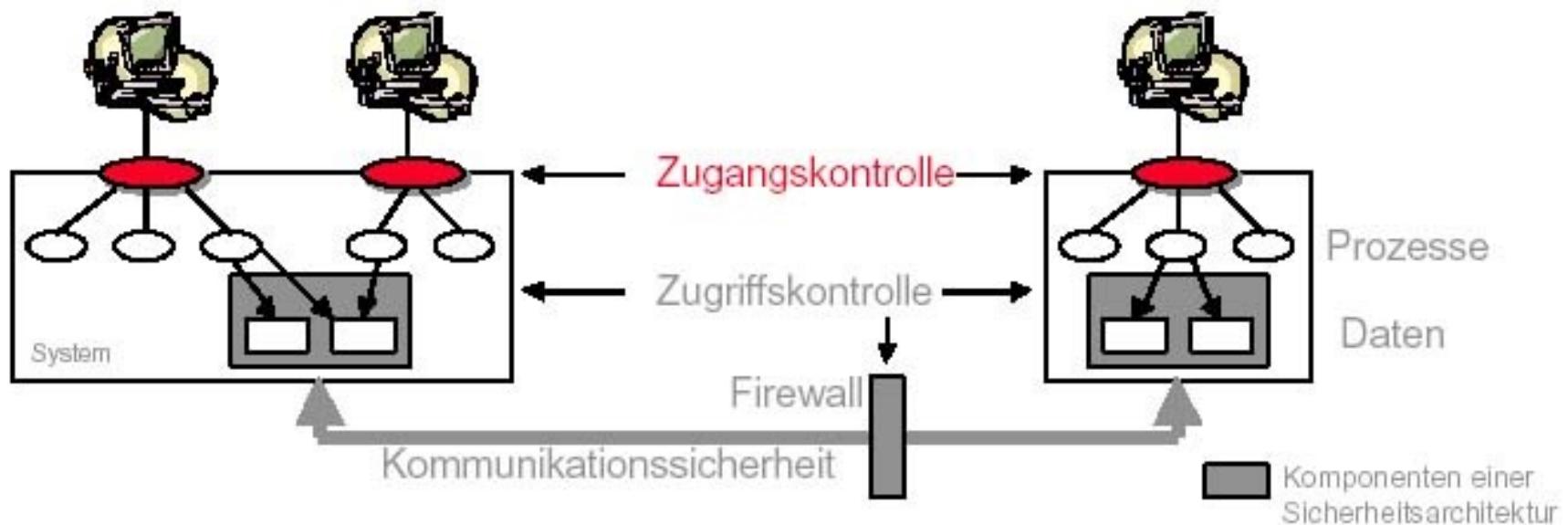
Authentification
validate the assertion



Authorization
execute an access policy



Definitionen für Zugangskontrolle und Zugriffskontrolle



▶ Authentifizierung / Authentisierung:

Bei der Anmeldung an einem System wird im Rahmen der Authentifizierung die Identität der Person, die sich anmeldet, geprüft und verifiziert. Der Begriff wird auch verwendet, wenn die Identität von IT-Komponenten oder Anwendungen geprüft wird.

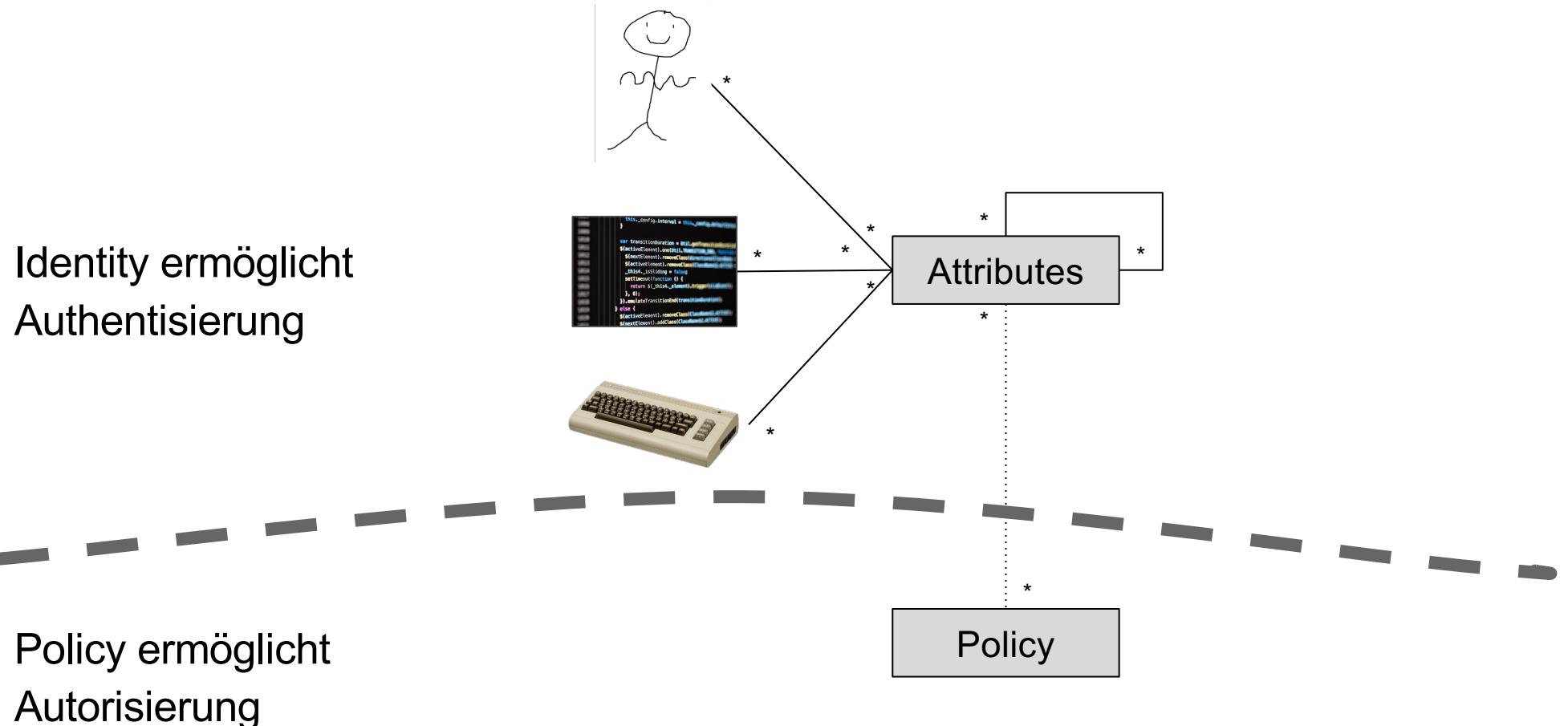
▶ Autorisierung:

