OWASP Belgium
Chapter

The OWASP
Foundation
http://www.owasp.org

The Belgian e-ID: hacker vs developer

Erwin Geirnaert
ZION SECURITY

Frank Cornelis

Fedict

Agenda

- eID Card Introduction
- eID Card Integration
- Examples of bad implementations

Who are we?

- Frank: eID Architect at Fedict
  - eID Middleware
  - eID Applet
  - eID Trust Service, jTrust, eID IdP, eID DSS
- Erwin: whitehat hacker at ZION SECURITY

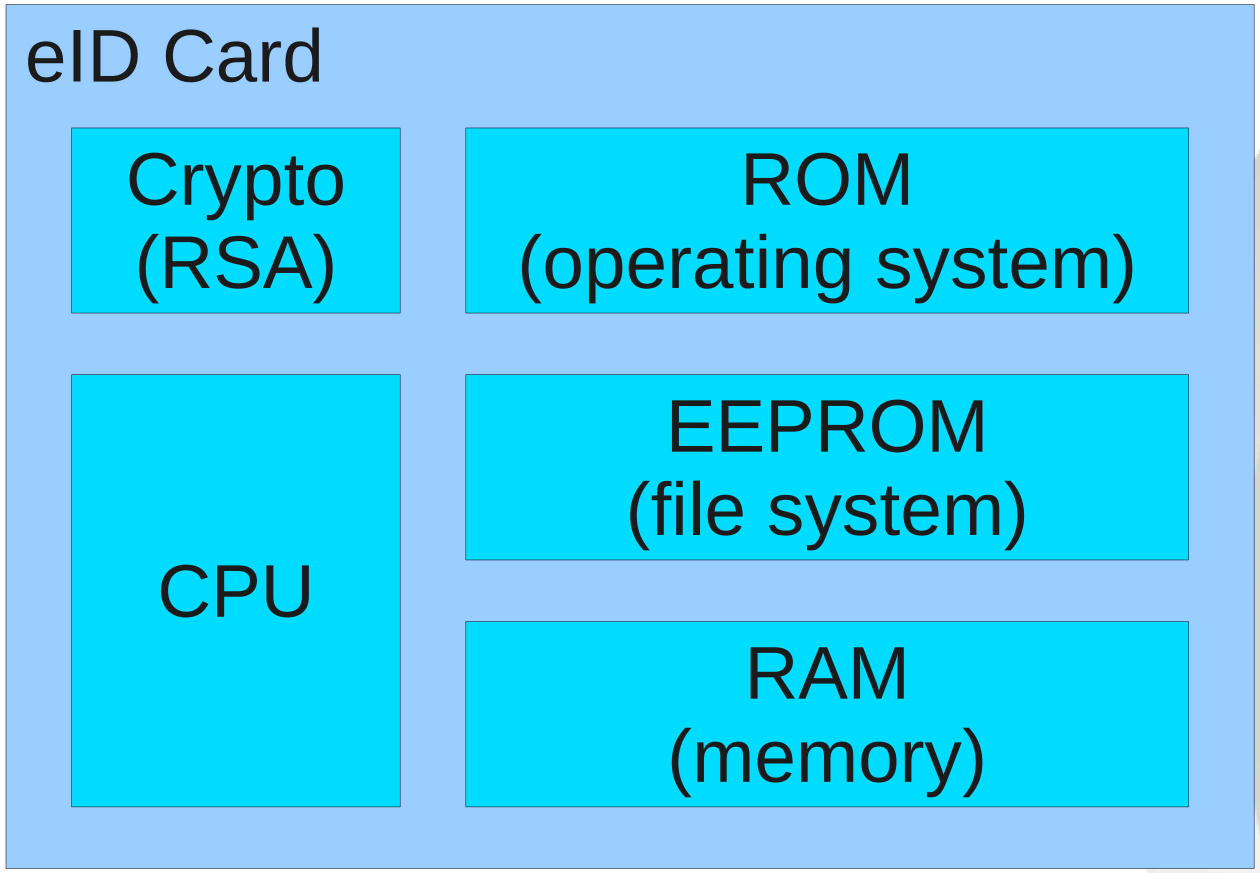
