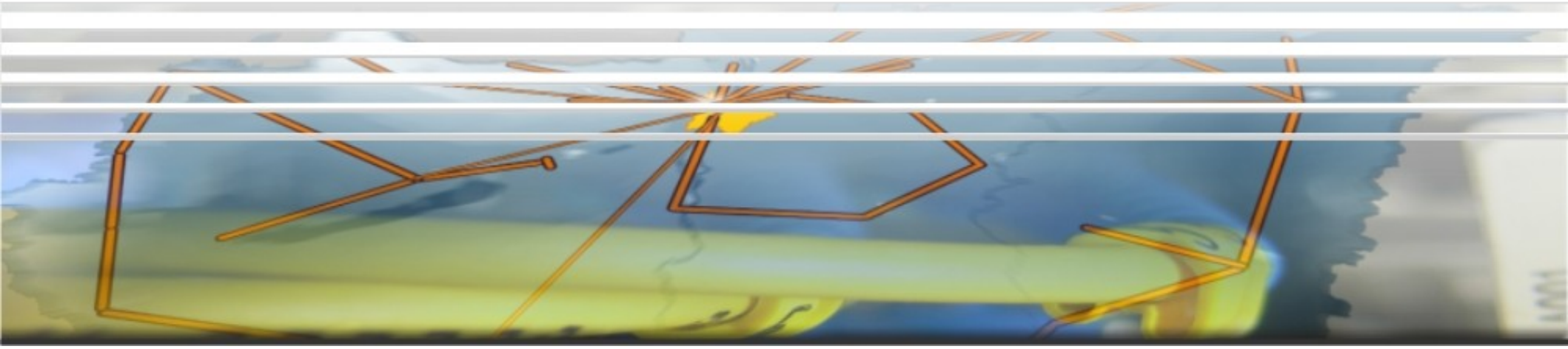
WebRTC Identity in SAML Federations



May 19, 2015
Budapest / Hungary

Mihály Mészáros
NIIF Institute

Agenda

- Education & Research Identity Federations
  - SAML Terminology
  - Overview of SAML auth call flow
- WebRTC Identity model
  - WebRTC(W3C) API, RTCWEB(IETF) security model
  - Terminology
  - Example call flow
  - Trust model
- Demonstrator
- Under the hood details
  - JSON Web Token
  - Trust chain between Federated Auth(SAML) and WebRTC

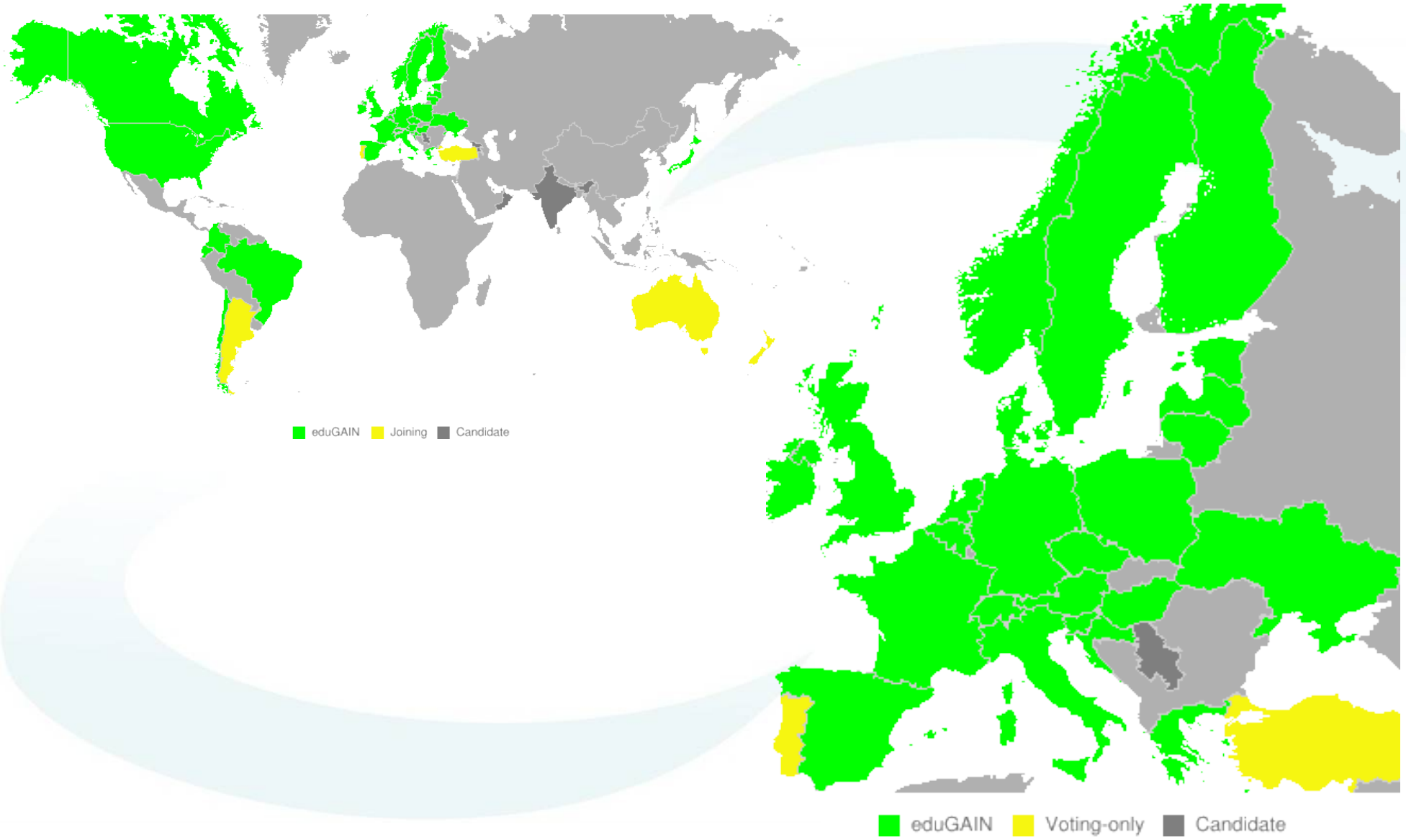
Why Identity/authentication is important in RTC? (education in focus)

- If we hear or see each-other why it is important?
  - We can identify the other party.
- Some use-cases where Identity could give a plus
  - In a University VoIP / Video Conference network.
    - Connection with SIP RFC4474
  - e-Learning consultations
  - Meeting with an Unknown Professor far away
  - Share research results e.g. VC with a publisher
  - Speak to a Service Provider where trust matters
    - e.g. Bank, etc.
  - Remote interviews for grants/scholarship
  - Any work group/task force meeting where participants have never met before
  - Etc.

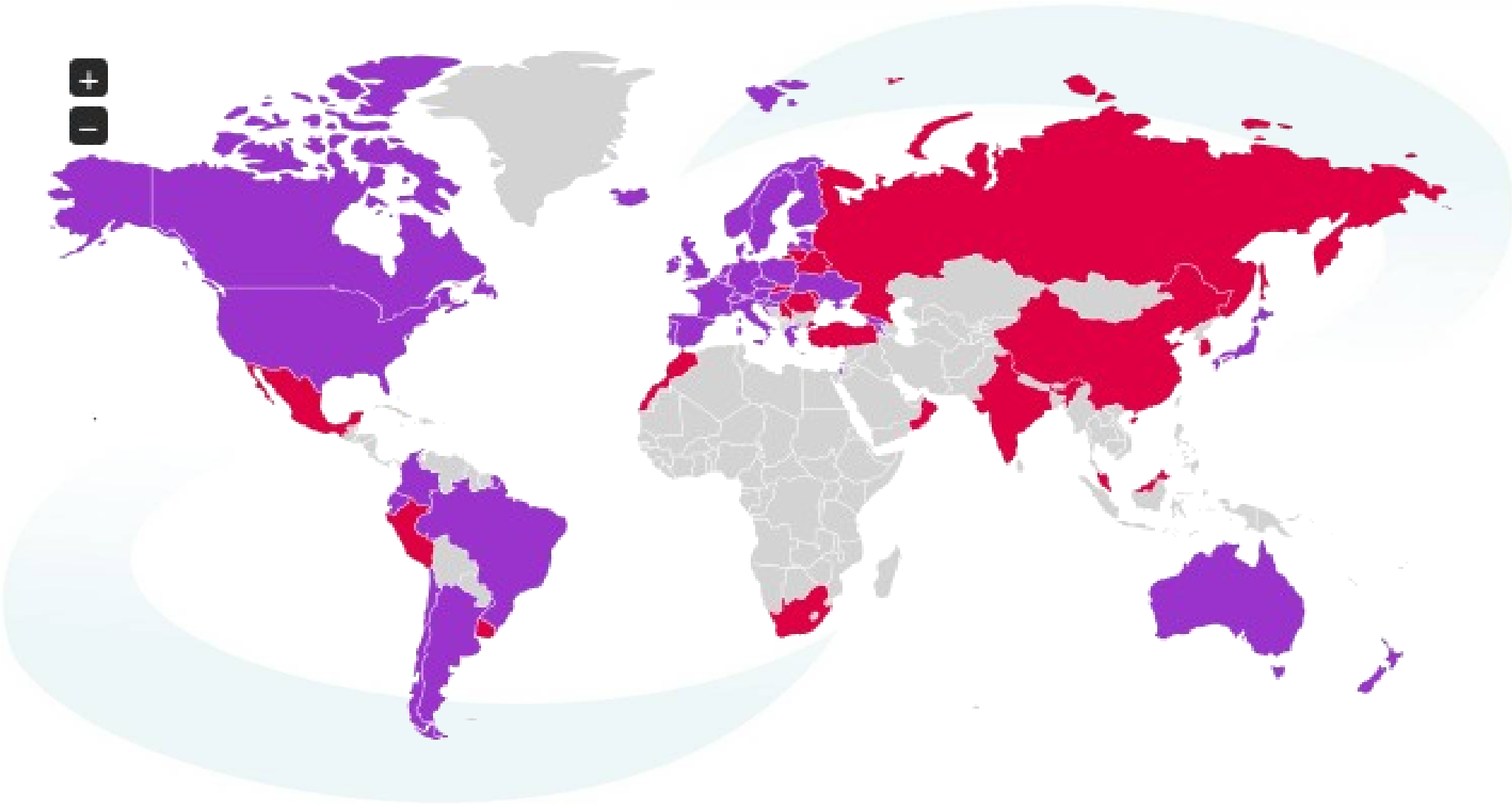
EduID/HREF, eduGAIN, REFEDS

- eduID.hu/HREF – Hungarian identity federation
  - Hungarian Research and Educational Federation
    - HREF is a SAML2-based Identity Federation of Hungarian higher education and research institutions, public collections and other content providers
- EduGAIN – The European AAI Federation and beyond
  - Interconnects identity federations around Europe
  - One of the GÉANT services
- REFEDS - World wide
  - Research and Education Federations
    - REFEDS is a community of identity federations from around the world

EduGAIN



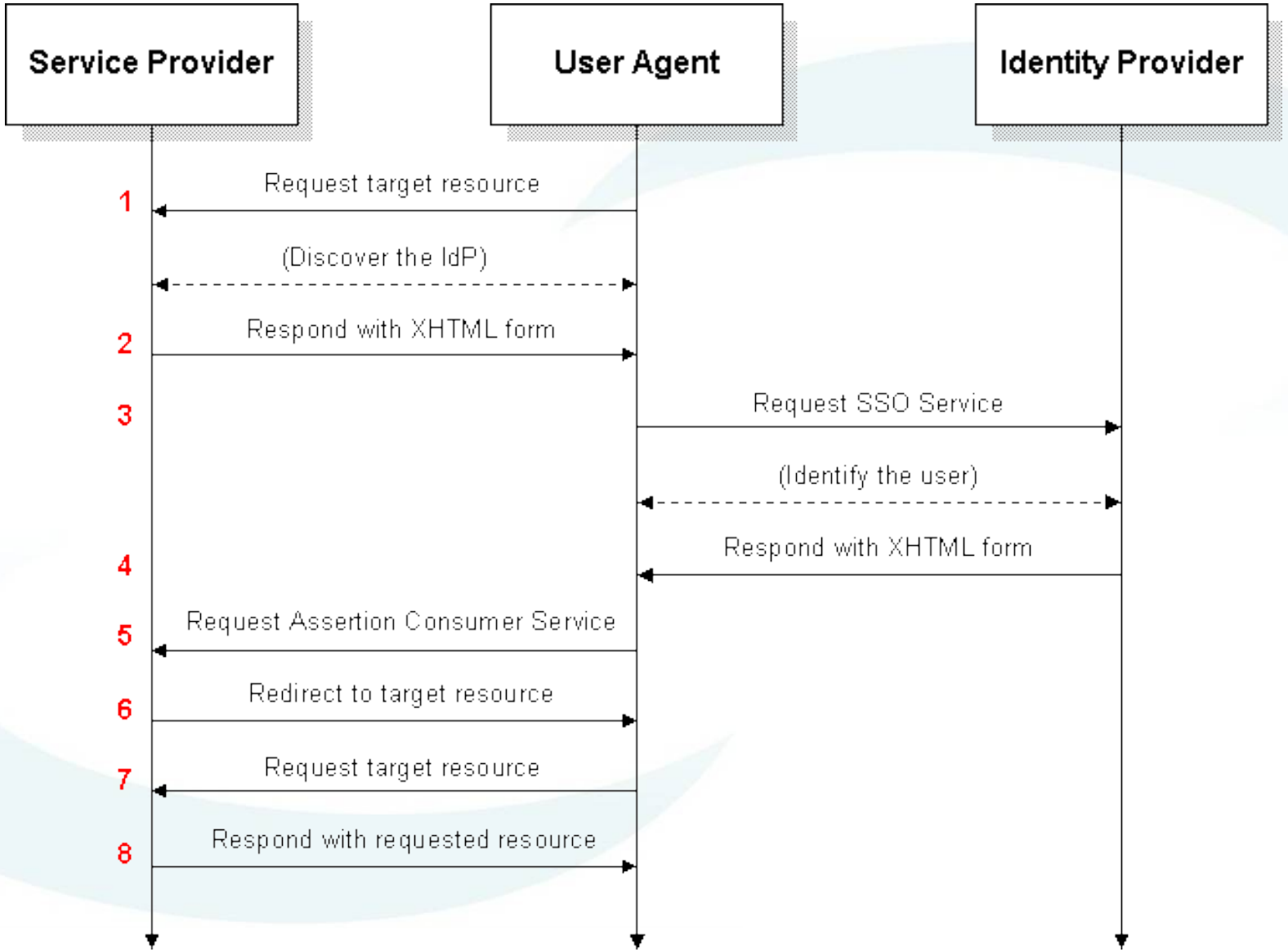
REFEDS



SAML Terminology

- The principal (typically a user)
- IdP: An Identity Provider (IdP), also known as Identity Assertion Provider, is responsible for
  - providing identifiers for users looking to interact with a system, and
  - asserting to such a system that such an identifier presented by a user is known to the provider, and
  - possibly providing other information about the user that is known to the provider.
- SP: The service provider (SP) requests and obtains an identity assertion from the identity provider. On the basis of this assertion, the service provider can make an access control decision – in other words it can decide whether to perform some service for the connected principal.