Bei einer Autorisierung wird geprüft, ob eine Person, IT-Komponente oder Anwendung zur Durchführung einer bestimmten Aktion berechtigt ist.



Unterscheide zwischen Identität und Policy

Identity ermöglicht
Authentisierung



Policy ermöglicht
Autorsierung

Quelle: QAware Vortrag, Christian Fritz, Andreas Zitzelsberger



Zugangskontrolle - Authentication

- ▶ Authentisierung ist möglich durch
 - ▶ Wissen
 - ▶ Besitz
 - ▶ Persönliches Merkmal
- ▶ Passwort-Verfahren
- ▶ Biometrie
- ▶ Challenge-Response-Verfahren
- ▶ Zertifikate / Signaturen
 - ▶ Basierend auf einer PKI





Passwortverfahren

- ▶ Weit verbreitet, einfach implementierbar, mobil
- ▶ Probleme
 - ▶ Passwort-Cracking
 - ▶ Falls jemand Passwort weiß hat er uneingeschränkten Zugang
 - ▶ Vergessen von Passwörtern → Sichere Recovery Prozedur erforderlich
 - ▶ Unverschlüsselte Übertragung über Netzwerke
- ▶ Maßnahmen zur Erhöhung der Sicherheit
 - ▶ Mindestlänge, Sonderzeichen, regelmäßige Änderungen???
 - ▶ Sperrung/Verzögerung der Kennung nach geringer Anzahl von Fehlversuchen
 - ▶ Anzeige letzter Login
 - ▶ Abspeicherung als Hash-Wert
 - ▶ Verwendung eines Salts (zufällig gewählte Bitfolge) bei der Berechnung des Hashwerts
- ▶ Zusatzmaßnahmen
 - ▶ Einmalpasswörter, TAN (Transaktionsnummern), z.B. Online Banking
 - ▶ NONCE, Captcha, Sicherheitsfrage, Zeitstempel





Biometrische Techniken

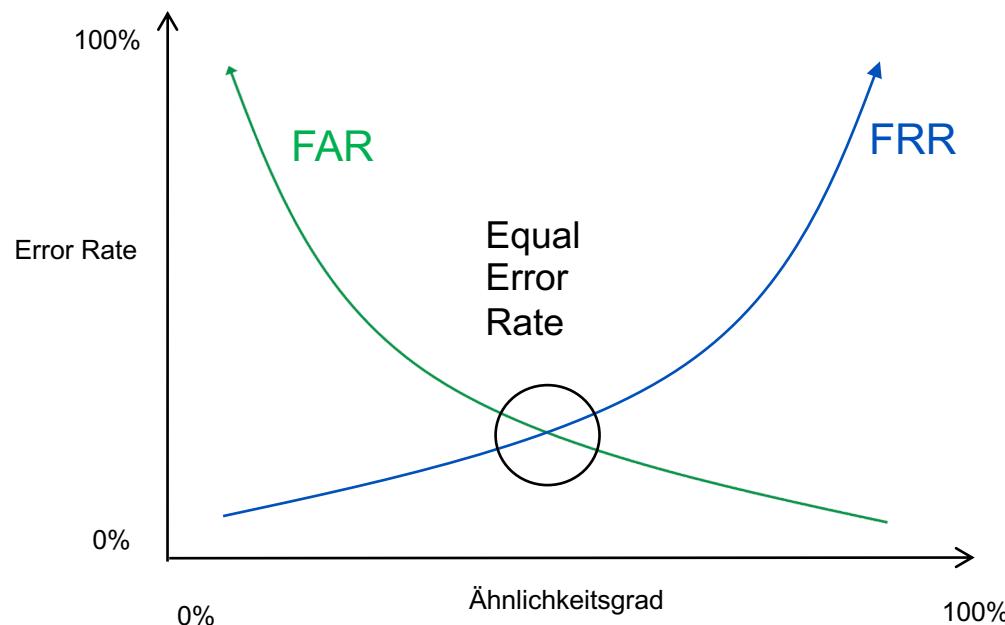


- ▶ Fingerabdruck, Gesichtserkennung, Handerkennung, Iris, Retina, Tippverhalten, Stimme, Unterschrifterkennung
- ▶ Ermöglichen eine eindeutige Identifikation von Personen
- ▶ Probleme
 - ▶ Können gefälscht, abgefangen, gewaltsam missbraucht werden
 - ▶ Einspielen von Daten unter Umgehung des biometrischen Sensors
 - ▶ Lassen sich teilweise nicht austauschen
 - ▶ Unbemerkte Erfassung und flächendeckende Überwachung möglich
 - ▶ Sicherheitsproblem: Referenzdatenbank
 - ▶ Täuschen 100% Sicherheit vor



