Cryptonite

SSO:
Single Sign-On
Security
Overview

Áron SZABÓ

H.A.C.K.
Hackerspace Budapest (hsbp.org)
2015-04-10

Single Sign-On protocols

Kerberos + RADIUS/DIAMETER + LDAP

- IETF RFC 4120 – The Kerberos Network Authentication Service (V5)
- IETF RFC 2865 – Remote Authentication Dial In User Service
- IETF RFC 6733 – Diameter Base Protocol
- IETF RFC 4511 – Lightweight Directory Access Protocol

SAML

- OASIS SAML v2.0 – Security Assertion Markup Language
  joint standard of SAML v1.1 + Shibboleth v1.3 + Liberty ID-FF v1.2

OAuth

- IETF RFC 5849 – The OAuth 1.0 Protocol
- IETF RFC 6749 – The OAuth 2.0 Authorization Framework

etc.

OAuth

The OAuth v1.0/v1.0a and OAuth v2.0 are quite different protocols!

November, 2006: Twitter OpenID implementation started
April, 2007: OAuth group was created
July, 2007: Eran Hammer joined standardization group
June, 2009: session fixation attack was fixed
April, 2010: IETF RFC 5849 was published (OAuth v1.0a)
July, 2012: Eran Hammer left standardization group
October, 2012: IETF RFC 6749 was published (OAuth v2.0)

When compared with OAuth 1.0, the 2.0 specification is more complex, less interoperable, less useful, more incomplete, and most importantly, less secure.

/Eran Hammer, http://hueniverse.com/2012/07/26/oauth-2-0-and-the-road-to-hell/

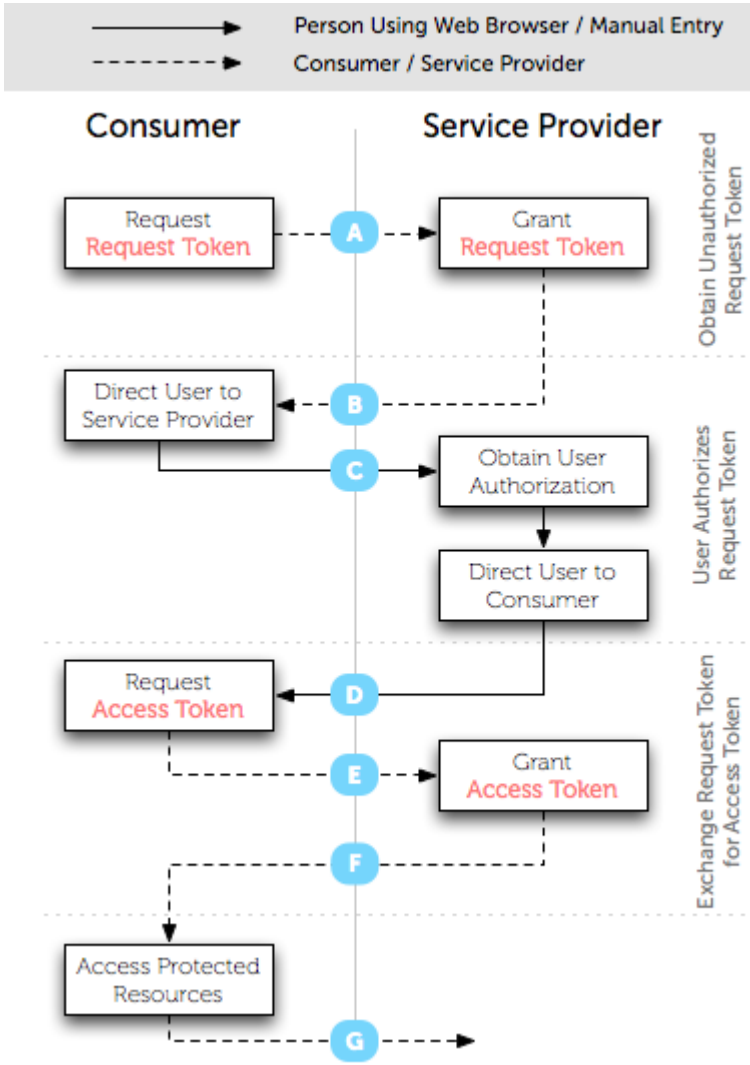
OAuth

Client:	3rd party application, e.g. image editor service
Resource Owner:	human user
Resource:	protected data to be accessed, e.g. image
Resource Server:	database of protected data, e.g. image hosting service
Authorization Server:	performs authentication and authorization