SAML HTTP POST binding



Typical simple SAML login example

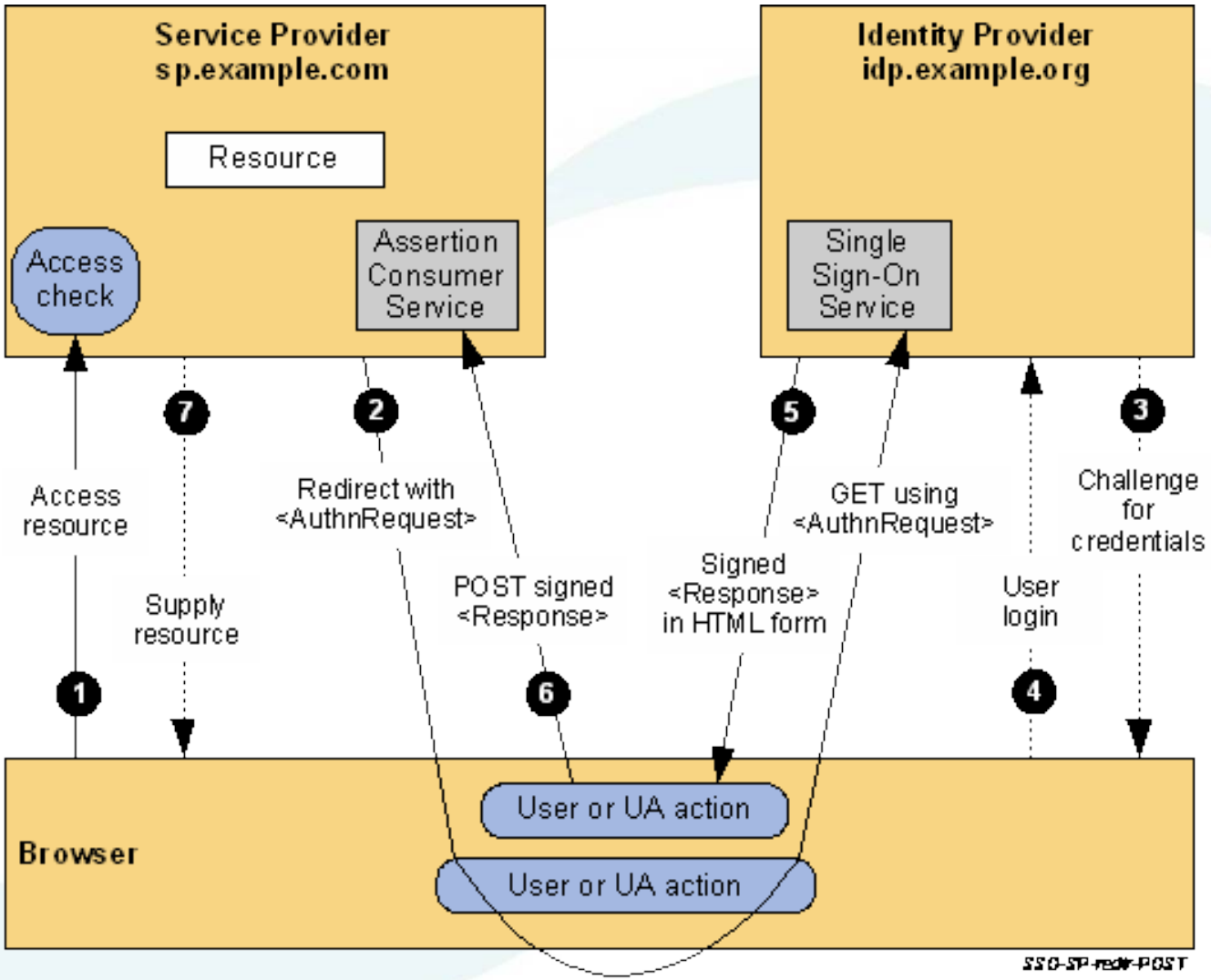
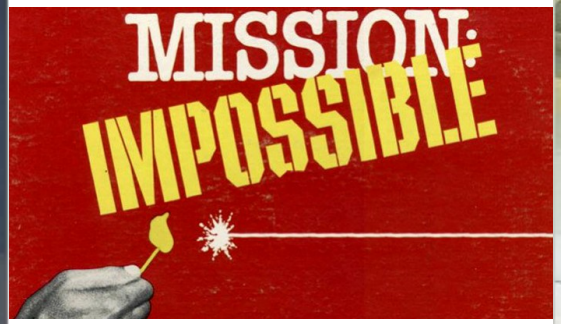


Image source:
http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02_html_m1960c26f.gif

Important barriers for WebRTC integration

- SAML IdP and SP mutually authenticate each other:
  - The IdP Assertion is signed by the IdP.
    - In WebRTC we need a signed identity from the federation, so this isn't a barrier of course. We need this signed assertion.
  - Commonly, the Response object (that encapsulates the Assertion) is signed, but furthermore in order to prevent browser-based attacks the IdP could encrypt the entire SAML response Object (the envelope around the assertion) with the SP public key, so only the SP could access the response and so the assertion.
    - So to fetch identity assertion from IDP directly could be a problem!
- There is no SAML standard way to get signed assertion directly from IdP if we aren't an SP.
- It is not possible to run a Web App code in a browser that act as a SAML SP.

The challenge: WebRTC Identity & SAML
Is it possible?



Goal: Authenticated RTC / VC even on a not fully trusted web site

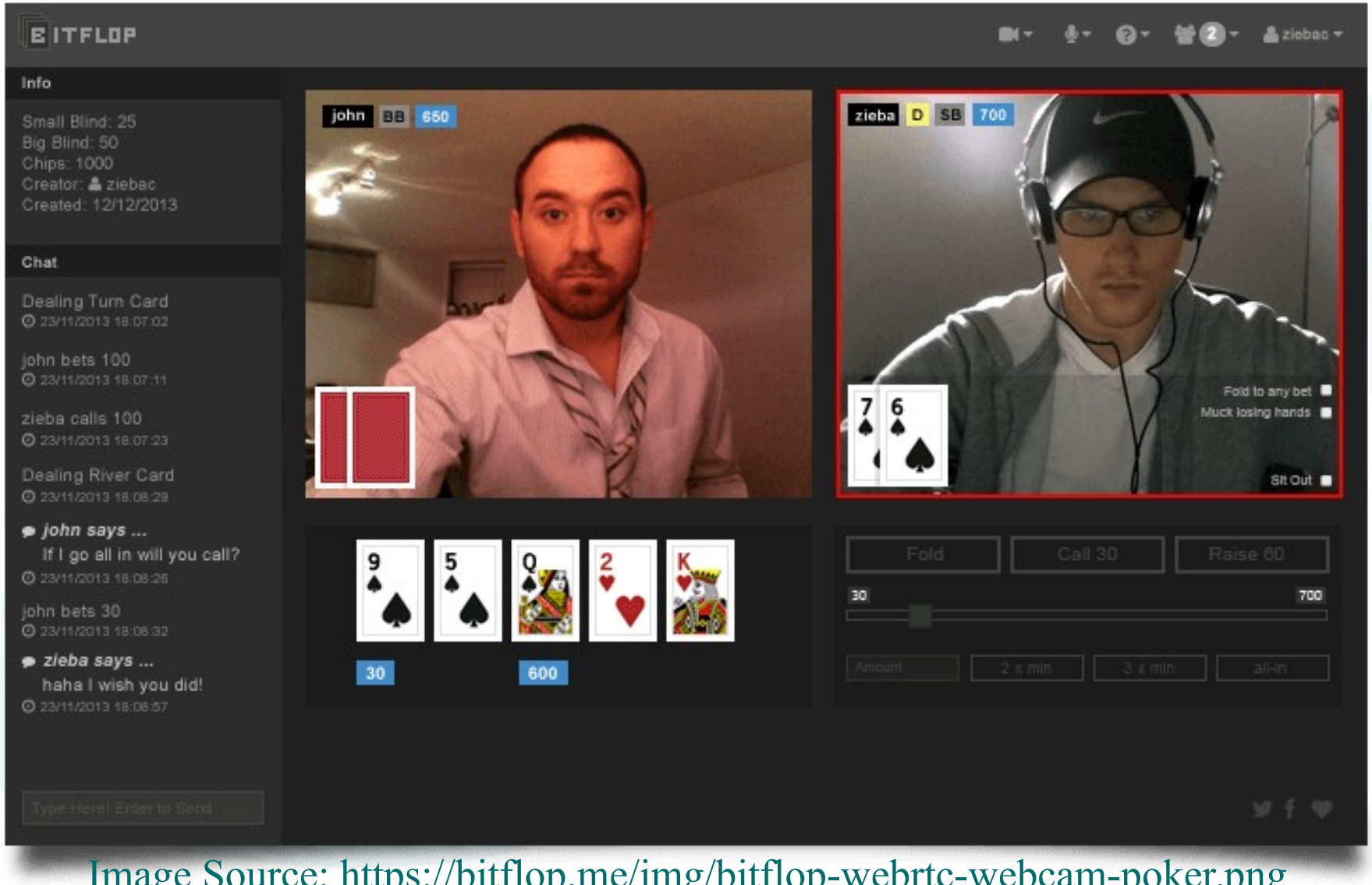
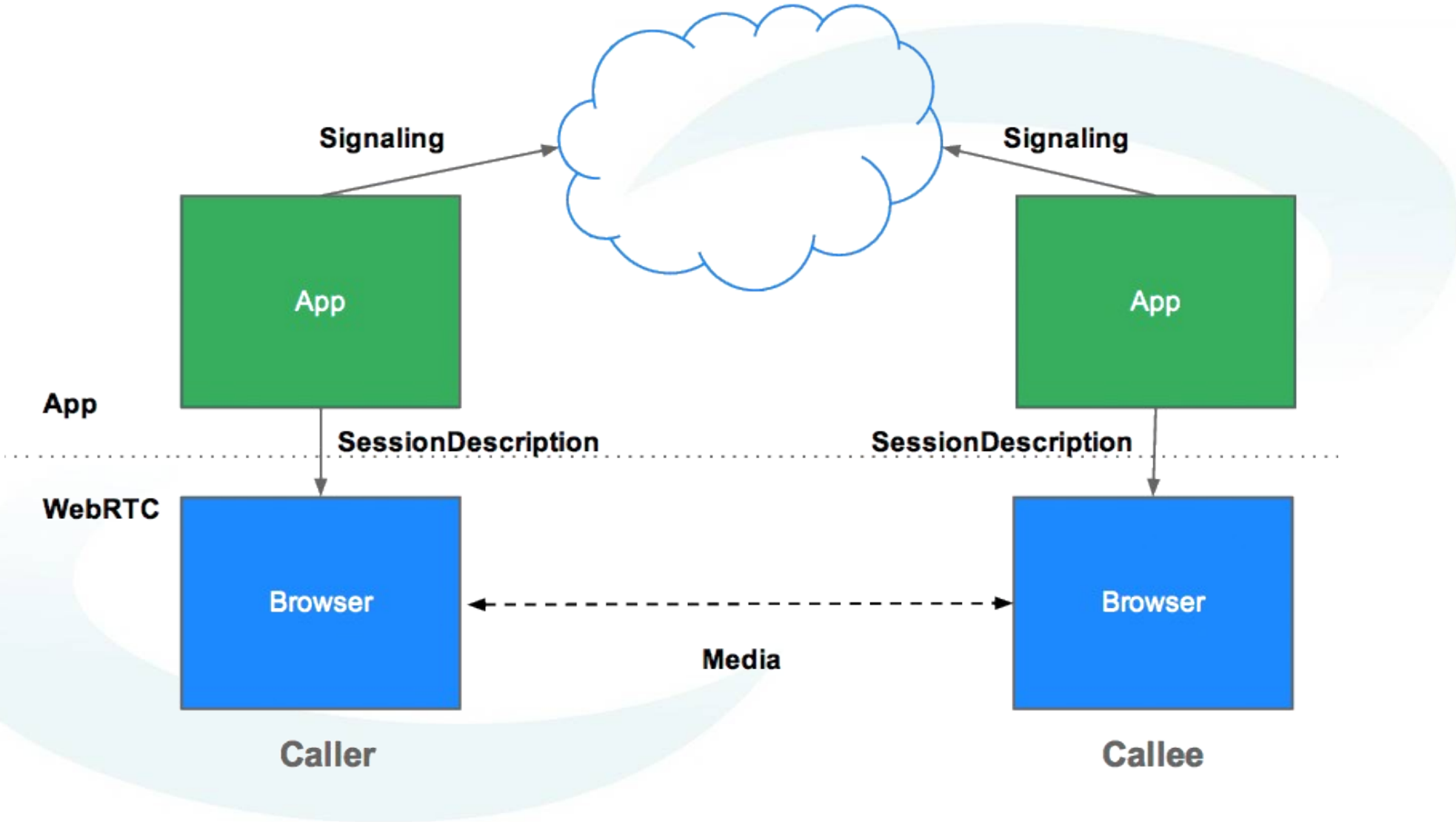


Image Source: https://bitflop.me/img/bitflop-webrtc-webcam-poker.png

Trust in Browser only
(slide source: EKR rtcweb13.pdf)

- Browser acts as the Trusted Computing Base
  - Only piece of the system user can really trust
  - Job is to enforce user's desired security policies
- Authenticated entities
  - Identity is checked by the browser (sometimes transitively)
- Unauthenticated entities
  - Random other network elements who send and receive traffic
- Authenticated ≠ trusted
  - authentication is the basis of trust decisions

JavaScript Session Establishment Protocol (JSEP)
IETF RTCWEB Workgroup



W3C WebRTC Identity API

- API Identity related part
  - PeerConnection functions
    - setIdentityProvider(provider, protocol)
    - getIdentityAssertion()
  - PeerConnection event handlers
    - onpeeridentity
    - onidentityresult
    - onidpassertionerror
    - onidpvalidationerror
  - PeerConnection Attribute
    - PeerIdentity (readonly)

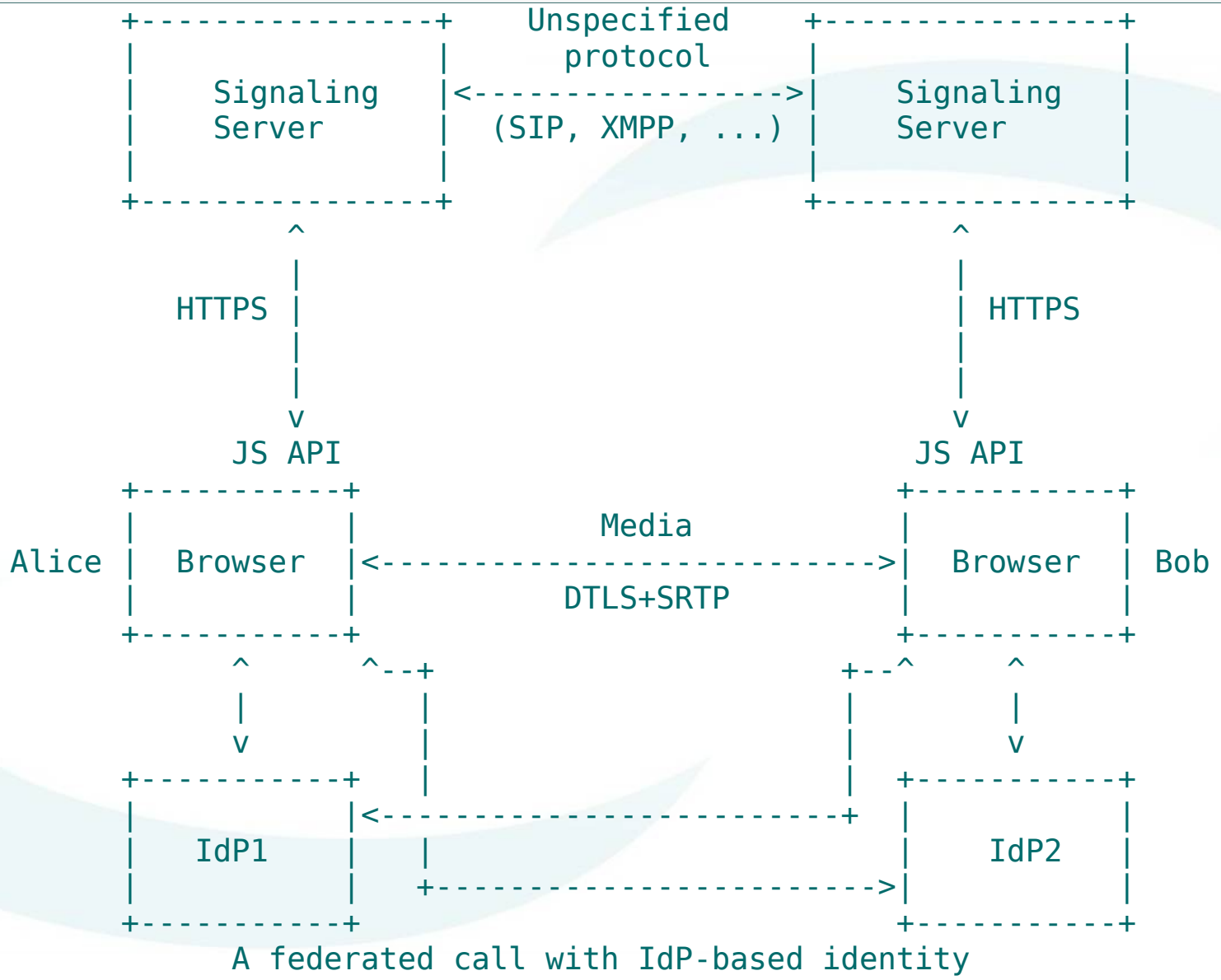
Requesting Identity Assertions (W3C WebRTC API)

- The RTCPeerConnection instantiates an IdP proxy as described in Identity Provider Selection section and waits for the IdP to signal that it is ready.
- The RTCPeerConnection sends a "SIGN" message to the IdP proxy. This message includes the material the RTCPeerConnection desires to be bound to the user's identity.
- If the user has been authenticated by the IdP, and the IdP is willing to generate an identity assertion, the IdP generates an identity assertion. This step depends entirely on the IdP. The methods by which an IdP authenticates users or generates assertions is not specified, though this could involve interacting with the IdP server or other servers.
- The IdP proxy sends a response containing the identity assertion to the RTCPeerConnection over the message channel.
- The RTCPeerConnection may store the identity assertion for use with future offers or answers.
- If the identity request was triggered by a createOffer() or createAnswer(), then the assertion is inserted in the offer/answer SDP.