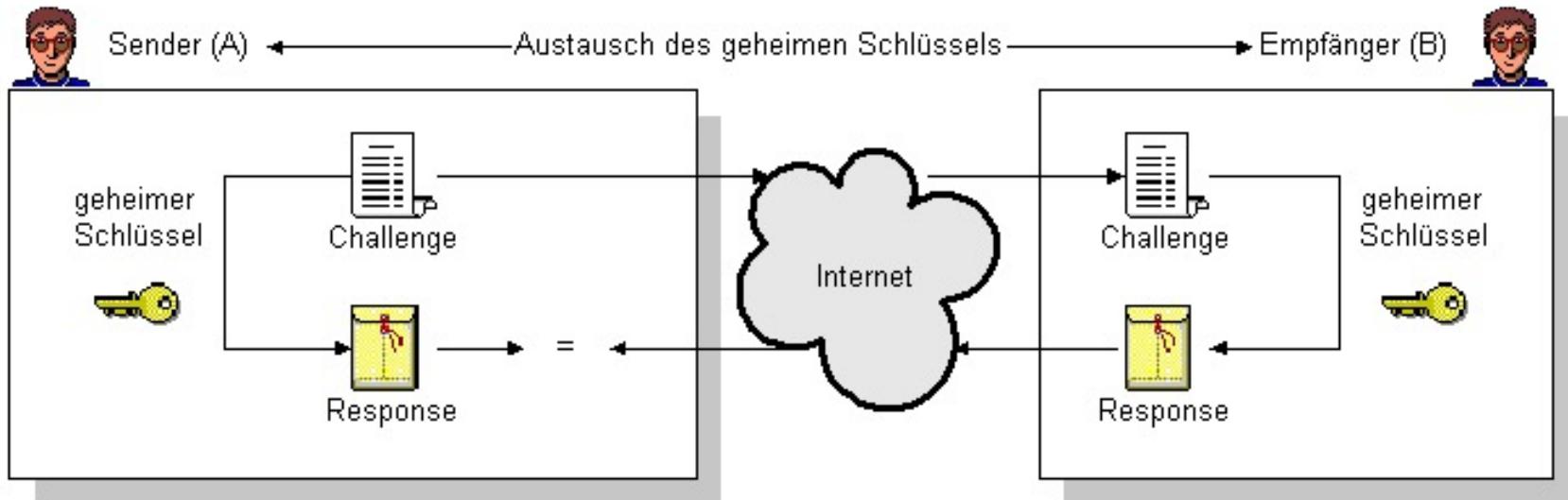
Probleme bei Verifikation von Biometrischer Authentifizierung

- ▶ Fehlakzeptanzrate (**False Acceptance Rate FAR**)
 - ▶ Fälschliche Akzeptanz einer unberechtigten Person (false positiv)
- ▶ Fehlerabweisungsrate (**False Rejection Rate FRR**)
 - ▶ Fälschliche Rückweisung einer berechtigten Person (false negative)





Challenge Response



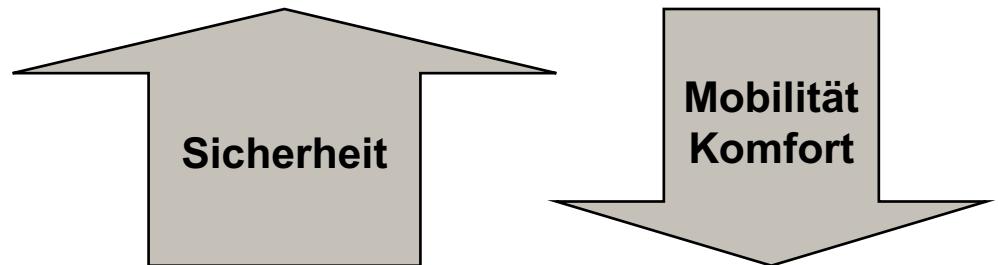
- ▶ Sender sendet eine zufällige Nachricht an einen Empfänger (Challenge)
- ▶ Dieser verschlüsselt sie mit einem geheimen Schlüssel und sendet sie zurück (Response)
- ▶ Sender überprüft das Ergebnis mit einer eigenen verschlüsselten Version
- ▶ Anwendung: Authentifikation von Chipkarten gegenüber Arbeitsplatzrechner



Welches Authentifizierungsverfahren setze ich ein?

- ▶ Die Erhöhung der Sicherheit führt meist zu Verlust des Komfort und der Mobilität für den User

- ▶ Smart Card / Biometrie / HW-Token
- ▶ Software-Zertifikate
- ▶ Zugriffs-Token (TAN)
- ▶ Benutzername + Passwort