eID Card Introduction

eID in Belgium

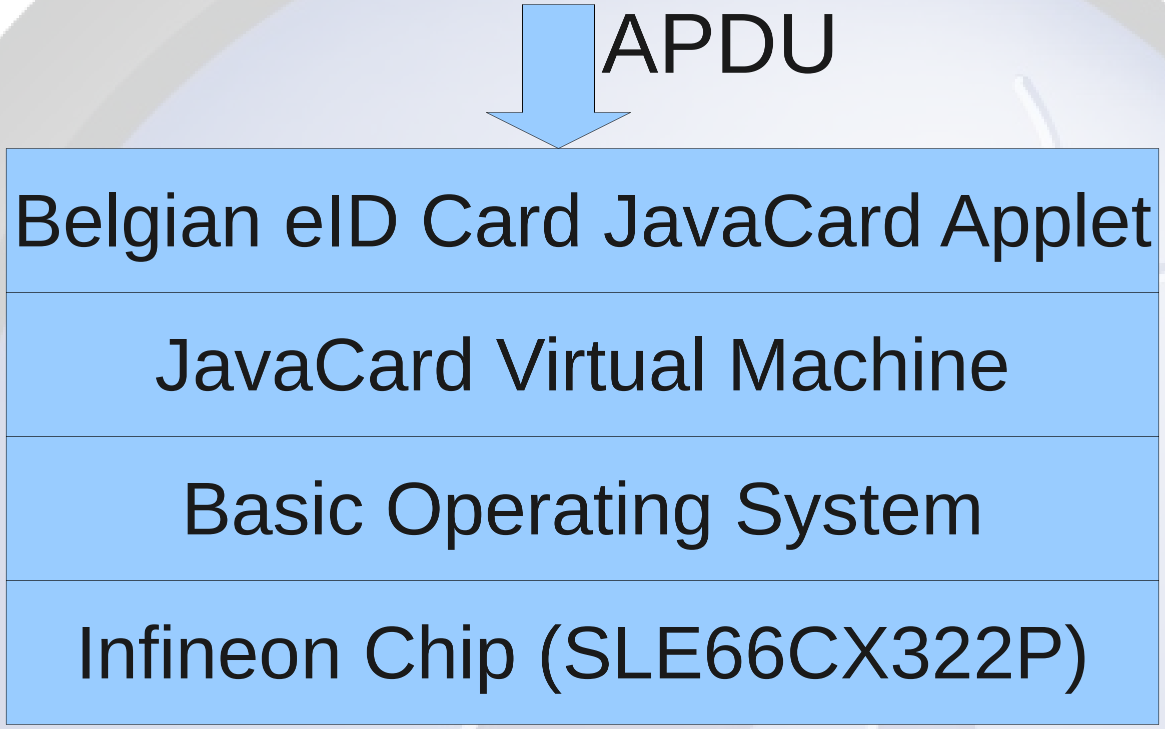
- eID cards issued (16/01/2010)
  - 8.220.456 citizen eID cards (full deployment)
  - 511.774 foreigner eID cards
  - 186.011 kids eID cards
- Technology
  - RSA 1024 smart card
  - QC with 5 year validity
- Involved major organizations:
  - FedICT: Federal ICT
    - PKI, client software, SOA solutions
  - National Registry
    - user database, card issuing