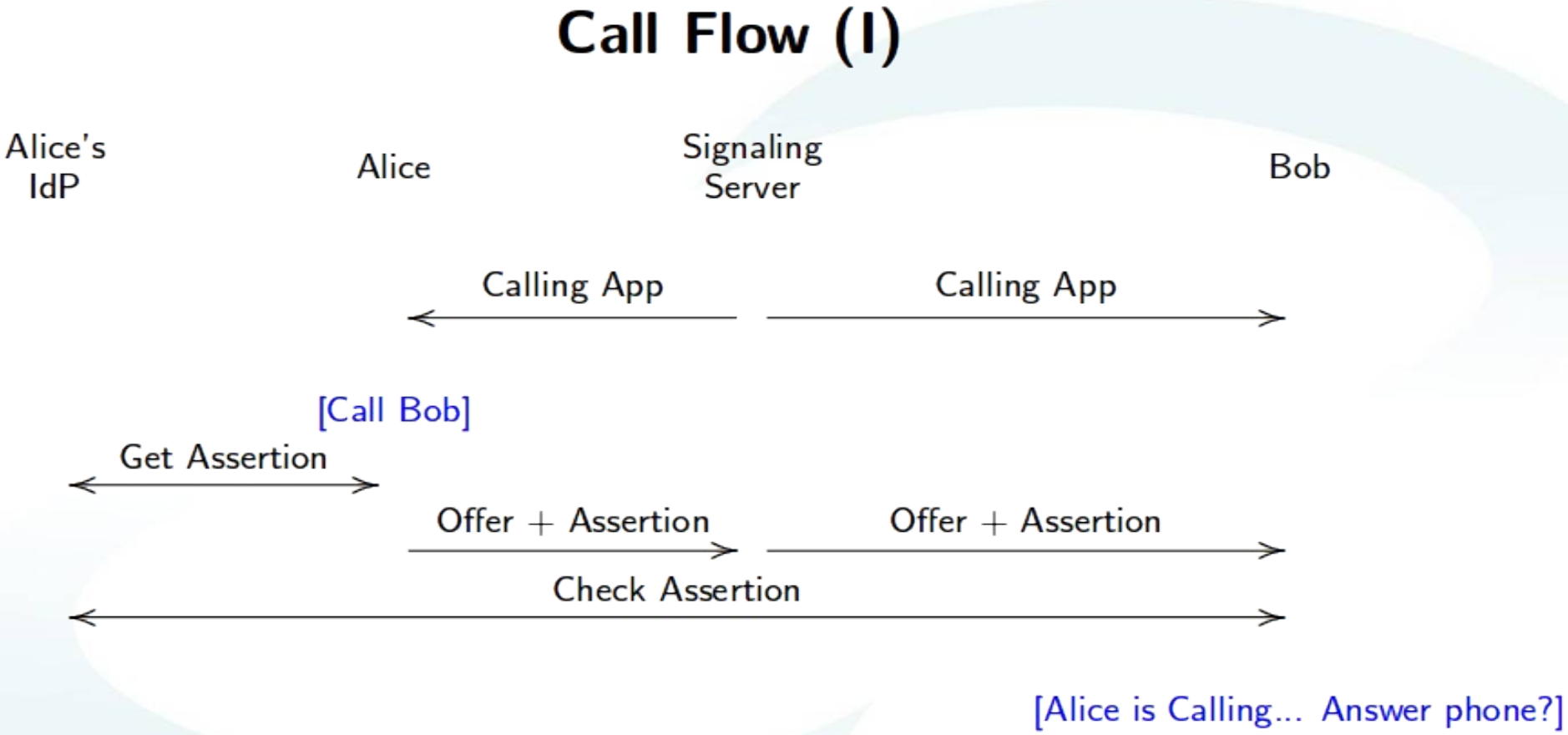
Verifying Identity Assertions (W3C WebRTC API)

- The RTCPeerConnection instantiates an IdP proxy as described in Identity Provider Selection section and waits for the IdP to signal that it is ready.
- The RTCPeerConnection sends a "VERIFY" message to the IdP proxy. This message includes the assertion from the offer/answer which is to be verified.
- The IdP proxy verifies the identity assertion (depending on the authentication protocol this could involve interacting with the IDP server).
- Once the assertion is verified, the IdP proxy sends a response containing the verified assertion results to the RTCPeerConnection over the message channel.
- The RTCPeerConnection validates that the fingerprint provided by the IdP in the validation response matches the certificate fingerprint that is, or will be, used for communications. This is either by:
- Ensuring that there is only a single a=fingerprint value across all media sections in the SDP that matches the fingerprint provided by the IdP.
- Waiting for all DTLS connections to be establishes and checking that the certificate fingerprints on all connections matches the one provided by the IdP.
- The RTCPeerConnection validates that the domain portion of the identity matches the domain of the IdP as described in [RTCWEB-SECURITY-ARCH].
- The RTCPeerConnection stores the assertion in the peerIdentity attribute and raises a simple event named peeridentity at itself. The assertion information to be displayed must contain the domain name of the IdP as provided in the assertion.
- The browser may display identity information to a user in browser UI. Any user identity information that is displayed in this fashion must use a mechanism that cannot be spoofed by content.

RTCWEB Security architecture Overview

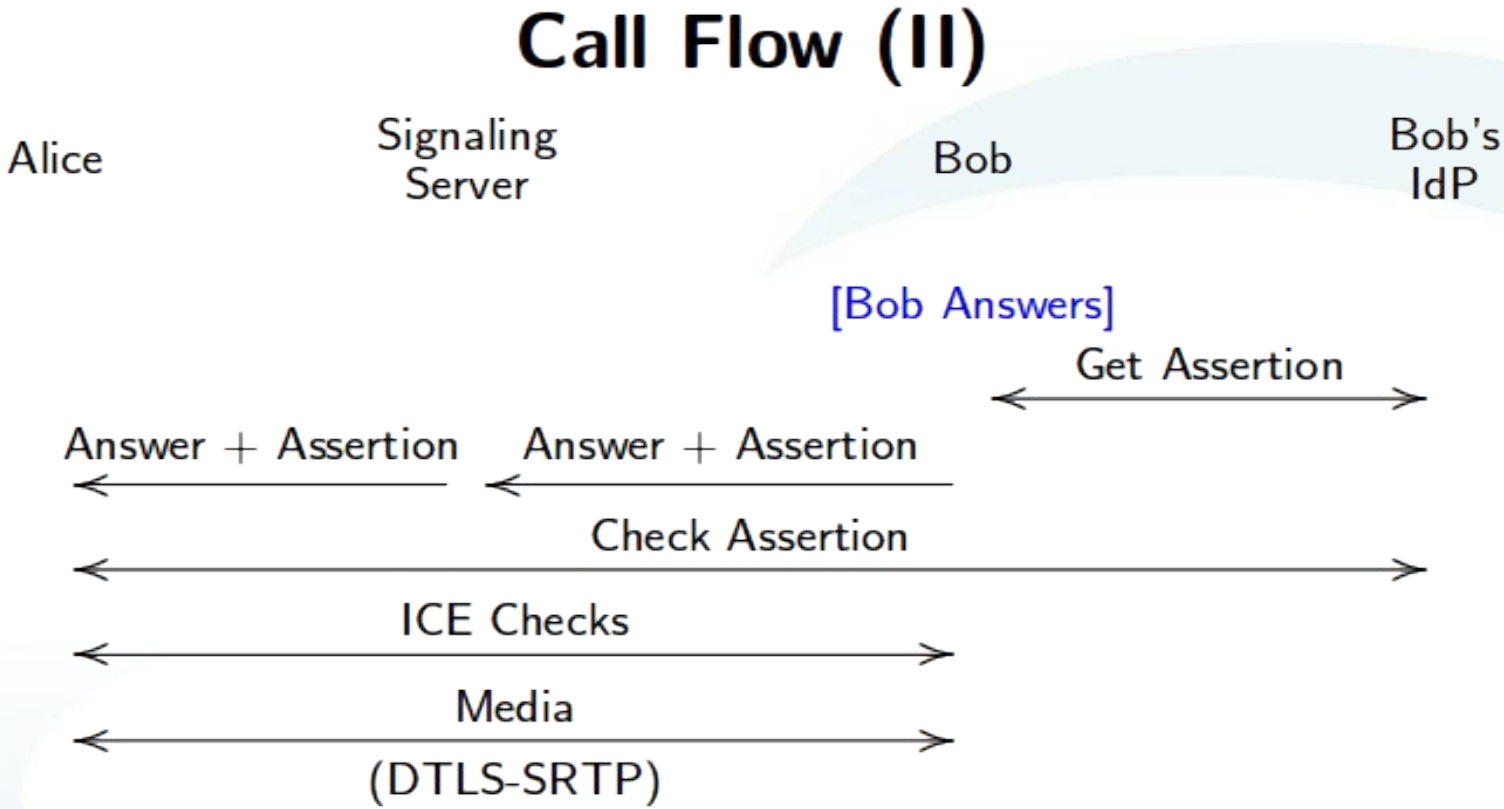


Offer (Fingerprint + Assertion)



Origin: http://www.ietf.org/proceedings/82/slides/rtcweb-13.pdf

Answer (Fingerprint+Assertion)



Origin: http://www.ietf.org/proceedings/82/slides/rtcweb-13.pdf

SDP (Session Description Protocol)

```
V=0
o=mozilla...THIS_IS_SDPARTA-37.0.2 6918706470784164318 0 IN IP4 0.0.0.0
S=-
T=0 0
a=fingerprint:sha-256 93:C0:23:2F:A2:00:00:0D:51:AC:D2:54:65:F4:3B:7D:92:DC:88:33:51:23:40:72:91:83:5B:01:2F:50:78:3F
a=group:BUNDLE sdparta_0 sdparta_1 sdparta_2
a=ice-options:trickle
a=identity:eyJpZHAiOnsiZG9tYWluIjoibmlpZi5odSIsInByb3RvY29sIjoiaWRwLmh0bWwifSwiYXNzZXJ0a
W9uIjoiZXlKaGJHY2lPaUpTVXpJMU5pSXNJblI1Y0NJNklrcFhVeUo5LmV5SmpiMjUwWlc1MGN5STZleUptYVc1b
lpYSndjbWx1ZENJNlczc2lZV3huYjNKcGRHaHRJam9pYzJoaExUSTFOaUlzSW1ScFoyVnpkQ0k2SWprek9rTXdPa
kl6T2pKR09rRXlPakF3T2pBd09qQkVPalV4T2tGRE9rUXlPalUwT2pZMU9rWTBPak5DT2pkRU9qa3lPa1JET2pnN
E9qTXpPalV4T2pJek9qUXdPamN5T2preE9qZ3pPalZDT2pBeE9qSkdPalV3T2pjNE9qTkdJbjFkZlN3aWFXUmxib
lJwZEhraU9pSnRhWE5wUUc1cGFXWXVhSFVpZlEuSTVQdGhKNFFDT05TOFVXd25OOUh3MEdaTDl3d0RBVGRrTWtFW
llmdlNVTTJ6Umd5R09WSGgzRmpnc2FPZklkRnFsNUx6azBFbndVOTNQOUlCQ0xZOWtia3V1c0V1S25YRGVNLTNIN
WFmdTJvZl9CTlZjUnB3MmdBdlNBbVR6SlltcEpqMFEtdmV0TmtVT1huZE9HLUIzT3ZGb3QwZVNENlZSNUdhb2wyc
GduS3FSTktOd3dacEZ1eUZZbFRodHJIdGNiT19WV3o4QnZpTThKS25OdExWd1JxNUhMX2ZLTlRCNzFDYkoyWmh5W
XU1UEdwWDhXcXJMWC1ybm5YSFY3RnhoTTh5OHdrLWd5cnRZazVnbFlZeUFrcTVqZklSXzRzWER5d19Qc1BWTW1aZ
XltenVGV3BQTzVFWlJYR0ZpRjFET0o4Q0Q3Z3Zta2dUdlBXSWpkemtBIn0=
m=audio 9 RTP/SAVPF 109 9 0 8
c=IN IP4 0.0.0.0
a=sendrecv
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=ice-pwd:3969306b18d3deed8c120a8bdec02112
a=ice-ufrag:93f1515f
a=mid:sdparta_0
a=rtcp-mux
a=rtpmap:109 opus/48000/2
a=rtpmap:9 G722/8000/1
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=setup:actpass
a=ssrc:2076263852 cname:{074bac48-3b26-4f2c-9cf9-06021ca088be}
m=video 9 RTP/SAVPF 120 126 97
…
```