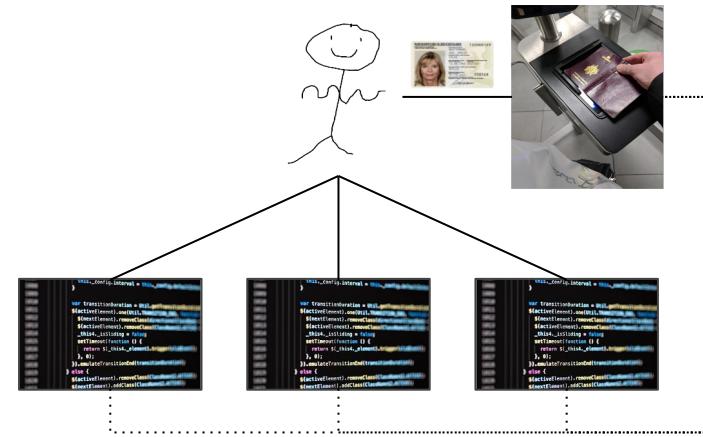


- ▶ Heute wird immer häufiger eine **Zweifaktor-Authentifizierung** gefordert
 - ▶ Hauptgrund: Passwortverfahren ist zu schwach
 - ▶ Zweiter Faktor ist häufig ein **One Time Token (OTT)** der über einen anderen Kanal (**Two Channel Authentication**) übertragen wird
 - ▶ Sicherheitsmerkmale vom zweiten Faktor: einmalig, zeitlich beschränkt gültig
 - ▶ Bsp. RSA Secure ID, TAN, SMS, spezielle mobile Apps (z.B. Google Authenticator)



Single Sign On (SSO)

- ▶ Ein Benutzer kann nach einer einmaligen Authentifizierung auf alle Rechner und Dienste, für die er berechtigt ist, zugreifen, ohne sich jedes Mal neu anmelden zu müssen



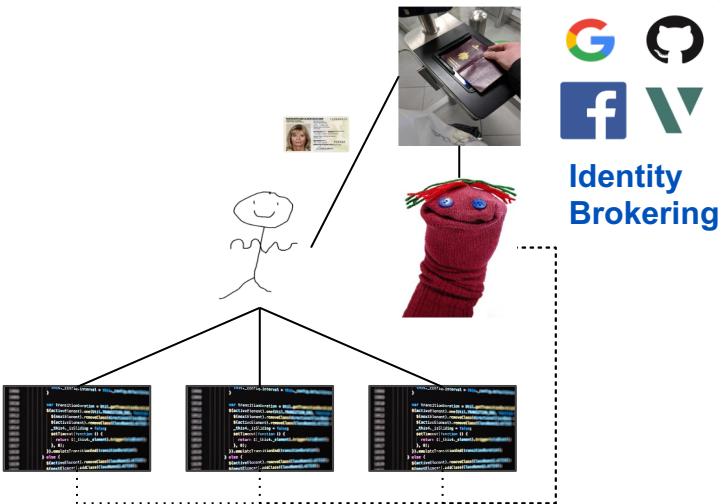
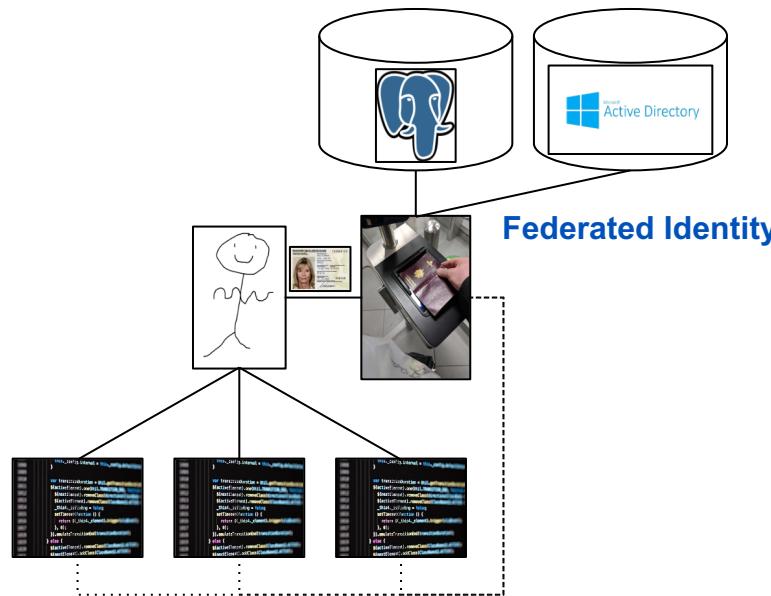
Quelle: QAware Vortrag, Christian Fritz, Andreas Zitzelsberger

- ▶ Nach erfolgreicher Authentifizierung erhält User einen digitalen Token der als digitaler Ausweis für Anwendungen verwendet wird
- ▶ Vorteile
 - ▶ Benutzer muss sich nur einmal anmelden
 - ▶ Passwort muss nur ein einziges mal übertragen werden, Benutzer benötigt nur ein Passwort
 - ▶ Rechteentzug/Sperrung eines Benutzers ist an einer zentralen Stelle möglich
- ▶ Nachteile
 - ▶ SSO Server ist Schwachstelle des Systems
 - ▶ Verfügbarkeit: bei Ausfall sind alle Services gesperrt
 - ▶ Bei Diebstahl der SSO Identität ist Zugriff auf viele Systeme möglich



Weitere Aspekte zu Single Sign On

- ▶ **Federated SSO:** Über Unternehmensgrenzen hinweg
- ▶ **Single Sign Out:** Anwender wird durch einmaliges Abmeldung bei allen weiteren SSO-Diensten abgemeldet
- ▶ **Identity Brokering:** Vertrauen zwischen Anbieter ermöglicht den Zugriff von extern authentifizierten Benutzern
- ▶ Beispiele für SSO
 - ▶ Google (Portallösung für verschiedene Google Dienste)
 - ▶ Sibboleth (Open Source SSO basierend auf SAML),
 - ▶ Windows Azure Active Directory
 - ▶ OpenID
 - ▶ OAuth
 - ▶ SAML (Security Assertion Markup Language)