Resource Owner uses a Client application
Resource Owner requests access to a chosen Resource via Client application
Resource is stored at Resource Server
Client redirects Resource Owner to Authorization Server
Resource Owner performs authentication and grants access to Client application
Authorization Server sends a token (ticket) to Client application
Client sends this token (ticket) to Resource Server
Resource Owner gains access to the chosen Resource via Client application

No password or other sensitive user credential was given to Client or Resource Server, just a temporary, authorization ticket!

OAuth v1.0/v1.0a



OAUTH AUTHENTICATION FLOW v1.0a

A Consumer Requests Request Token

Request includes
oauth_consumer_key
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_callback

B Service Provider Grants Request Token

Response includes
oauth_token
oauth_token_secret
oauth_callback_confirmed

C Consumer Directs User to Service Provider

Request includes
oauth_token (optional)

D Service Provider Directs User to Consumer

Request includes
oauth_token
oauth_verifier

E Consumer Requests Access Token

Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_verifier

F Service Provider Grants Access Token

Response includes
oauth_token
oauth_token_secret

G Consumer Accesses Protected Resources

Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)

OAuth v1.0/v1.0a

request: request token (temporary credentials)	http://oauthbin.com/v1/request-token?...
response: request token (temporary credentials)	oauth_token=requestkey&oauth_token_secret=requestsecret
request: authorized request token (authorized temporary credentials)	http://oauthbin.com/v1/access-token?...
response: authorized request token (authorized temporary credentials)	oauth_token=accesskey&oauth_token_secret=accesssecret
request: access token (access data)	http://oauthbin.com/v1/echo?...
response: access token (access data)	data=OAuth 1.0a credentials requested by Aron (using HOTP or TOTP value)

OAuth v1.0/v1.0a

```
request: request token (temporary credentials)
http://oauthbin.com/v1/request-token
     ?data=requested-resource
     /* oauth_consumer_key: equivalent to a username */
     &oauth_consumer_key=key
     /* oauth_signature_method: PLAINTEXT|HMAC-SHA1|RSA-SHA1 */
     &oauth_signature_method=HMAC-SHA1
     /* oauth_signature: base64 and URL encoded signature value */
     &oauth_signature=AXN1dRsvzPxHxl9FNgFDXyJHdLM%3D
     /* oauth_timestamp: UNIX timestamp */
     &oauth_timestamp=1413820413
     /* oauth_nonce: random unique hexadecimal value */
     &oauth_nonce=17cb023cab1c5ca3b7d5eb808154c0c048040593
     /* oauth_version: must be set to "1.0" */
     &oauth_version=1.0
     /* oauth_callback: absolute URI */
     &oauth_callback=

response: request token (temporary credentials)
     /* oauth_token: temporary credentials, identifier */
     oauth_token=requestkey
     /* oauth_token_secret: temporary credentials, shared-secret */
     &oauth_token_secret=requestsecret
     /* oauth_callback_confirmed: must be set to "true" */
     &oauth_callback_confirmed=true
```

OAuth v1.0/v1.0a

```
request: authorized request token (authorized temporary credentials)
http://oauthbin.com/v1/access-token
     ?data=requested-resource
     /* oauth_consumer_key: equivalent to a username */
     &oauth_consumer_key=key
     /* oauth_token: temporary credentials, identifier */
     &oauth_token=requestkey
     /* oauth_signature_method: PLAINTEXT|HMAC-SHA1|RSA-SHA1 */
     &oauth_signature_method=HMAC-SHA1
     /* oauth_signature: base64 and URL encoded signature value */
     &oauth_signature=4jBs8nrJX7gH7ZPbxnBg3Q%2Fmb1Y%3D
     /* oauth_timestamp: UNIX timestamp */
     &oauth_timestamp=1413820413
     /* oauth_nonce: random unique hexadecimal value */
     &oauth_nonce=3d066fd60c1b7805b4703f31a8a23a50847e623e
     /* oauth_version: must be set to "1.0" */
     &oauth_version=1.0
     /* oauth_verifier: verification code */
     &oauth_verifier=

response: authorized request token (authorized temporary credentials)
     /* oauth_token: temporary credentials, identifier */
     oauth_token=accesskey
     /* oauth_token_secret: temporary credentials, shared-secret */
     &oauth_token_secret=accesssecret
```