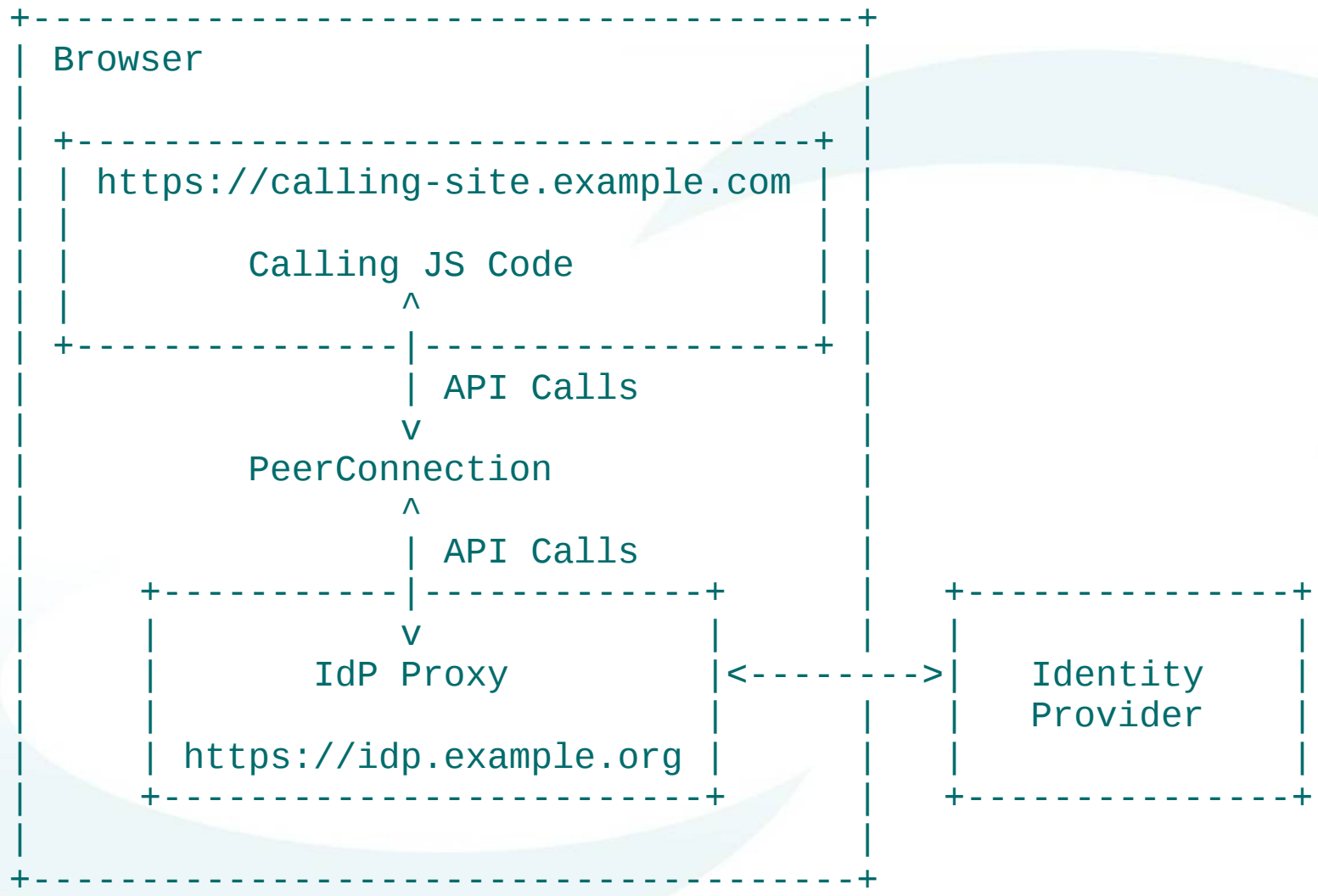
Base64decode (a=identity)


```
a=identity:eyJpZHAiOnsiZG9tYWluIjoibmlpZi5odSIsInByb3RvY29sIjoiaWRwLmh0bWwifSwiYXNzZXJ0aW9u
IjoiZXlKaGJHY2lPaUpTVXpJMU5pSXNJblI1Y0NJNklrcFhVeUo5LmV5SmpiMjUwWlc1MGN5STZleUptYVc1blpYSnd
jbWx1ZENJNlczc2lZV3huYjNKcGRHaHRJam9pYzJoaExUSTFOaUlzSW1ScFoyVnpkQ0k2SWprek9rTXdPakl6T2pKR0
9rRXlPakF3T2pBd09qQkVPalV4T2tGRE9rUXlPalUwT2pZMU9rWTBPak5DT2pkRU9qa3lPa1JET2pnNE9qTXpPalV4T
2pJek9qUXdPamN5T2preE9qZ3pPalZDT2pBeE9qSkdPalV3T2pjNE9qTkdJbjFkZlN3aWFXUmxiblJwZEhraU9pSnRh
WE5wUUc1cGFXWXVhSFVpZlEuSTVQdGhKNFFDT05TOFVXd25OOUh3MEdaTDl3d0RBVGRrTWtFWllmdlNVTTJ6Umd5R09
WSGgzRmpnc2FPZklkRnFsNUx6azBFbndVOTNQOUlCQ0xZOWtia3V1c0V1S25YRGVNLTNINWFmdTJvZl9CTlZjUnB3Mm
dBdlNBbVR6SlltcEpqMFEtdmVOTmQOUlCQ0xZOWtia3V1c0V1S25YRGVNLTNINWFmdTJvZl9CTlZjUnB3Mm
RnhoTTh5OHdrLWd5cnRZazVnbFlZeUFrcTVqZklSXzRzWER5d19Qc1BWTW1aZXltenVGV3BQTzVFWlJYR0ZpRjFET0o
4Q0Q3Z3Zta2dUdlBXSWpkemtBIn0=
```


Base64decoded identity:


```
{
  "idp": {
    "domain": "niif.hu",
    "protocol": "idp.html"
  },
  "assertion":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXUyJ9.eyJjb250ZW50cyI6eyJmaW5nZXJwcmludCI6W3siYWxnb3JpdGhtIj
oic2hhLTI1NiIsImRpZ2VzdCI6IjkzOkMwOjIzOjJGOkEyOjAwOjAwOjBEOjUxOkFDOkQyOjU0OjY1OkY0OjNCOjdEOj
kyOkRDOjg4OjMzOjUxOjIzOjQwOjcyOjkxOjgzOjVCOjAxOjJGOjUwOjc4OjNGIn1dfSwiaWRlbnRpdHkiOiJtaXNpQG
5paWYuaHUifQ.I5PthJ4QCONS8UWwnN9Hw0GZL9wwDATdkMkEZYfvSUM2zRgyGOVHh3FjgsaOfIdFql5Lzk0EnwU93P9
IBCLY9kbkuusEuKnXDeM-3H5afu2of_BNVcRpw2gAvSAmTzJYmpJj0Q-vetNkUOXndOG-
B3OvFot0eSD6VR5Gaol2pgnKqRNKNwwZpFuyFYlThtrHtcbO_VWz8BviM8JKnNtLVwRq5HL_fKNTB71CbJ2ZhyYu5PGp
X8WqrLX-rnnXHV7FxhM8y8wk-
gyrtYk5glYYyAkq5jfIR_4sXDyw_PsPVMmZeymzuFWpPO5EZRXGFiF1DOJ8CD7gvmkgTvPWIjdzkA"
}
```

WebRTC App & Sandboxed IdP Proxy & IdP



IdP Proxy

https://example.org/.well-known/idp-proxy/protocol

In order to provide security without trusting the calling site, the PeerConnection component of the browser must interact directly with the IdP. The details of the mechanism are described in the W3C API specification, but the general idea is that the PeerConnection component downloads JS from a specific location on the IdP dictated by the IdP domain name. That JS (the "IdP proxy") runs in an isolated security context within the browser and the PeerConnection talks to it via a secure message passing channel.

Note that there are two logically separate functions here:

- Identity assertion generation.
- Identity assertion verification.

IdP Proxy

- This approach is flexible
- Glue between the AAI and WebRTC
  - Allows multiple AAI mechanism
  - In RFC example for
    - BrowserID
    - OAuth
  - RFC mentioning other Identity Provider technologies:
    - Federated Google Login
    - Facebook Connect
    - OpenID
    - WebFinger
  - The beauty of this concept that the two parties could use different IdP technologies to prove their Identity to each other!

WebRTC Identity with BrowserID

(Example from security model)

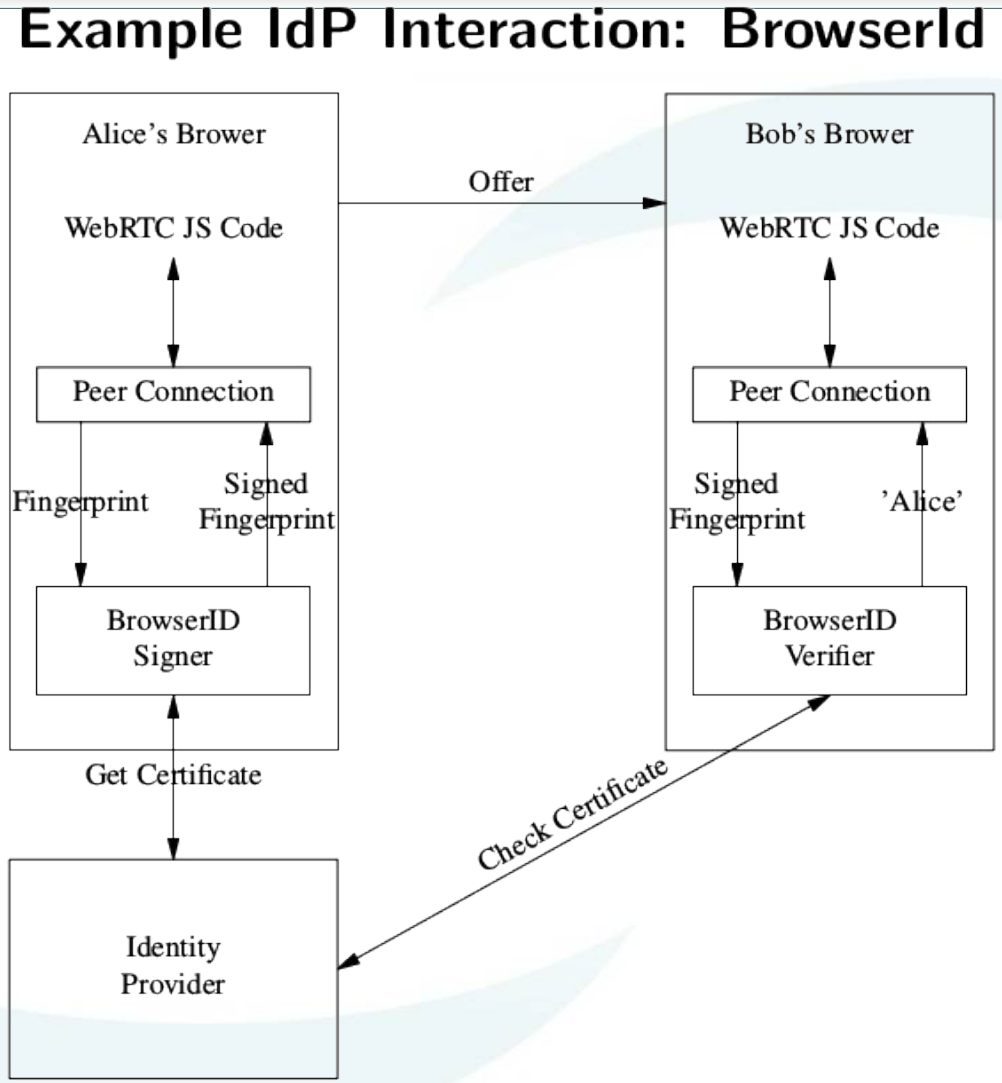


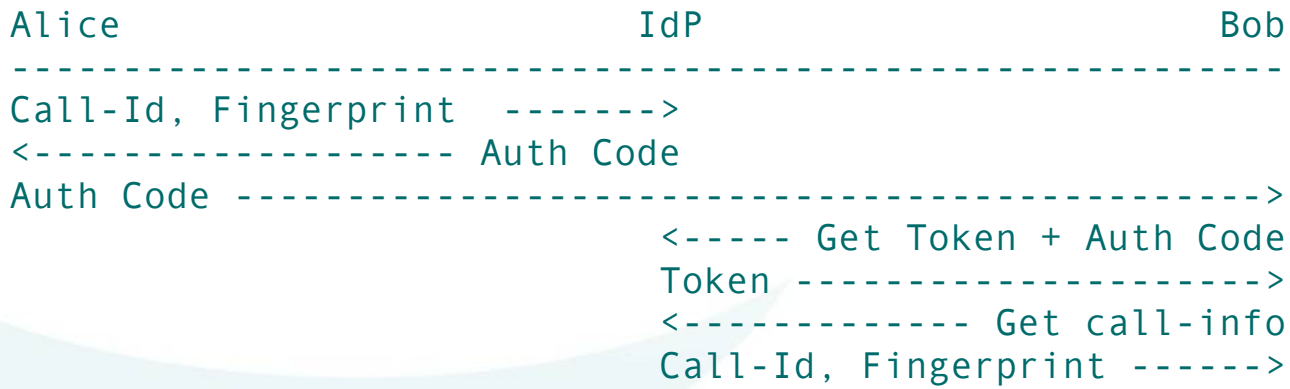
Image Source: http://www.ietf.org/proceedings/82/slides/rtcweb-13.pdf

WebRTC Identity with OAuth

(Example from security model)

Mapping of OAuth concepts to WebRTC concepts:

OAuth	WebRTC
Client	Relying party
Resource owner	Authenticating party
Authorization server	Identity service
Resource server	Identity service



SAML+ WebRTC Identity Demonstrator

https://goo.gl/WpXR0W

https://ficko.vvc.niif.hu/

Firefox version:37.0.2

Under the Hood

- WebRTC is a moving target.
  - Identity is in Draft and Eric Rescorla (EKR) author of the webrtc security model works also in Mozilla team on implementation of the WebRTC Identity in Firefox
  - Even on the only WebRTC identity implementation, this feature is not enabled by default!
  - The security model draft is slightly changing version to version. Iterating like a feedback mechanism according the gathered implementation experiences and knowledge.
- Firefox implemented only the WebRTC security model identity part.
  - At least AFAIK
- Enable WebRTC Identity in Firefox:
  - about:config
    - Identity enable
      - media.peerconnection.identity.enabled;true
    - To debug sandbox application enable the following too:
      - devtools.debugger.remote-enabled;true
      - devtools.chrome.enabled;true
      - devtools.inspector.remote;true

Design concept of the demo IdP proxy code

- Assertion + encryption key fingerprint are glued together in a signed JSON Web Token (JWT) bundle.
- Signature is based on asymmetric cryptography. So with a public-private X.509 key pair.
  - The private key is not stored in IdP Proxy code!
  - Signing server side not in the browser.
- To connect the two worlds (SAML & WebRTC)
  - The (identity domain part) SP's private key is used to sign the JWT.
  - Federation metadata could be used to validate the signature.
- User Authentication is required to access the JWT Signer Web Application service. That has only one input the RTC communication encryption key fingerprint.
  - Signing code could be fully integrated in SP as module in case Simplesamlphp.
  - Or the same service could be an app on top of Shibboleth

Terminology, Identity & Proxy URL examples

- According WebRTC security model
  - The Identity Provider (IdP):
    - JWT Signer App on top of SAML SP
  - IdP Proxy:
    - HTML and JS Glue code in a sandbox (From FF38 only JS)
- My own terminology
  - JSON Web Token Signer Service:
    - PHP code cryptographically binding the encryption key fingerprint to the identity
- Examples
  - Identity:
    - misi@niif.hu
  - IdP Proxy location URL:
    - https://niif.hu/.well-known/idp-proxy/default

Trust chain between Federated Auth(SAML) and WebRTC,

- Sign
  - SAML IdP → SAML SP
    - mutual cert based authN
    - Signed assertion verification
  - SAML SP → JWT signer service
    - On server the application receives the identity attribute. e.g. EPPN
      - Environment for Shibboleth or PHP var for SimpleSamlPhp (SSP)
    - Receives in HTTP from sandbox the RTC encryption key fingerprint.
    - Signs with the SAML SP private key
- Verify
  - Get the federation Metadata
  - Extracts the identity domain part SP public key
  - Verify the signature of the JSON Web Token (JWS) based on it