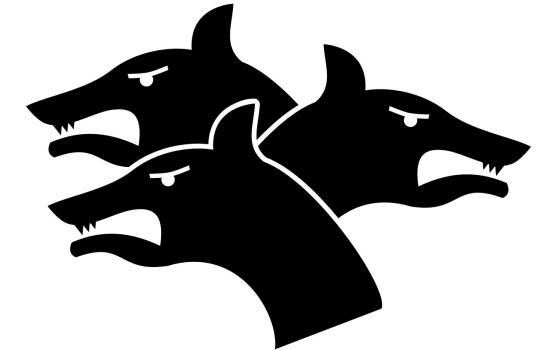


Quelle der Bilder: QAware Vortrag, Christian Fritz, Andreas Zitzelsberger



Kerberos Authentifizierungssystem

- ▶ Authentifizierungsprotokoll das bei Windows eingesetzt wird
- ▶ Kerberos ermöglicht Single Sign On (SSO)
- ▶ Der Authentifikations-Dienst wird von einen vertrauenswürdigen Server erbracht (**Kerberos Server KS**)
- ▶ Ein **Ticket Granting Server** (TGS) vergibt Tickets für Zugriff auf Dienste
- ▶ Keine Übertragung von Benutzerpasswörtern, Verzicht auf asymmetrische Verfahren, verwendet symmetrische Verfahren
- ▶ Problem: KS und TGS müssen immer online erreichbar sein
- ▶ Zwei Aufgaben:
 - ▶ Subjekte authentifizieren (KS)
 - ▶ Sitzungsschlüssel austauschen (TGS)

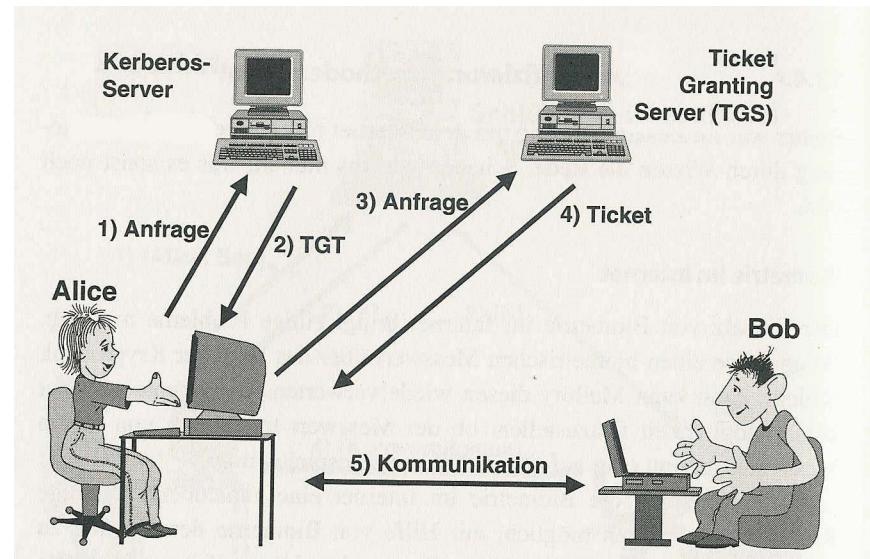




Kerberos Funktionsweise

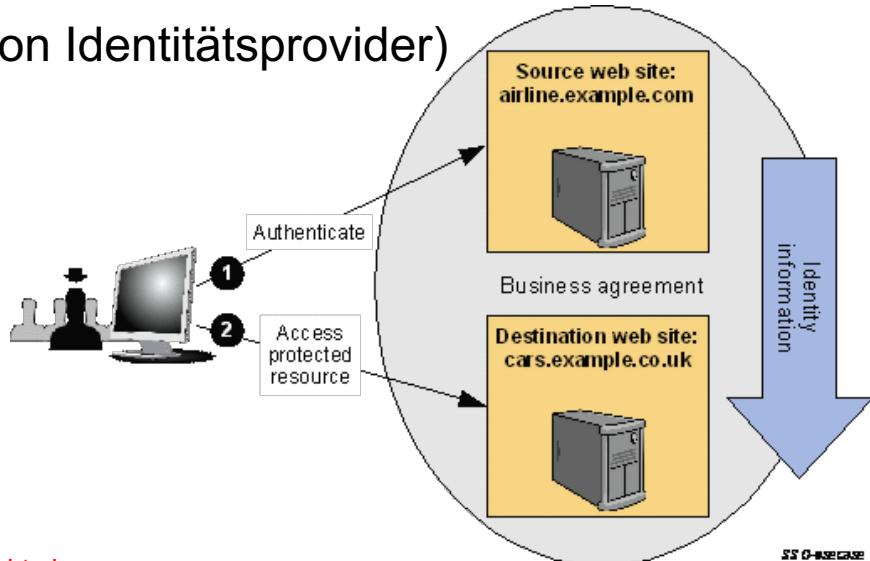
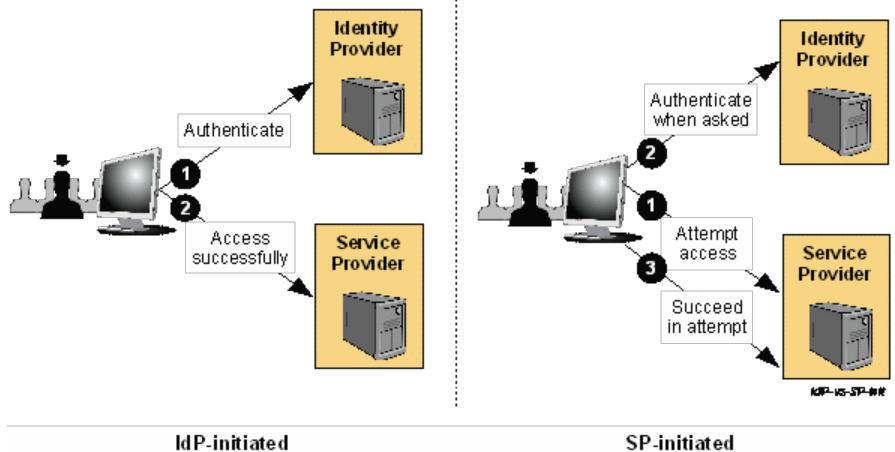
- 0) Alice und Bob vereinbaren gemeinsamen Schlüssel (Key-Alice, Key-Bob) mit Kerberos Server
- 1) Initialisierungsnachricht mit Angabe von TGS
- 2) KS generiert **Schlüssel K1**,
verschlüsselt ihn mit Key von Alice und sendet ihn an Alice,
KS bildet $TGT = EncKey-TGS(\text{Name Alice}, K1, \text{weitere Angaben})$ und verschlüsselte es
- 3) Alice sendet TGT (Ticket Granting Ticket)unverändert an TGS
- 4) TGS generiert **Schlüssel K2**,
bildet Ticket: $EncKey-Bob(\text{Name Alice}, K2, \text{weitere Angaben})$
sendet Ticket und $Enc-K1(k2)$ an Alice
- 5) Alice sendet Ticket und
verschlüsselte Nachricht (mit K2) an Bob