OAuth v1.0/v1.0a

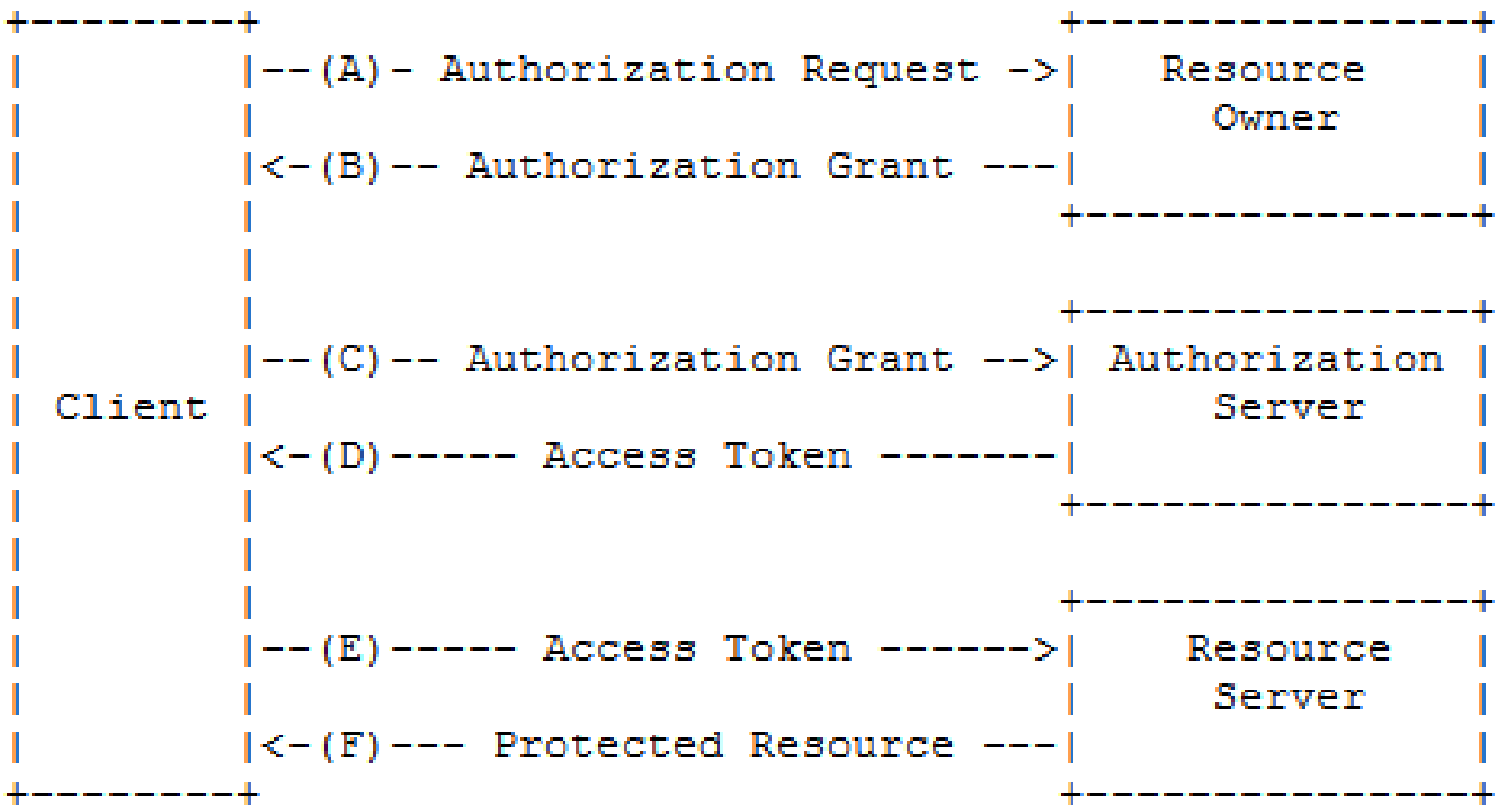
```
request: access token (access data)
http://oauthbin.com/v1/echo
     ?data=requested-resource
     /* oauth_consumer_key: equivalent to a username */
     &oauth_consumer_key=key
     /* oauth_token: temporary credentials, identifier */
     &oauth_token=accesskey
     /* oauth_signature_method: PLAINTEXT|HMAC-SHA1|RSA-SHA1 */
     &oauth_signature_method=HMAC-SHA1
     /* oauth_signature: base64 and URL encoded signature value */
     &oauth_signature=QiAgKrXCK2l%2BEp3Y5y9uxyBpC%2FY%3D
     /* oauth_timestamp: UNIX timestamp */
     &oauth_timestamp=1413820413
     /* oauth_nonce: random unique hexadecimal value */
     &oauth_nonce=555b9f54e42ee1b3e5478c4a3df497d6c46aa321
     /* oauth_version: must be set to "1.0" */
     &oauth_version=1.0

response: access token (access data)
     data=requested-resource
     <content>
```