SDP

```
V=0
o=mozilla...THIS_IS_SDPARTA-37.0.2 6918706470784164318 0 IN IP4 0.0.0.0
S=-
T=0 0
a=fingerprint:sha-256 93:C0:23:2F:A2:00:00:0D:51:AC:D2:54:65:F4:3B:7D:92:DC:88:33:51:23:40:72:91:83:5B:01:2F:50:78:3F
a=group:BUNDLE sdparta_0 sdparta_1 sdparta_2
a=ice-options:trickle
a=identity:eyJpZHAiOnsiZG9tYWluIjoibmlpZi5odSIsInByb3RvY29sIjoiaWRwLmh0bWwifSwiYXNzZXJ0a
W9uIjoiZXlKaGJHY2lPaUpTVXpJMU5pSXNJblI1Y0NJNklrcFhVeUo5LmV5SmpiMjUwWlc1MGN5STZleUptYVc1b
lpYSndjbWx1ZENJNlczc2lZV3huYjNKcGRHaHRJam9pYzJoaExUSTFOaUlzSW1ScFoyVnpkQ0k2SWprek9rTXdPa
kl6T2pKR09rRXlPakF3T2pBd09qQkVPalV4T2tGRE9rUXlPalUwT2pZMU9rWTBPak5DT2pkRU9qa3lPa1JET2pnN
E9qTXpPalV4T2pJek9qUXdPamN5T2preE9qZ3pPalZDT2pBeE9qSkdPalV3T2pjNE9qTkdJbjFkZlN3aWFXUmxib
lJwZEhraU9pSnRhWE5wUUc1cGFXWXVhSFVpZlEuSTVQdGhKNFFDT05TOFVXd25OOUh3MEdaTDl3d0RBVGRrTWtFW
llmdlNVTTJ6Umd5R09WSGgzRmpnc2FPZklkRnFsNUx6azBFbndVOTNQOUlCQ0xZOWtia3V1c0V1S25YRGVNLTNIN
WFmdTJvZl9CTlZjUnB3MmdBdlNBbVR6SlltcEpqMFEtdmV0TmtVT1huZE9HLUIzT3ZGb3QwZVNENlZSNUdhb2wyc
GduS3FSTktOd3dacEZ1eUZZbFRodHJIdGNiT19WV3o4QnZpTThKS25OdExWd1JxNUhMX2ZLTlRCNzFDYkoyWmh5W
XU1UEdwWDhXcXJMWC1ybm5YSFY3RnhoTTh5OHdrLWd5cnRZazVnbFlZeUFrcTVqZklSXzRzWER5d19Qc1BWTW1aZ
XltenVGV3BQTzVFWlJYR0ZpRjFET0o4Q0Q3Z3Zta2dUdlBXSWpkemtBIn0=
m=audio 9 RTP/SAVPF 109 9 0 8
c=IN IP4 0.0.0.0
a=sendrecv
a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
a=ice-pwd:3969306b18d3deed8c120a8bdec02112
a=ice-ufrag:93f1515f
a=mid:sdparta_0
a=rtcp-mux
a=rtpmap:109 opus/48000/2
a=rtpmap:9 G722/8000/1
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=setup:actpass
a=ssrc:2076263852 cname:{074bac48-3b26-4f2c-9cf9-06021ca088be}
m=video 9 RTP/SAVPF 120 126 97
…
```

Base64decode (a=identity)


```
a=identity:eyJpZHAiOnsiZG9tYWluIjoibmlpZi5odSIsInByb3RvY29sIjoiaWRwLmh0bWwifSwiYXNzZXJ0aW9u
IjoiZXlKaGJHY2lPaUpTVXpJMU5pSXNJblI1Y0NJNklrcFhVeUo5LmV5SmpiMjUwWlc1MGN5STZleUptYVc1blpYSnd
jbWx1ZENJNlczc2lZV3huYjNKcGRHaHRJam9pYzJoaExUSTFOaUlzSW1ScFoyVnpkQ0k2SWprek9rTXdPakl6T2pKR0
9rRXlPakF3T2pBd09qQkVPalV4T2tGRE9rUXlPalUwT2pZMU9rWTBPak5DT2pkRU9qa3lPa1JET2pnNE9qTXpPalV4T
2pJek9qUXdPamN5T2preE9qZ3pPalZDT2pBeE9qSkdPalV3T2pjNE9qTkdJbjFkZlN3aWFXUmxiblJwZEhraU9pSnRh
WE5wUUc1cGFXWXVhSFVpZlEuSTVQdGhKNFFDT05TOFVXd25OOUh3MEdaTDl3d0RBVGRrTWtFWllmdlNVTTJ6Umd5R09
WSGgzRmpnc2FPZklkRnFsNUx6azBFbndVOTNQOUlCQ0xZOWtia3V1c0V1S25YRGVNLTNINWFmdTJvZl9CTlZjUnB3Mm
dBdlNBbVR6SlltcEpqMFEtdmVOTmQOUlCQ0xZOWtia3V1S25YRGVNLTNINWFmdTJvZl9CTlZjUnB3MmRnhoTTh5OHd
RnhoTTh5OHdrLWd5cnRZazVnbFlZeUFrcTVqZklSXzRzWER5d19Qc1BWTW1aZXltenVGV3BQTzVFWlJYR0ZpRjFET0o
4Q0Q3Z3Zta2dUdlBXSWpkemtBIn0=
```


Base64decoded identity:


```
{
  "idp": {
    "domain": "niif.hu",
    "protocol": "idp.html"
  },
  "assertion":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXUyJ9.eyJjb250ZW50cyI6eyJmaW5nZXJwcmludCI6W3siYWxnb3JpdGhtIj
oic2hhLTI1NiIsImRpZ2VzdCI6IjkzOkMwOjIzOjJGOkEyOjAwOjAwOjBEOjUxOkFDOkQyOjU0OjY1OkY0OjNCOjdEOj
kyOkRDOjg4OjMzOjUxOjIzOjQwOjcyOjkxOjgzOjVCOjAxOjJGOjUwOjc4OjNGIn1dfSwiaWRlbnRpdHkiOiJtaXNpQG
5paWYuaHUifQ.I5PthJ4QCONS8UWwnN9Hw0GZL9wwDATdkMkEZYfvSUM2zRgyGOVHh3FjgsaOfIdFql5Lzk0EnwU93P9
IBCLY9kbkuusEuKnXDeM-3H5afu2of_BNVcRpw2gAvSAmTzJYmpJj0Q-vetNkUOXndOG-
B3OvFot0eSD6VR5Gaol2pgnKqRNKNwwZpFuyFYlThtrHtcbO_VWz8BviM8JKnNtLVwRq5HL_fKNTB71CbJ2ZhyYu5PGp
X8WqrLX-rnnXHV7FxhM8y8wk-
gyrtYk5glYYyAkq5jfIR_4sXDyw_PsPVMmZeymzuFWpPO5EZRXGFiF1DOJ8CD7gvmkgTvPWIjdzkA"
}
```

JSON Web Token (JWT) decoded


```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXUyJ9.eyJjb250Z
W50cyI6eyJmaW5nZXJwcmludCI6W3siYWxnb3Jpd
GhtIjoic2hhLTI1NiIsImRpZ2VzdCI6IjkzOkMwOjIzOj
JGOkEyOjAwOjAwOjBEOjUxOkFDOkQyOjU0OjY1
OkY0OjNCOjdEOjkyOkRDOjg4OjMzOjUxOjIzOjQw
OjcyOjkxOjgzOjVCOjAxOjJGOjUwOjc4OjNGIn1dfS
wiaWRlbnRpdHkiOiJtaXNpQG5paWYuaHUifQ.I5Pth
J4QCONS8UWwnN9Hw0GZL9wwDATdkMkEZYfv
SUM2zRgyGOVHh3FjgsaOfIdFql5Lzk0EnwU93P9IB
CLY9kbkuusEuKnXDeM-
3H5afu2of_BNVcRpw2gAvSAmTzJYmpJj0Q-
vetNkUOXndOG-
B3OvFot0eSD6VR5Gaol2pgnKqRNKNwwZpFuyFYl
ThtrHtcbO_VWz8BviM8JKnNtLVwRq5HL_fKNTB7
1CbJ2ZhyYu5PGpX8WqrLX-rnnXHV7FxhM8y8wk-
gyrtYk5glYYyAkq5jfIR_4sXDyw_PsPVMmZeymzuF
WpPO5EZRXGFiF1DOJ8CD7gvmkgTvPWIjdzkA
```



```
{
 "alg": "RS256",
 "typ": "JWS"
}
```

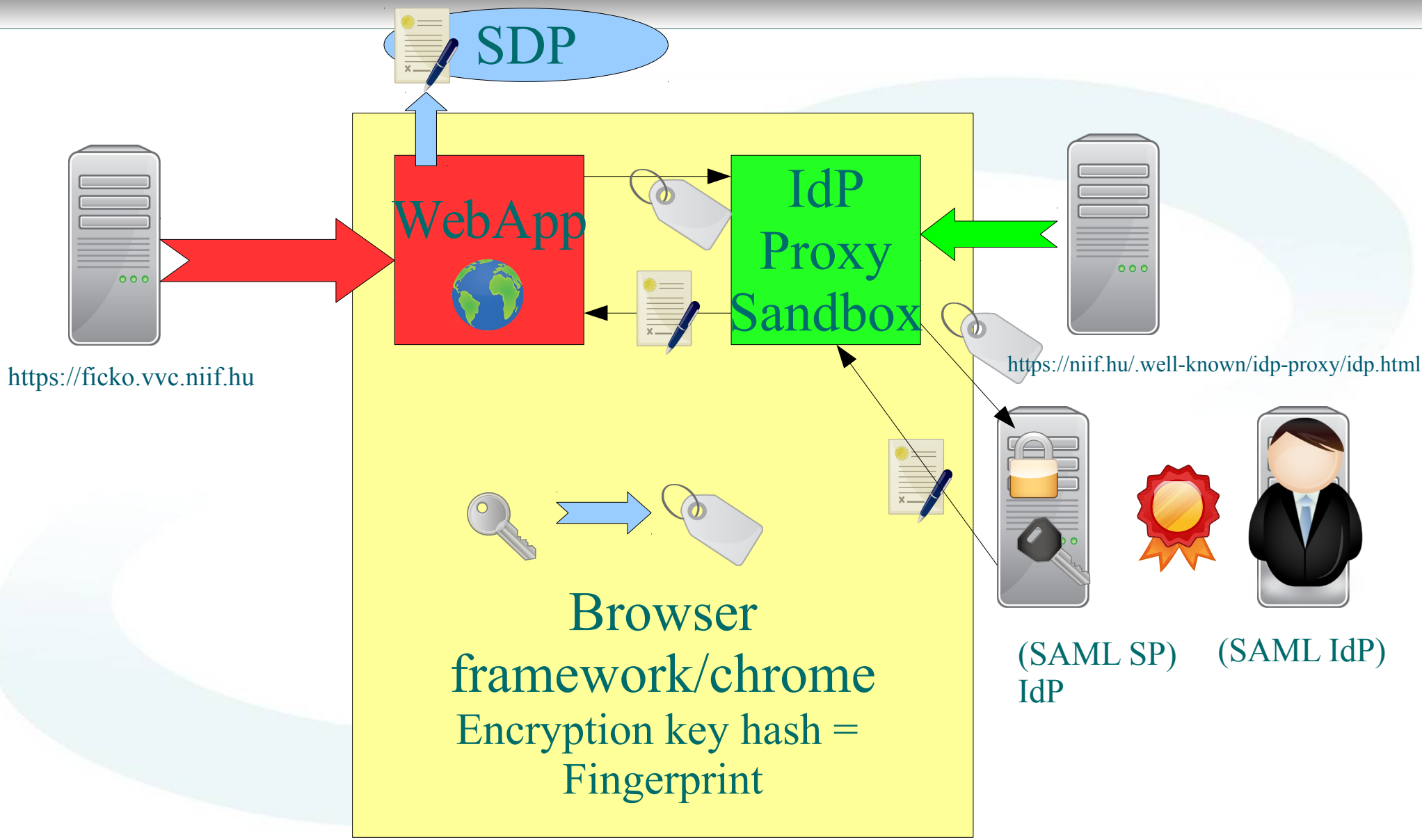


```
{
 "contents": {
  "fingerprint": [
   {
    "algorithm": "sha-256",
    "digest":
"93:C0:23:2F:A2:00:00:0D:51:AC:D2:54:65:F4:3B:7D:92:DC:88:33
:51:23:40:72:91:83:5B:01:2F:50:78:3F"
   }
  ]
 },
 "identity": "misi@niif.hu"
}
```

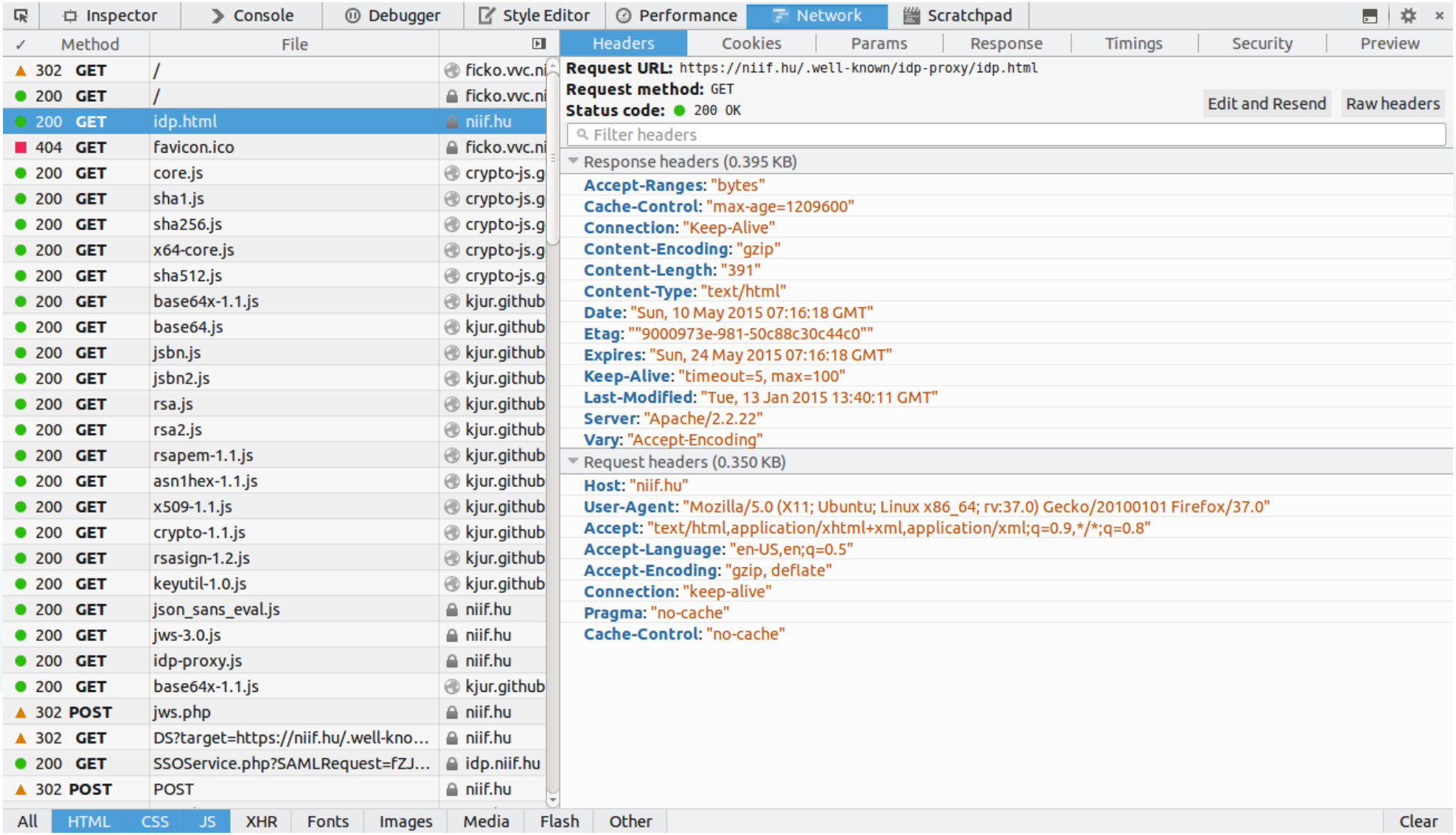

RSASHA256(
 base64UrlEncode(header) + "." +
 base64UrlEncode(payload),


```
157LM7FfPXkXMf6h18VA/RJqs049apX/x0GC4iJbaSl+Bf70uod0
Ualsk2IyNzkA
jc0VaaycjZIGTlm2MrQ4bzsnyBk3OwEh2l8ZaNnutIk3
-----END CERTIFICATE-----
```