Quelle: Klaus Schmeh: Kryptografie, dpunkt.verlag



► Security Assertion Markup Language (SAML)

- ▶ **SAML 2.0** ist ein XML Framework zum Austausch von Authentifizierungs- und Autorisierungsinformationen (OASIS Standard)
- ▶ Einsatz: SSO, SaaS (Software as a Service)
- ▶ Zwei Haupttypen von SAML-Provider
 - ▶ Identitätsprovider (überprüft Endbenutzer)
 - ▶ Dienstanbieter (benötigt Authentifizierung von Identitätsprovider)



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 sind Aussage über Eigenschaften (Identität, Attribute) und Berechtigungen eines Benutzers

```
1: <saml:AttributeStatement>
2:   <saml:Attribute
3:     xmlns:xs= "http://www.w3.org/2001/XMLSchema#"
4:     NameFormat="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
5:     Name="urn:oid:2.5.4.42"
6:     FriendlyName="givenName">
7:     <saml:AttributeValue xsi:type="xs:string"
8:       xs: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)

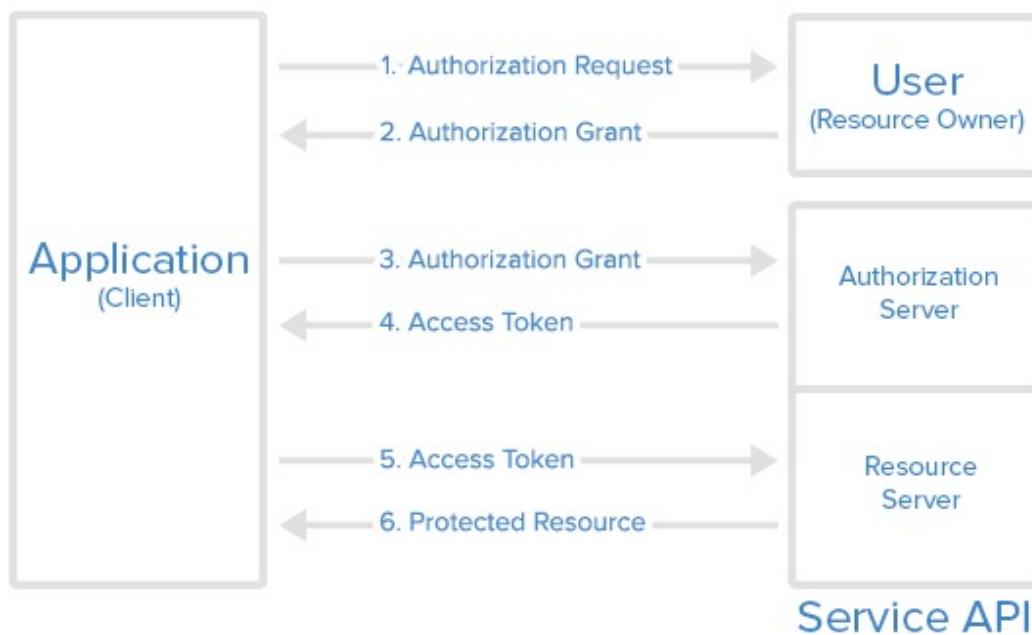


- ▶ OAuth 2.0 ist ein Standard für die Autorisierung von API-Zugriffen im Web
- ▶ OAuth 2.0 ermöglicht Delegation von Autorisierung
- ▶ Ein **Zugriffstoken** enthält die Autorisierung für einen Client (z.B. Kunden) und eine Menge von Berechtigungen.
- ▶ Ein **Refresh-Token** kann verwendet werden, um ein neues Zugriffstoken zu erhalten
- ▶ OAuth 2.0 ist sehr gut geeignet für Serverseitige Applikationen (z.B. Cloud Native Services) in denen die Zugriffstoken sicher gespeichert werden können
 - ▶ Benutzer erteilt einmalig die Genehmigung einen Service aufzurufen
 - ▶ Server speichert Access- und Refresh-Token



OAuth 2.0 Autorisierungs Protokollfluss

Abstract Protocol Flow



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

Antwort von Authorization Server mit **access token** und **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": "tGzv3JOkF0XG5Qx2TIKWIA",
  "example_parameter": "example_value"
}
```