No static secret keys of Client application are sent through network.

Temporary authorization tickets are computed (HMAC-SHA1 or RSA-SHA1) and sent based on Client application static secret keys.

OAuth v2.0


Client
Resource Owner
Authorization Server
Resource Server
(A)- Authorization Request ->
<-(B)-- Authorization Grant ---
--(C)-- Authorization Grant -->
<-(D)----- Access Token -------
--(E)----- Access Token ------>
<-(F)--- Protected Resource ---

OAuth v2.0

request: Authorization Request	https://www.facebook.com/dialog/oauth?...
response: Authorization Grant	code: AQA75xHsFI76597xURbgMHrag3C477863Qh4Hbp95ajIpgAw26uHTU7Ead_AksJi...
request: Authorization Grant	https://graph.facebook.com/oauth/access_token?...
response: Access Token	access_token: CAACztOcnnVYBAJ1ZC60vHT1dFjSyoiBzec7csSZAFcklNWQdcjsPtzVa8DuB8lY...
request: Access Token	https://graph.facebook.com/me?...
response: Protected Resource	authenticated user: Aron Szabo, 1174323737, https://www.facebook.com/aron.szabo.18

OAuth v2.0

```
request: Authorization Request
https://www.facebook.com/dialog/oauth
     /* response_type: code|token */
     ?response_type=code
     /* client_id: client application public credential */
     &client_id=123456789012345
     /* redirect_uri: absolute URI */
     &redirect_uri=https%3A%2F%2Fwww.anyuri.hu%2F
     /* scope: access rights, needed attributes */
     &scope=user_birthday
     /* state: session identifier */
     &state=12345678901234567890123456789012

response: Authorization Grant
     /* code: temporary authorization code */
     code=AQA8[...]OQm0
```

OAuth v2.0

```
request: Authorization Grant
https://graph.facebook.com/oauth/access_token
     /* grant_type: authorization_code|password|client_credentials|refresh_token */
     ?grant_type=authorization_code
     /* code: temporary authorization code */
     &code=AQA8[...]OQm0
     /* redirect_uri: absolute URI */
     &redirect_uri=https%3A%2F%2Fwww.anyuri.hu%2F
     /* client_id: client application public credential */
     &client_id=123456789012345
     /* client_id: client application private credential */
     &client_secret=12345678901234567890123456789012

response: Access Token
     /* access_token: temporary access token */
     access_token=CAAC[...]ZAxJ
```

OAuth v2.0

```
request: Access Token
https://graph.facebook.com/me
     /* access_token: temporary access token */
     ?access_token=CAAC[...]ZAxJ

response: Protected Resource
     data=requested-resource
     <content>
```

The static secret keys (client_secret) of Client application are sent through network.

There is no security layer inside OAuth v2.0 standard, the environment shall provide such functionality (e.g. SSL/TLS at least).

There is no protection (e.g. nonce, timestamp) against replay attacks.

The access_token shall be kept in secret.