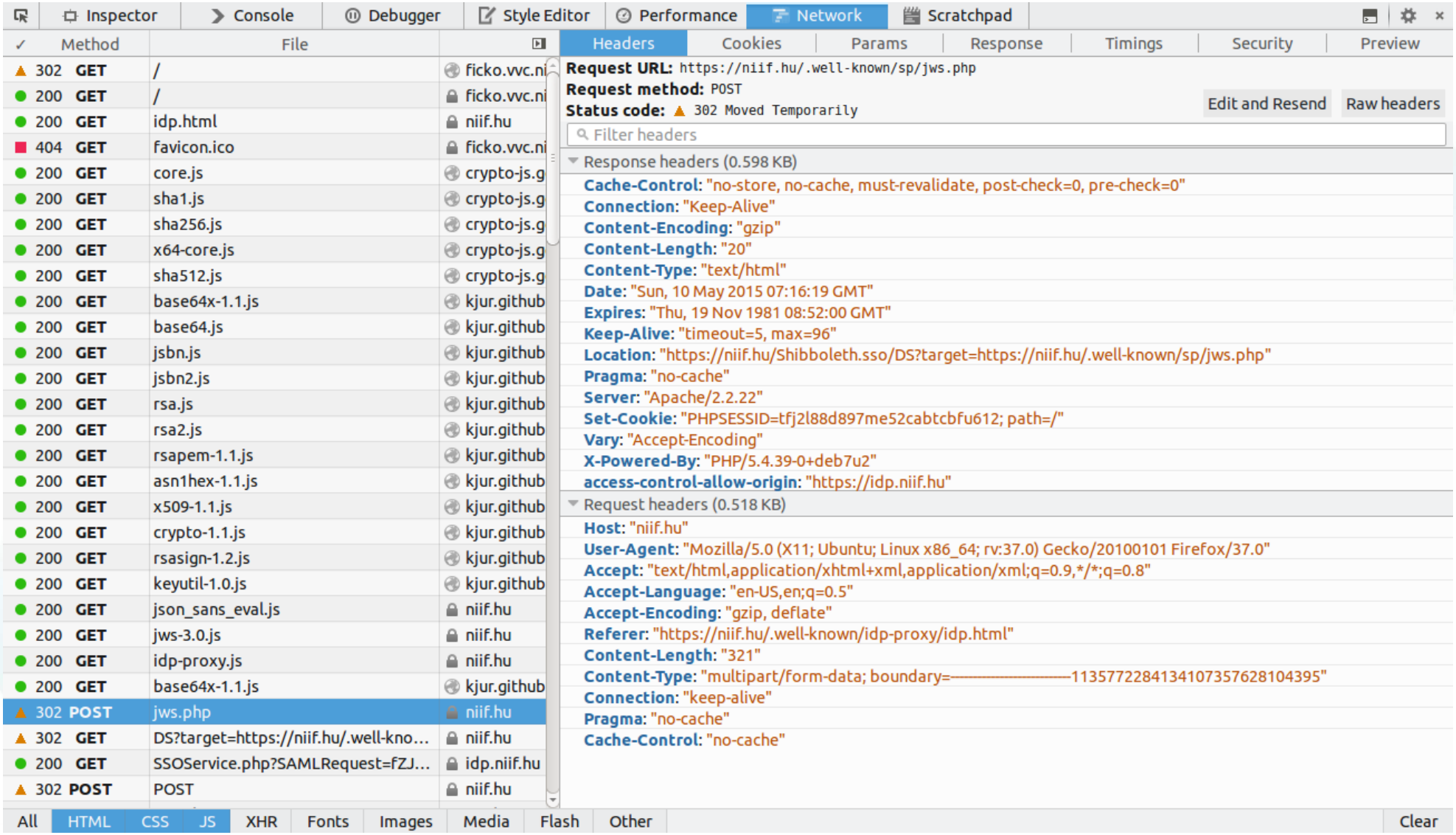
Bind/Sign (Identity assertion/certificate + Encryption key fingerprint/hash)



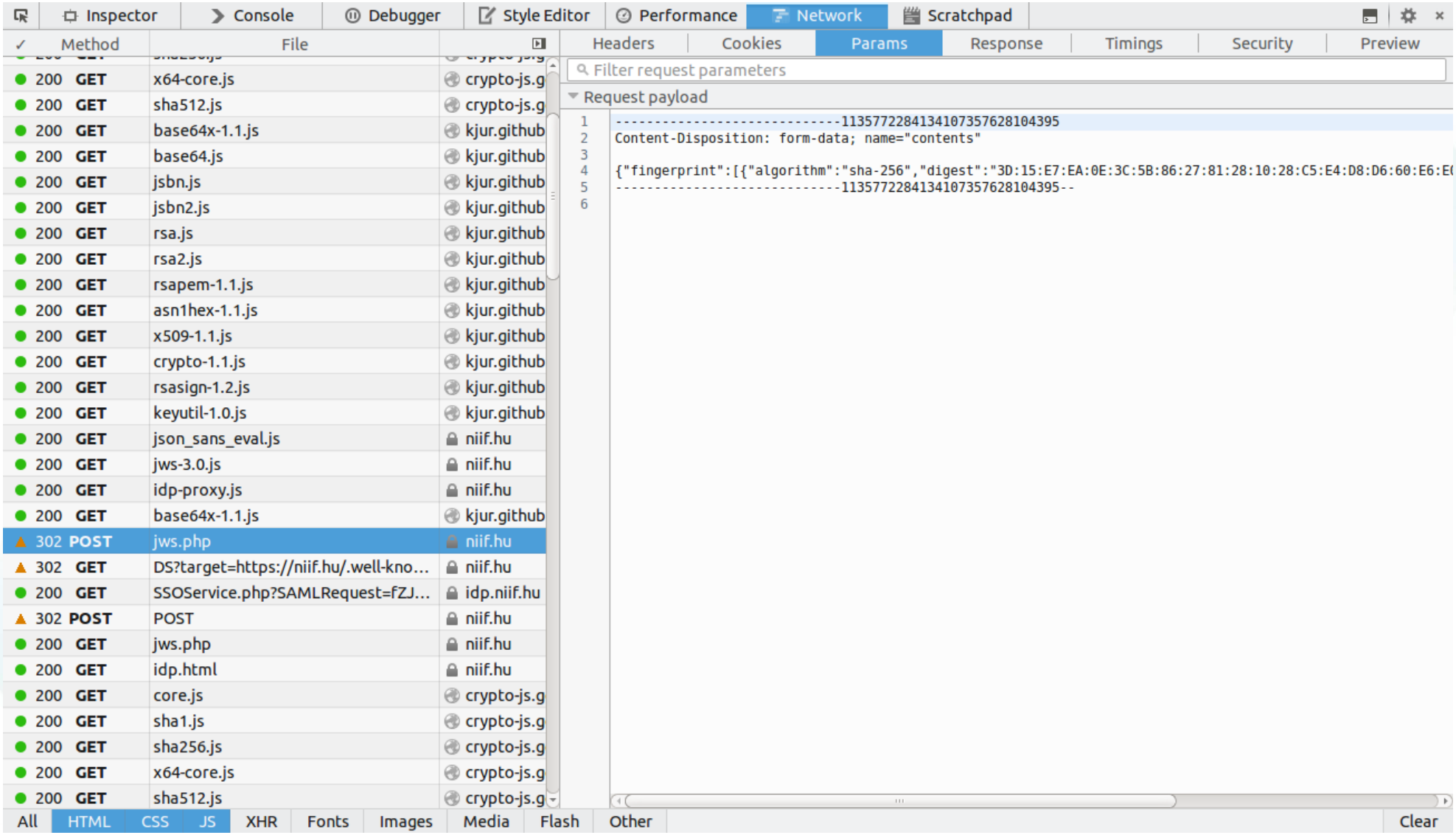
Load IdP proxy code



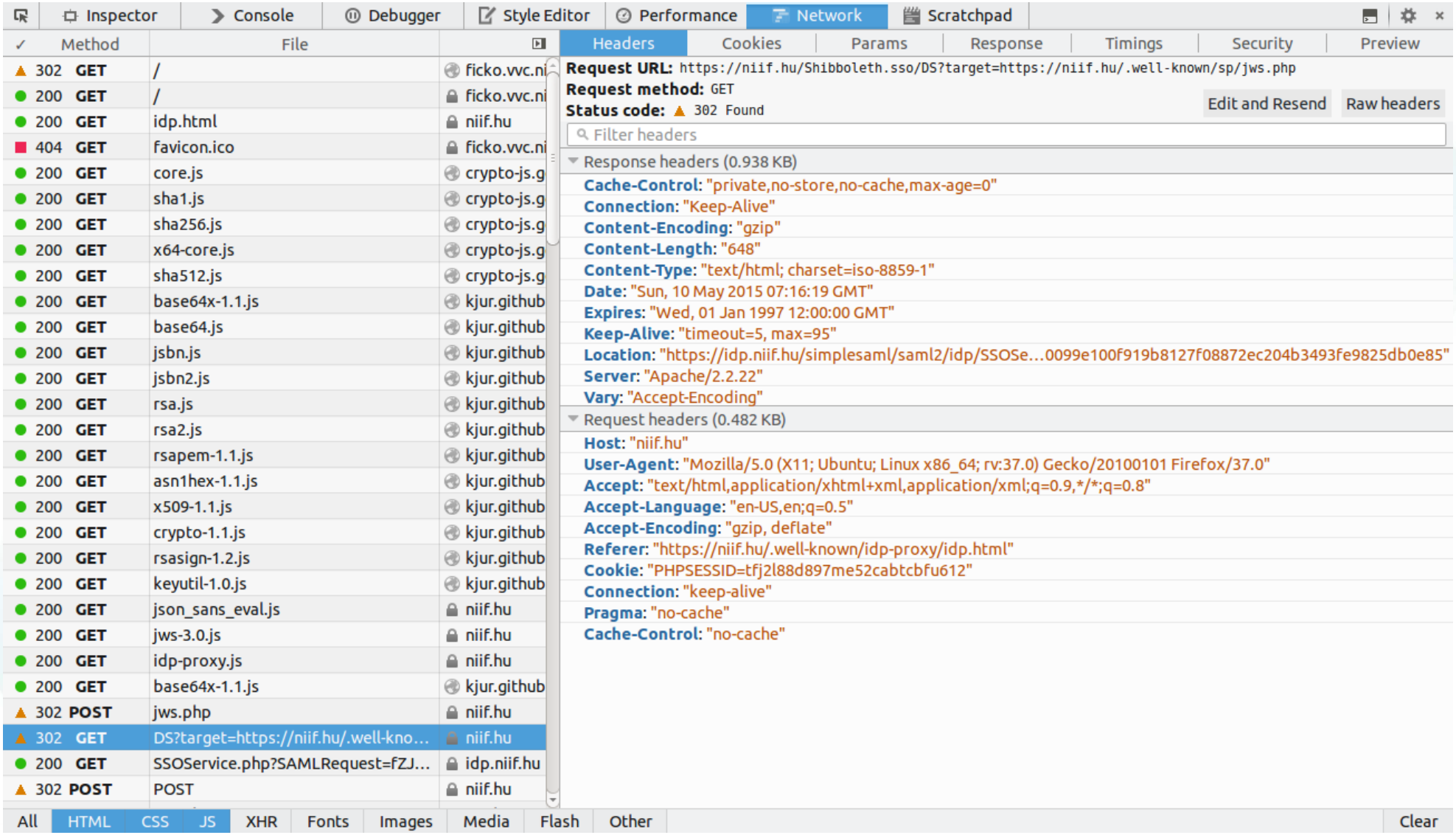
IdP proxy received SIGN request
new XMLHttpRequest to JWT signer service