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

Authorization Grant = Zugriffsgenehmigung

Access Token= Zugriffstoken



SSO im Web



- ▶ **OpenID Connect** erweitert OAuth um alle notwendigen Funktionen für Login und SSO
- ▶ Dezentralität:
es gibt viele OpenID-Provider,
ein Wechsel ist möglich
- ▶ Authentifizierungsinformationen
werden in einem signierten
JSON Web Token JWT
gespeichert
 - ▶ Claims (UserID, email, ...)
 - ▶ Metadata
 - ▶ Optional Encryption



Simple & Mobile Friendly

JSON Based

REST Friendly

In simplest cases,
just copy and paste

Mobile & App
Friendly

e.g., ID Token is signed JSON

```
{  
  "iss": "https://client.example.com",  
  "sub": "24400320",  
  "aud": "s6BhdRkqt3",  
  "nonce": "n-0S6_WzA2Mj",  
  "exp": 1311281970,  
  "iat": 1311280970,  
  "auth_time": 1311280969,  
  "acr": "2",  
  "at_hash":  
    "MTIzNDU2Nzg5MDEyMzQ1Ng"  
}
```

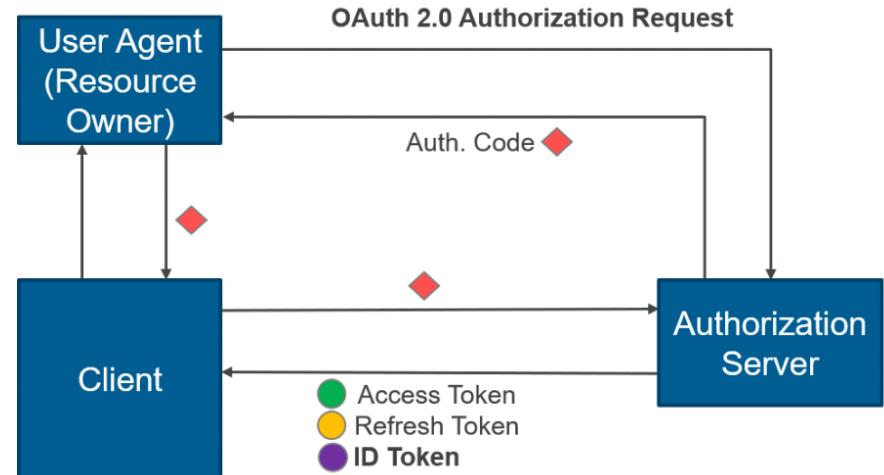
Quelle:

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



Das Zusammenspiel von OAuth, OpenID Connect und JWT

- ▶ Ressource Owner gibt Client den **Authorization Code** über einen OAuth Request
- ▶ Client holt mit Authorization Code die Token zum Zugriff auf die Ressource
- ▶ **Access Token**: ermöglicht Zugriff auf die Ressource und die userinfo
- ▶ **Refresh Token**: dient zur Erneuerung eines abgelaufenen Access Tokens
- ▶ **ID Token** und **userinfo** geben Zugriff auf die Ressource und sind im 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

This block displays a JSON object with two properties: 'userinfo' and 'id_token'. Annotations explain the requirements for these properties: '/userinfo' is noted to provide additional claims beyond the scope, 'essential' is highlighted as crucial for correct function, and 'values' is noted to be derived from an array.

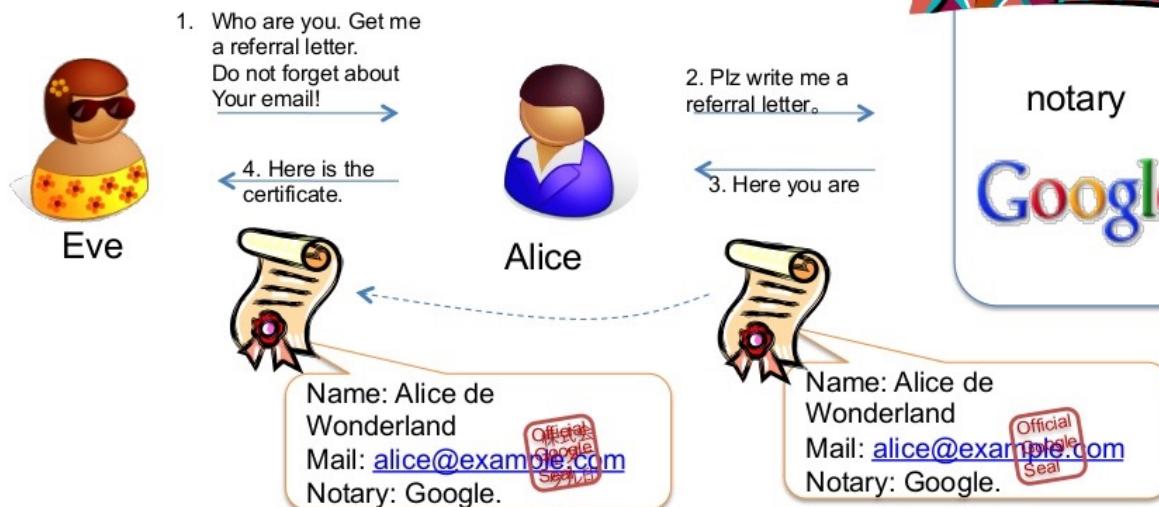
Quelle <https://www.oose.de/blogpost/oauth-openid-connect-und-jwt-wie-haengt-das-alles-zusammen-teil-2/>