OAuth

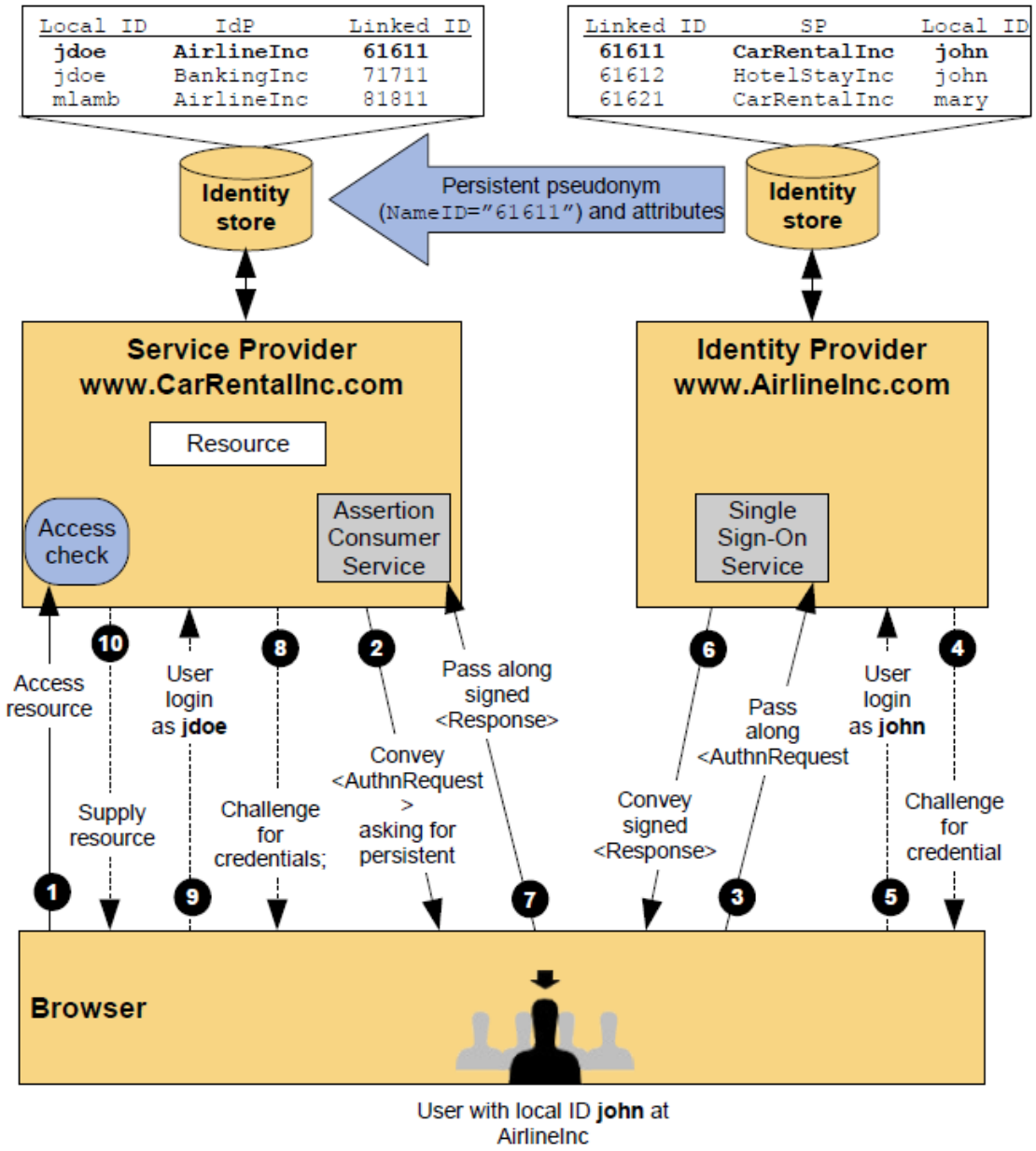
In both cases the OAuth v1.0/v1.0a oauth_token and OAuth v2.0 access_token as tickets are soft tokens that can be intercepted and replayed (e.g. man-in-the-middle attack, replay attack, session hijacking). But...

OAuth v1.0/v1.0a protocol is still less vulnerable to these kind of attacks because of sending no static private credentials and using oauth_timestamp and oauth_nonce parameters as signed data.

So, once again...