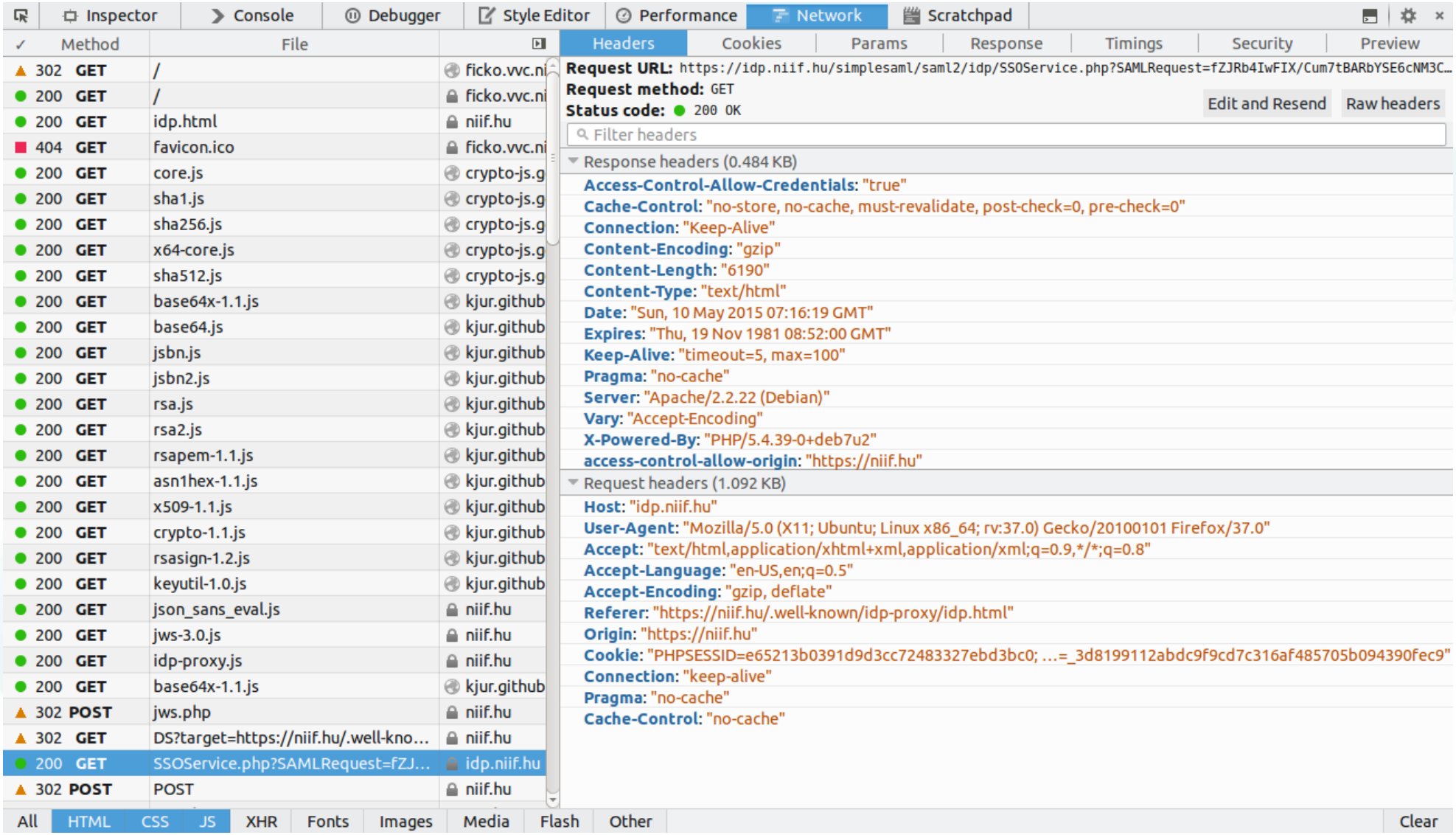
Send the fingerprint to sign



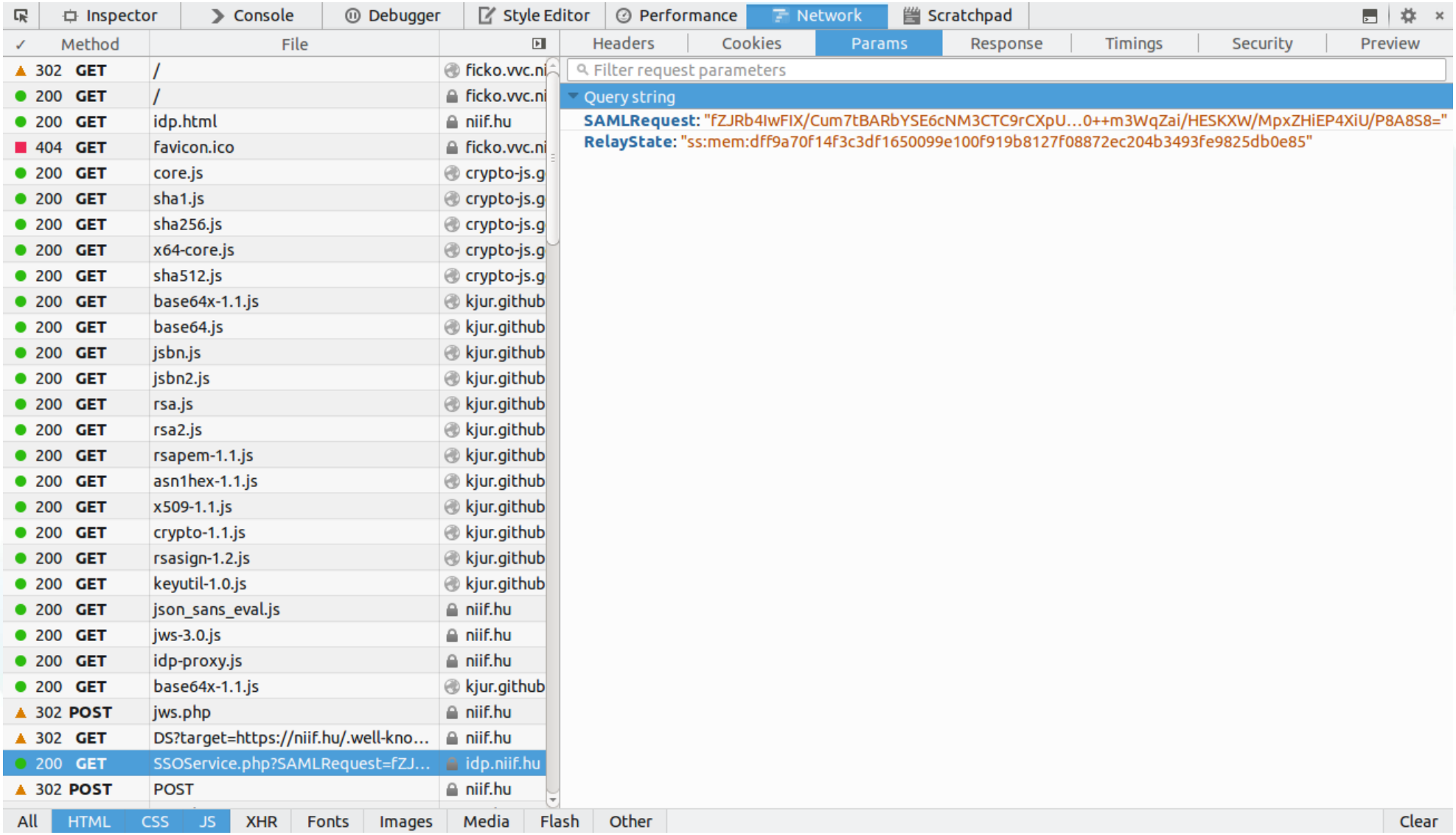
niif.hu DS redirect to NIIF IdP SSO



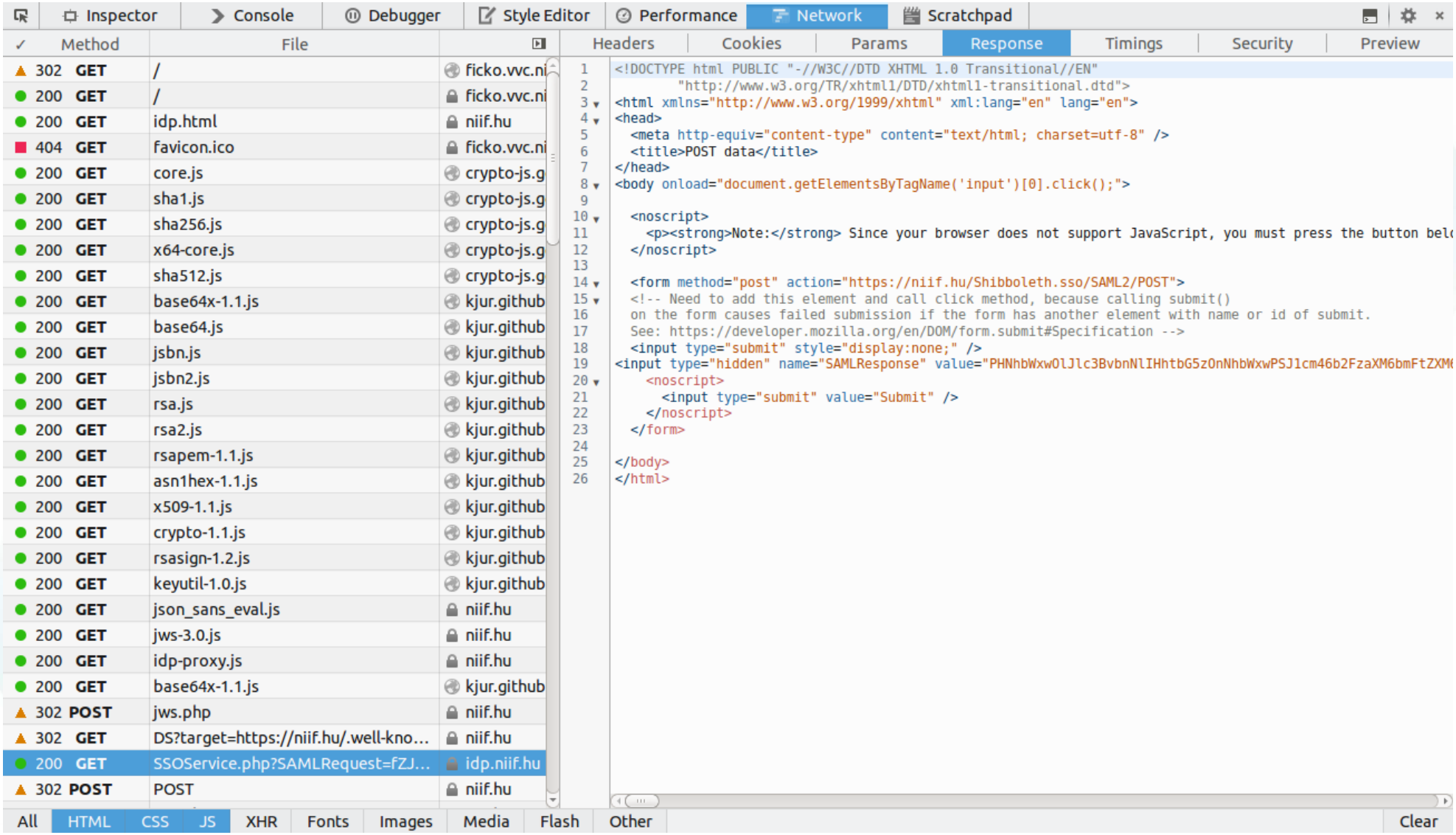
IdP SSO, CORS


Inspector
Console
Debugger
Style Editor
Performance
Network
Scratchpad
Method
File
Headers
Cookies
Params
Response
Timings
Security
Preview
302 GET /
200 GET /
200 GET idp.html
404 GET favicon.ico
200 GET core.js
200 GET sha1.js
200 GET sha256.js
200 GET x64-core.js
200 GET sha512.js
200 GET base64x-1.1.js
200 GET base64.js
200 GET jsbn.js
200 GET jsbn2.js
200 GET rsa.js
200 GET rsa2.js
200 GET rsapem-1.1.js
200 GET asn1hex-1.1.js
200 GET x509-1.1.js
200 GET crypto-1.1.js
200 GET rsasign-1.2.js
200 GET keyutil-1.0.js
200 GET json_sans_eval.js
200 GET jws-3.0.js
200 GET idp-proxy.js
200 GET base64x-1.1.js
302 POST jws.php
302 GET DS?target=https://niif.hu/.well-kno...
200 GET SSOService.php?SAMLRequest=fZJ...
302 POST POST
Request URL: https://idp.niif.hu/simplesaml/saml2/idp/SSOService.php?SAMLRequest=fZJRb4IwFIX/Cum7tBARbYSE6cNM3C...
Request method: GET
Status code: 200 OK
Edit and Resend
Raw headers
Filter headers
Response headers (0.484 KB)
Access-Control-Allow-Credentials: "true"
Cache-Control: "no-store, no-cache, must-revalidate, post-check=0, pre-check=0"
Connection: "Keep-Alive"
Content-Encoding: "gzip"
Content-Length: "6190"
Content-Type: "text/html"
Date: "Sun, 10 May 2015 07:16:19 GMT"
Expires: "Thu, 19 Nov 1981 08:52:00 GMT"
Keep-Alive: "timeout=5, max=100"
Pragma: "no-cache"
Server: "Apache/2.2.22 (Debian)"
Vary: "Accept-Encoding"
X-Powered-By: "PHP/5.4.39-0+deb7u2"
access-control-allow-origin: "https://niif.hu"
Request headers (1.092 KB)
Host: "idp.niif.hu"
User-Agent: "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:37.0) Gecko/20100101 Firefox/37.0"
Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
Accept-Language: "en-US,en;q=0.5"
Accept-Encoding: "gzip, deflate"
Referer: "https://niif.hu/.well-known/idp-proxy/idp.html"
Origin: "https://niif.hu"
Cookie: "PHPSESSID=e65213b0391d9d3cc72483327ebd3bc0; ...=_3d8199112abdc9f9cd7c316af485705b094390fec9"
Connection: "keep-alive"
Pragma: "no-cache"
Cache-Control: "no-cache"
All
HTML
CSS
JS
XHR
Fonts
Images
Media
Flash
Other
Clear