Authentifizierung mit Zertifikat

SAML unterstützt Authentifizierung und Autorisierung

SAML Authentication



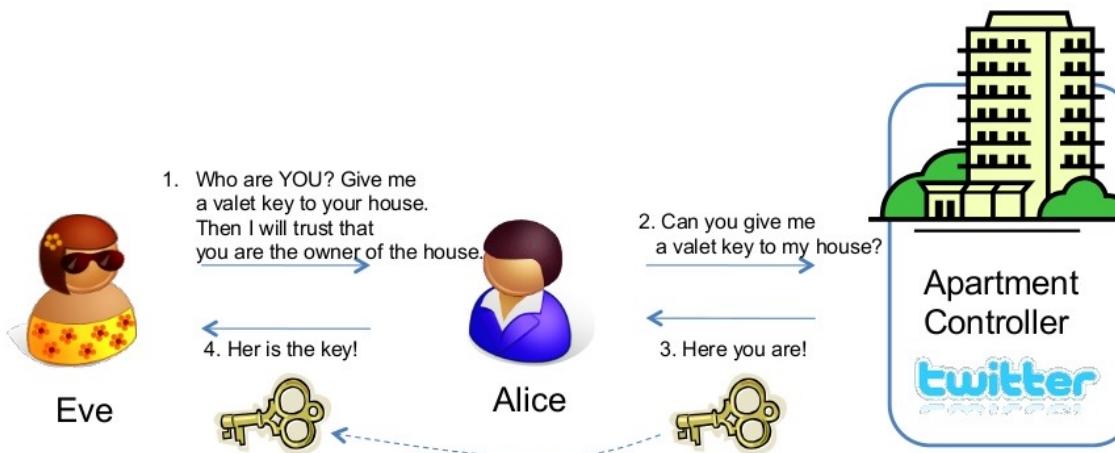
Quelle: http://de.slideshare.net/nat_sakimura/introduction-to-openid-connect/37



Authentifizierung mit Zugriffstoken

OAuth unterstützt Autorisierung

Pseudo-Authentication using OAuth



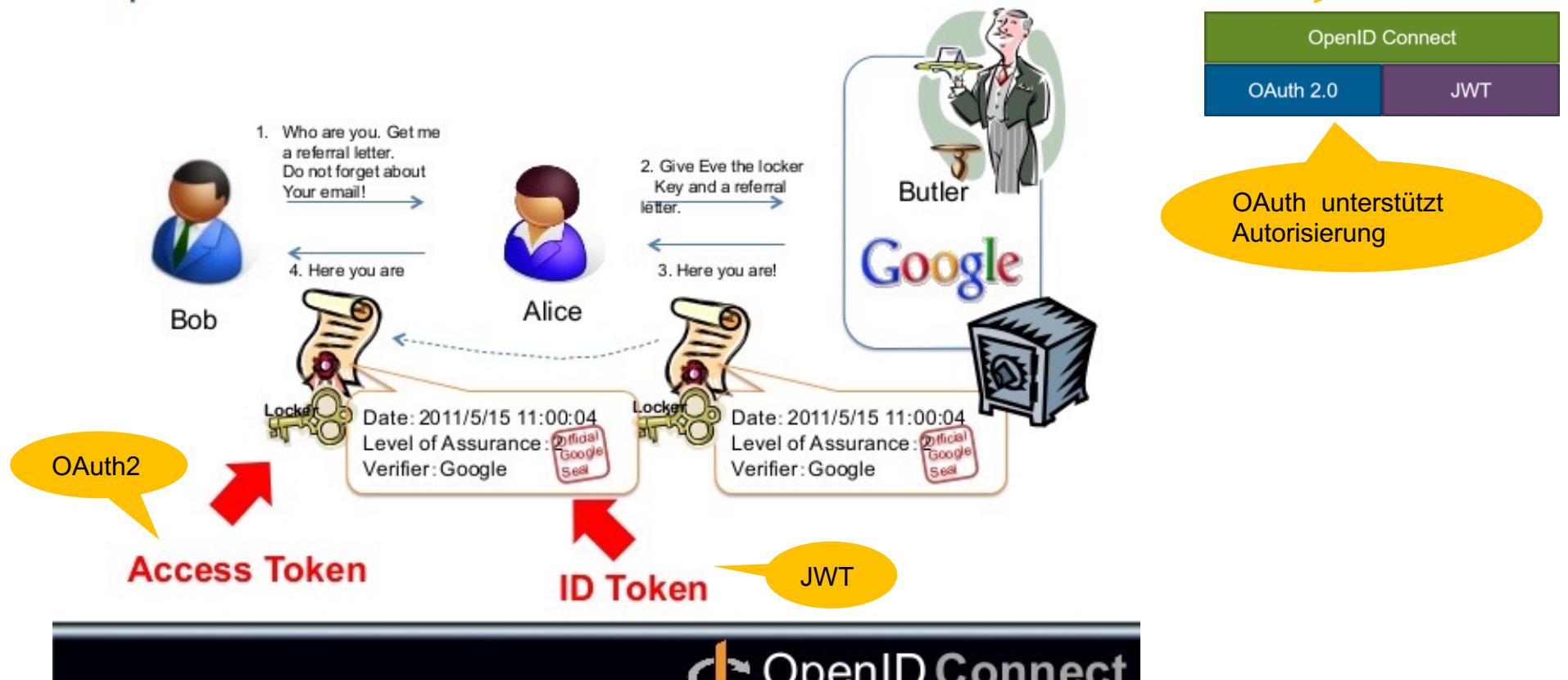
 OpenID Connect

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



Das Zusammenspiel zwischen OpenID und OAuth

OpenID Connect Authentication



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

Video:



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