> *When compared with OAuth 1.0, the 2.0 specification is more complex, less interoperable, less useful, more incomplete, and most importantly, less secure.*
>
> /Eran Hammer, http://hueniverse.com/2012/07/26/oauth-2-0-and-the-road-to-hell/

SAML v2.0


Local ID IdP Linked ID
jdoe AirlineInc 61611
jdoe BankingInc 71711
mlamb AirlineInc 81811
Linked ID SP Local ID
61611 CarRentalInc john
61612 HotelStayInc john
61621 CarRentalInc mary
Identity store
Persistent pseudonym (NameID="61611") and attributes
Identity store
Service Provider www.CarRentalInc.com
Resource
Access check
Assertion Consumer Service
Identity Provider www.AirlineInc.com
Single Sign-On Service
Access resource
Supply resource
User login as jdoe
Challenge for credentials;
Convey <AuthnRequest> asking for persistent
Pass along signed <Response>
Convey signed <Response>
Pass along <AuthnRequest
User login as john
Challenge for credential
1
10
9
8
2
7
6
3
5
4
Browser
User with local ID john at AirlineInc

SAML v2.0

The SSO (Single Sign-On) core component of SAML model manages the whole workflow and issues the signed and encrypted ticket (Ticket Granting Server).

The IdP (Identity Provider) component of SAML model performs user authentication and stores user authentication credentials.

The IS (Identity Store) component of SAML model performs name ID mapping, retrieves user standing data, user attributes, authorization settings.

The ticket contains the following authentic information:

- who gained access privilege?
- how long has access privilege?
- what is the level of granted access privilege?
- (what can be accessed?)

The goal of an SSO system is to issue a single, centralized and widely acceptable authorization data. The single, centralized and widely acceptable authentication is just a sub-step (but important step) of issuing a ticket.

SAML v2.0

SP (Service Provider) to SSO (Single Sign-On)

```
<saml2p:AuthnRequest>
 <saml2:Issuer>
  <!-- ID of SP -->
 </saml2:Issuer>
 <!-- XML Signature -->
 <saml2p:Extensions>
  <saml2:AttributeStatement>
   <saml2:Attribute>
    <!-- ID of required user attribute -->
   </saml2:Attribute>
  </saml2:AttributeStatement>
 </saml2p:Extensions>
 <!-- ID of required user credential -->
 <saml2p:NameIDPolicy />
 <saml2:Conditions
  xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
  <saml2:AudienceRestriction>
   <saml2:Audience>
    <!-- ID of required user authentication level -->
   </saml2:Audience>
  </saml2:AudienceRestriction>
 </saml2:Conditions>
</saml2p:AuthnRequest>
```

SSO (Single Sign-On) redirects User to IdP (Identity Provider) and performs authentication (based on ID of required user authentication level)

SAML v2.0

SSO (Single Sign-On) to IS (Identity Store)

```
<saml2p:NameIDMappingRequest>
 <saml2:Issuer>
  <!-- ID of SSO -->
 </saml2:Issuer>
 <!-- XML Signature -->
 <saml2p:Extensions>
  <saml2:AttributeStatement>
   <saml2:Attribute>
    <!-- ID of required user attribute -->
   </saml2:Attribute>
  </saml2:AttributeStatement>
 </saml2p:Extensions>
 <saml2:NameID>
  <!-- ID and value of authenticated user -->
 </saml2:NameID>
 <!-- ID of required user credential (mapping ID and value of authenticated user) -->
 <saml2p:NameIDPolicy />
</saml2p:NameIDMappingRequest>
```

IS (Identity Store) returns requested user data to SSO (Single Sign-On)

(sending EncryptedAttribute)

SAML v2.0

SSO (Single Sign-On) to SP (Service Provider)

```
<saml2p:Response>
 [...]
 <!-- XML Encryption -->
 <saml2:Assertion>
  <saml2:Issuer>
   <!-- ID of SSO -->
  </saml2:Issuer>
  <!-- XML Signature -->
  <saml2:Subject>
   <!-- ID and value of authenticated and mapped user -->
  </saml2:Subject>
  <!-- validity of ticket (Assertion) -->
  <saml2:Conditions NotBefore="2014-10-20T10:00:00Z" NotOnOrAfter="2014-10-21T10:00:00Z">
   <saml2:Condition>
    <saml2:Audience>
     <!-- ID of required user authentication level -->
    </saml2:Audience>
   </saml2:Condition>
  </saml2:Conditions>
  <saml2:AttributeStatement>
   <saml2:Attribute>
    <!-- ID and value of required user attribute -->
   </saml2:Attribute>
  </saml2:AttributeStatement>
 </saml2:Assertion>
</saml2p:Response>
```

(sending EncryptedAssertion)

SAML v2.0

The SSO (Single Sign-On) core component of SAML model manages the whole workflow and issues the signed and encrypted ticket (Ticket Granting Server).