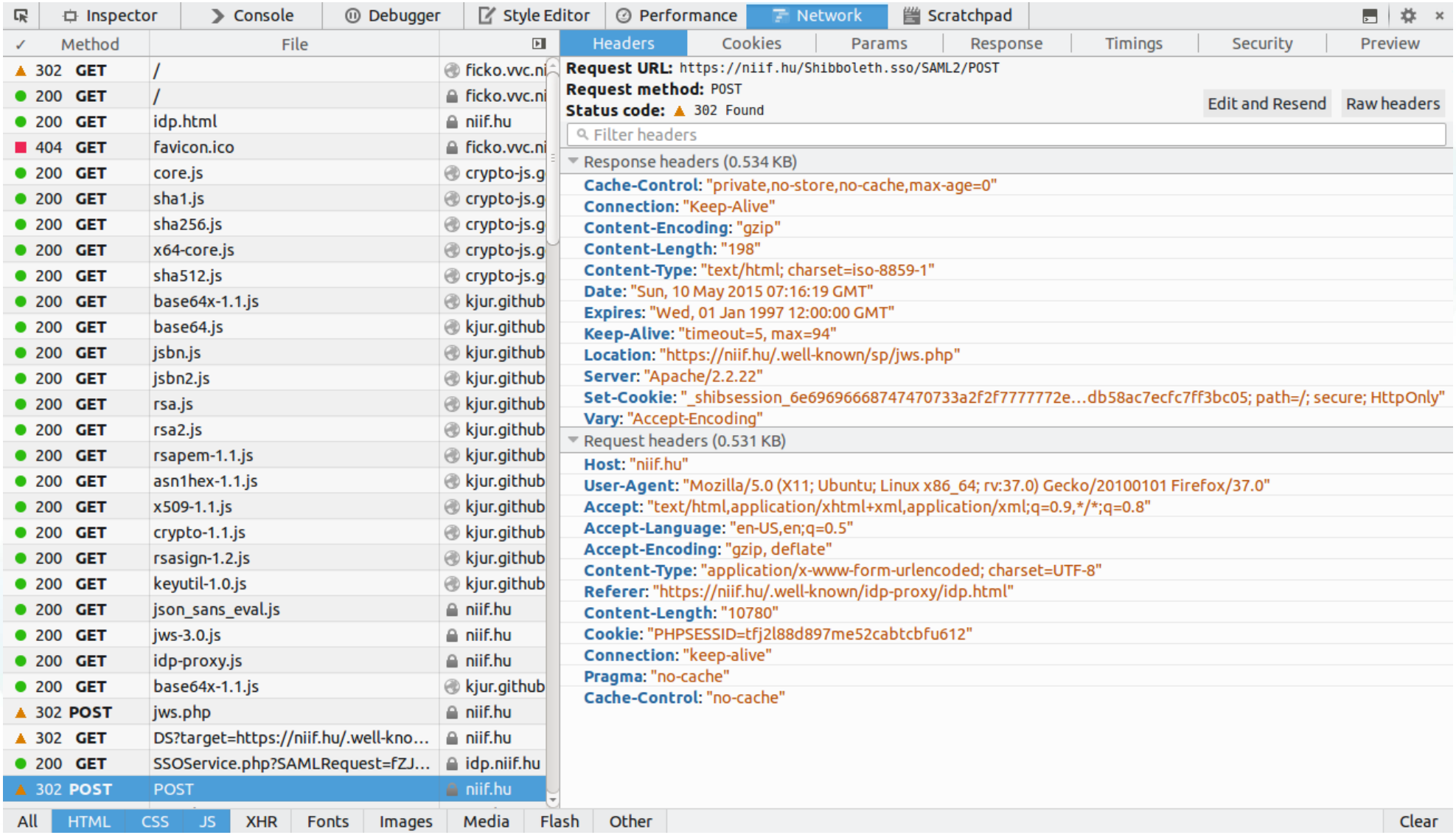
SAML SSO request parameters



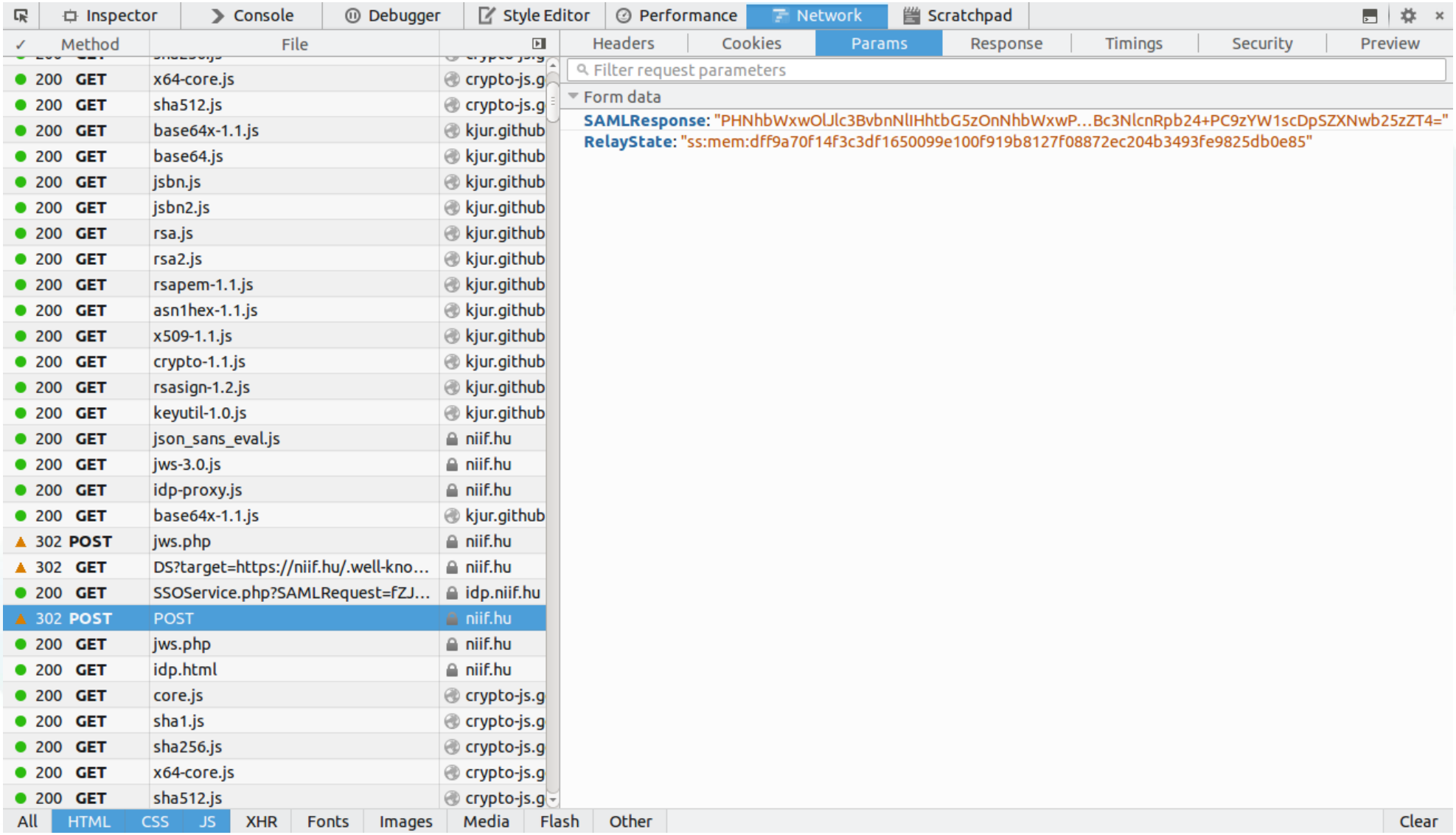
IdP SSO SAMLResponse



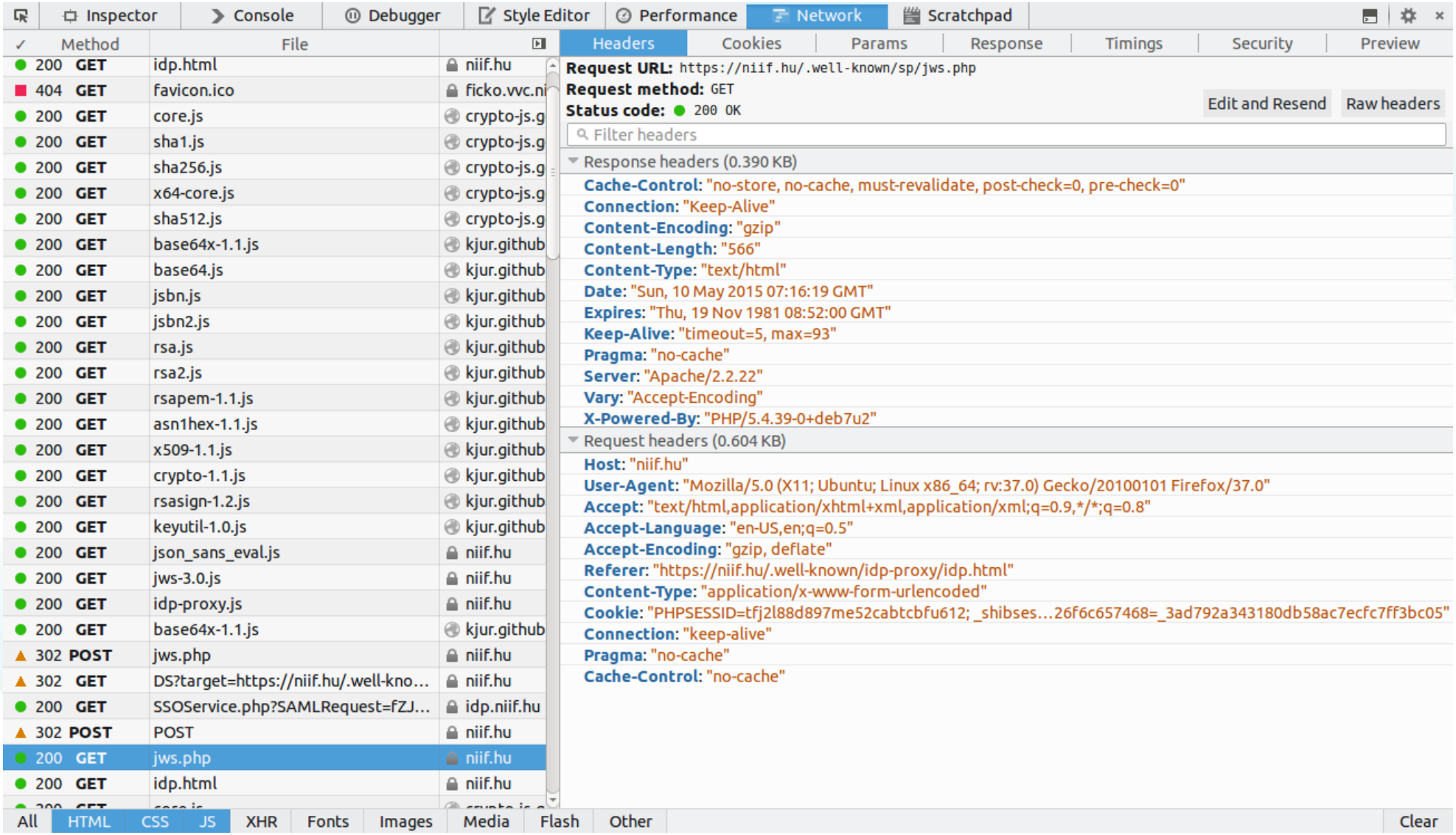
Post IdP SSO SAMLResponse to niif.hu SP



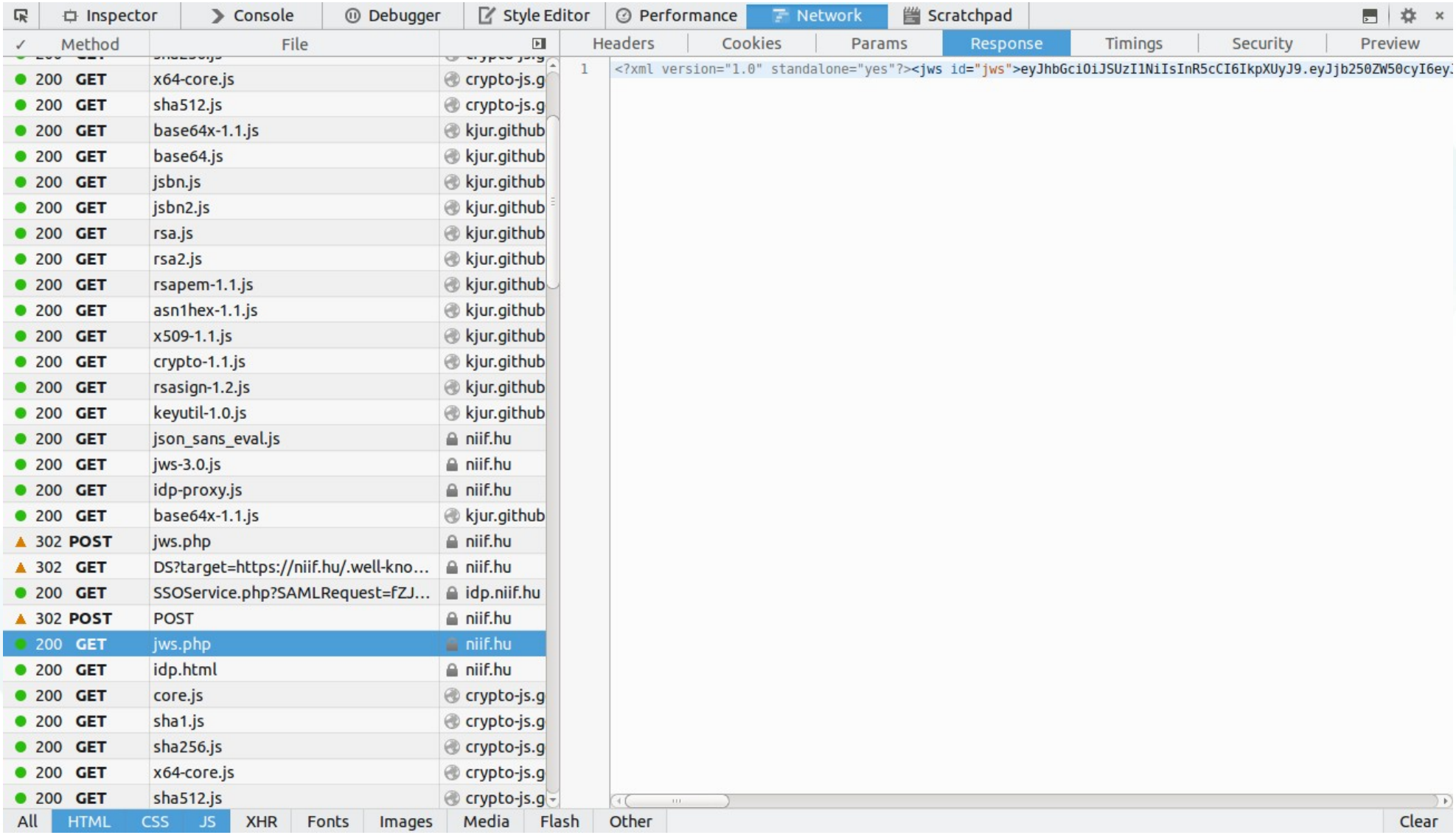
Parameters: SAMLResponse, RelayState



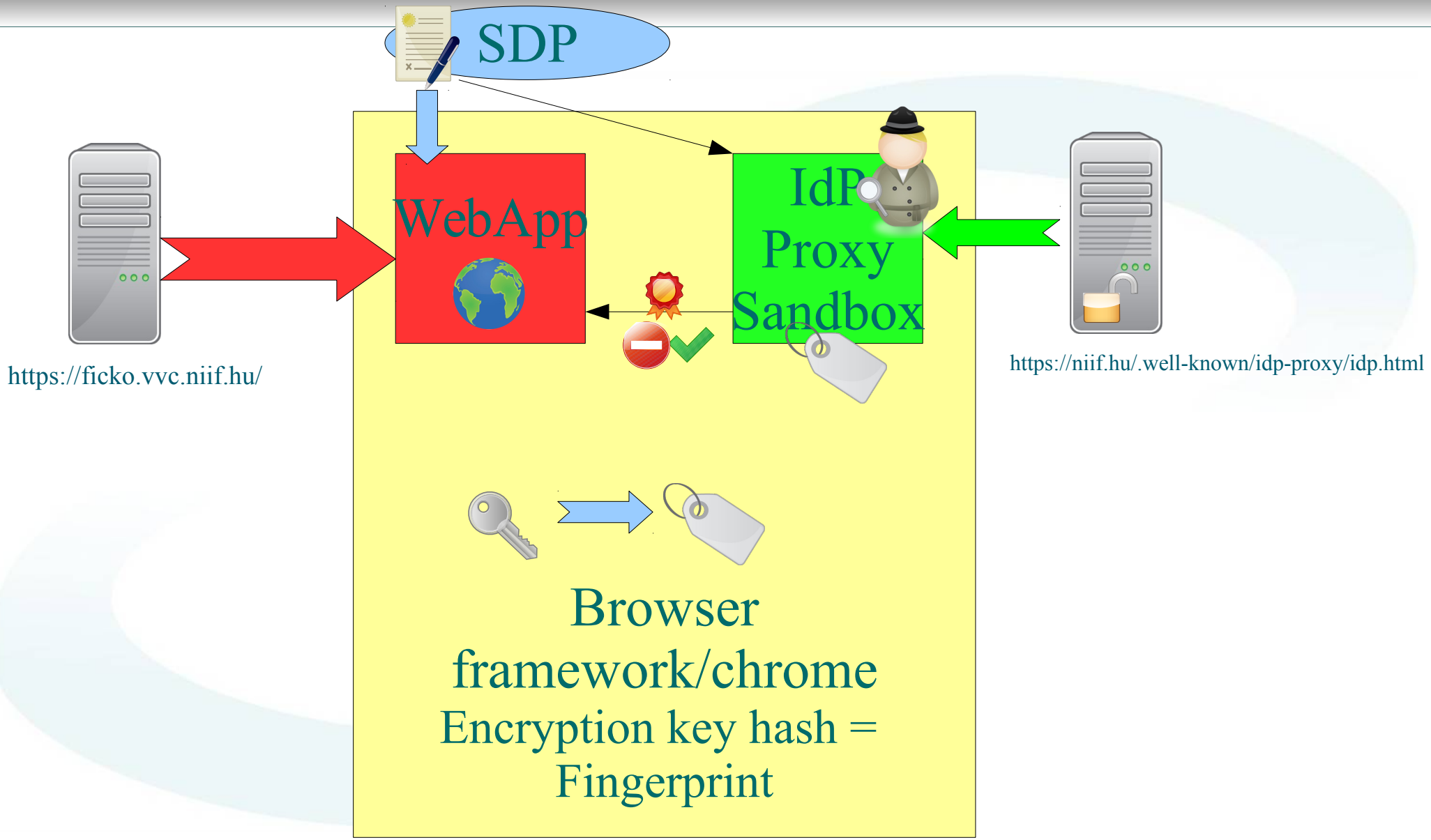
Redirect finally to the original request



Get the final signed JWT/JWS



Verify (Identity assertion/certificate + Encryption key fingerprint/hash)



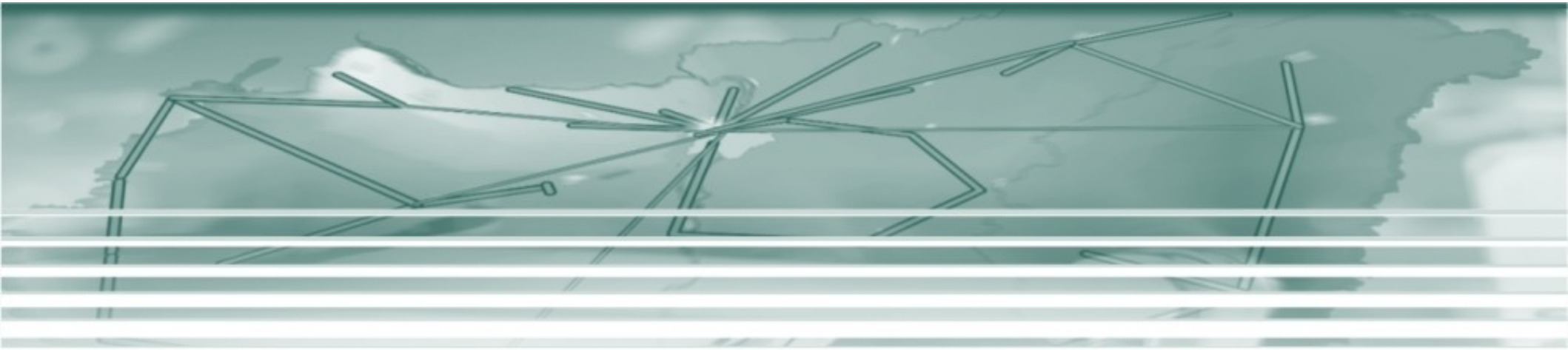
Assumptions and Limitations

- Limitations for SAML federation
  - User needs to have a valid session with his home IdP.
  - Automatic SAML attribute consent needed for automatic SP login (niif.hu)
  - SSO on IdP/SP (identity domain SP)
  - CORS on IdP side is needed.
    - Consequence of XMLHttpRequest usage for signing JWT.
  - All web server that serves any identity domain part
    - must have been deployed the identity-proxy code
    - and must have function as SAML SP.

Summary & Takeaways

- We have trusted World Wide federated AAI service deployed. There is a demand to use this already in place service with WebRTC for RTC.
- WebRTC Identity is still a moving target.
  - Security model is in draft phase. Not everybody is happy with it.
- Implementation issues
  - Lack of presentation of the verified Identity in the browser Chrome/Framework in current Firefox 37.0.2 implementation.
  - WebRTC identity lack of implementation Browsers
    - Critical Mass needed.
    - More interest needed from Web Application developers to implement it.
  - Only one attribute the Identity is handled and provided by the IdP-Proxy.
    - Lack of support for more than the identity attribute:
      More attributes needed like picture URL, entitlements, group membership etc. to support more sophisticated Authentication and call accept/reject decision.
- As I proved in my demonstration, the usage of a SAML based Identity federation as WebRTC Identity Provider is pretty complicated, but it is not completely impossible!

Thank You!



Big thanks to Gyufi, Kristóf, Sitya (NIIF AAI Team)
for their continuous help and support!

References

- Demo: https://youtu.be/aeXaWDNU_sg
- WebRTC
  - https://tools.ietf.org/html/draft-ietf-rtcweb-security-arch-11
  - http://w3c.github.io/webrtc-pc/#identity
  - http://www.ietf.org/proceedings/82/slides/rtcweb-13.pdf
- SAML
  - https://technical.edugain.org/status.php
  - https://refeds.org/resources/
  - http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0-cd-02.html
  - http://en.wikipedia.org/wiki/SAML_2.0
- JWT
  - http://kjur.github.io/jsjws/
  - https://github.com/ritou/php-Akita_JOSE
  - https://code.google.com/p/json-sans-eval/
  - http://jwt.io/