User viewpoint:

- good to Users who need end-to-end encryption of tickets and attributes

Are the ticket parameters (e.g. who? how long? on what security level?) imported and set (e.g. in cookies) when SPs create their own user sessions?

Are the ticket contents protected? Are they signed in order to keep integrity and authenticity? Are they encrypted in order to keep confidentiality? (beyond using SSL/TLS layer)

Are the tickets valid for a short period? Can stolen tickets be reused?

SAML v2.0

The IdP (Identity Provider) component of SAML model performs user authentication and stores user authentication credentials.

User viewpoint:

- good to Users who need less strong protection than UID/PWD
- not good to Users who need less strong protection than SSL/TLS

All the sent private user credentials (e.g. password of UID/PWD, one-time-password of SMS or TOTP, signed response of PKI form-based, smartcard-based authentication) as soft tokens can be intercepted and replayed (e.g. man-in-the-middle attack, replay attack, session hijacking). But...

During an SSL/TLS client authentication the handshake can not be performed - in practice - in case of a man-in-the-middle attack.

SAML v2.0

The IS (Identity Store) component of SAML model performs name ID mapping, retrieves user profile data, user attributes, authorization settings.

User viewpoint:

- not good to Users who do not want to be traced and require privacy

Do user attributes need to be retrieved from centralized database (by SPs) or can they be queried directly from User?

SAML v2.0

6.3 Subject Attribute Definitions

These are all attributes that can be queried about the subject of an authentication. They may be requested several times in one query, e.g. with isAgeOver this makes sense. Unknown attributes (attributes not listed in the following table) in a request are ignored.

Friendly Name	Name	Name Format	Description
eIdentifier	http://www.stork.gov.eu/1.0/eIdentifier	CC/CC/Base64	see [9]
Given Name	http://www.stork.gov.eu/1.0/givenName	UTF-8	see [9]
Surname	http://www.stork.gov.eu/1.0/surname	UTF-8	see [9]
Inherited Family Name	http://www.stork.gov.eu/1.0/inheritedFamilyName	UTF-8	see [9]
Adopted Family Name	http://www.stork.gov.eu/1.0/adoptedFamilyName	UTF-8	see [9]
Gender	http://www.stork.gov.eu/1.0/gender	"M" or "F"	see [9]
Date of Birth	http://www.stork.gov.eu/1.0/dateOfBirth	Date	see [9]
Country of Birth	http://www.stork.gov.eu/1.0/countryCodeOfBirth	ISO-3166-3	see [9]
Nationality	http://www.stork.gov.eu/1.0/nationalityCode	Country code	see [9]
Marital Status	http://www.stork.gov.eu/1.0/maritalStatus	S = Single M = Married P = Separated D = Divorced W = Widowed	see [9]
Text Residence Address	http://www.stork.gov.eu/1.0/textResidenceAddress	UTF-8 (with new lines)	see [9]
Canonical Residence Address	http://www.stork.gov.eu/1.0/canonicalResidenceAddress	XML	see 6.4.1
eMail Address	http://www.stork.gov.eu/1.0/eMail	e-mail	see [9]
Title	http://www.stork.gov.eu/1.0/title	UTF-8	see [9]
Residence Permit	http://www.stork.gov.eu/1.0/residencePermit	UTF-8	see [9]
Pseudonym	http://www.stork.gov.eu/1.0/pseudonym	UTF-8	see [9]
Age	http://www.stork.gov.eu/1.0/age	Numeric	see [9]
Is AgedOver	http://www.stork.gov.eu/1.0/isAgeOver	Requested age boundary if true, empty if false	see 5.1.4.8.1.1
Signed Document	http://www.stork.gov.eu/1.0/signedDoc	see below	see 6.5
Citizen QAA Level	http://www.stork.gov.eu/1.0/citizenQAALevel	Numeric {1:4}	Level between 1 and 4
Fiscal Number	http://www.stork.gov.eu/1.0/fiscalNumber	UTF-8	

Table 26: Stork Data definitions

The list of user attributes of STORK specification

source:

D5.8.3b Interface Specification
December 31st 2011

Thank you!