eID Card

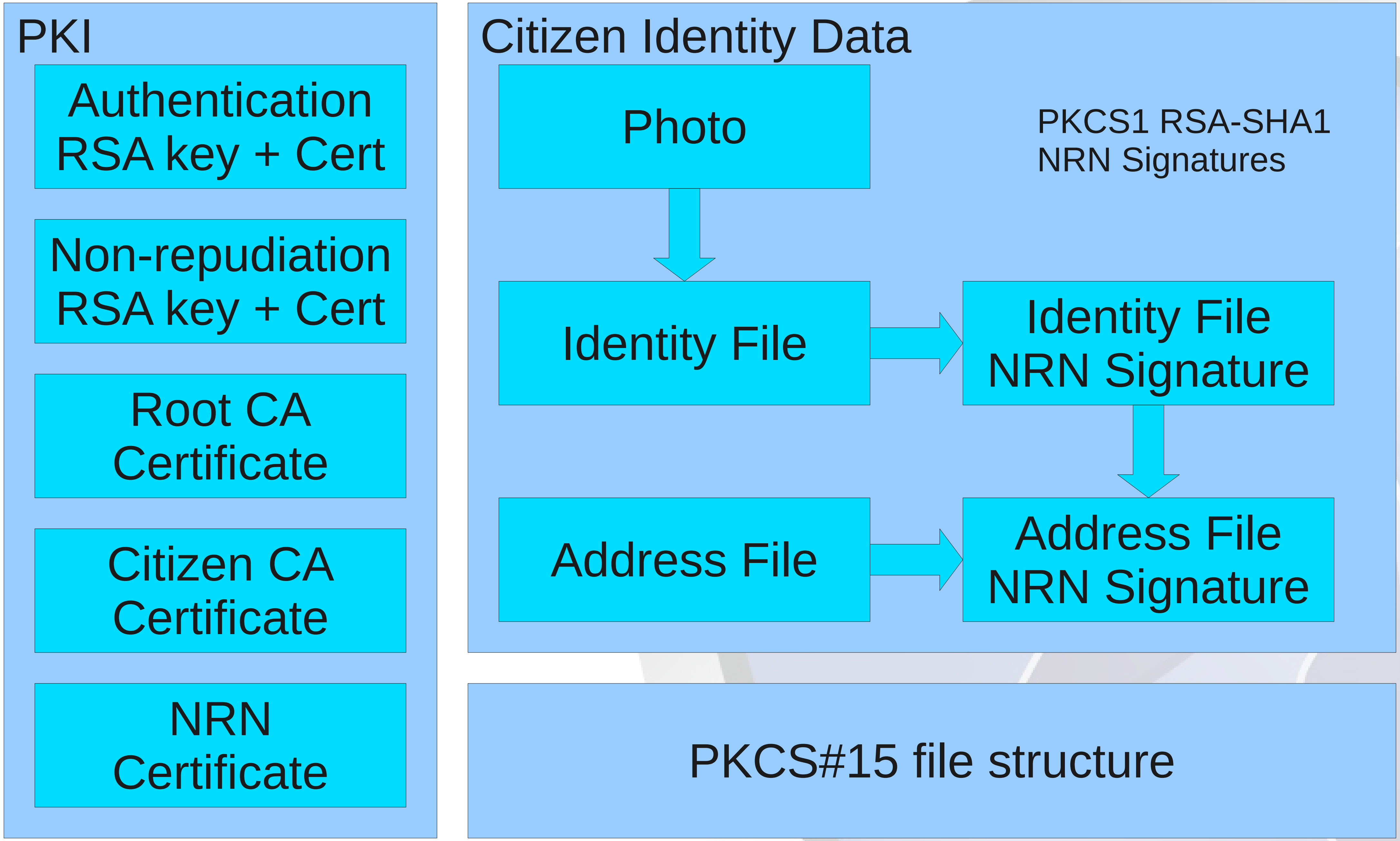
Physical Structure



Logical Structure



eID Card Content



PKI

Authentication RSA key + Cert

Non-repudiation RSA key + Cert

Root CA Certificate

Citizen CA Certificate

NRN Certificate

Citizen Identity Data

Photo

PKCS1 RSA-SHA1
NRN Signatures

Identity File

Identity File NRN Signature

Address File

Address File NRN Signature

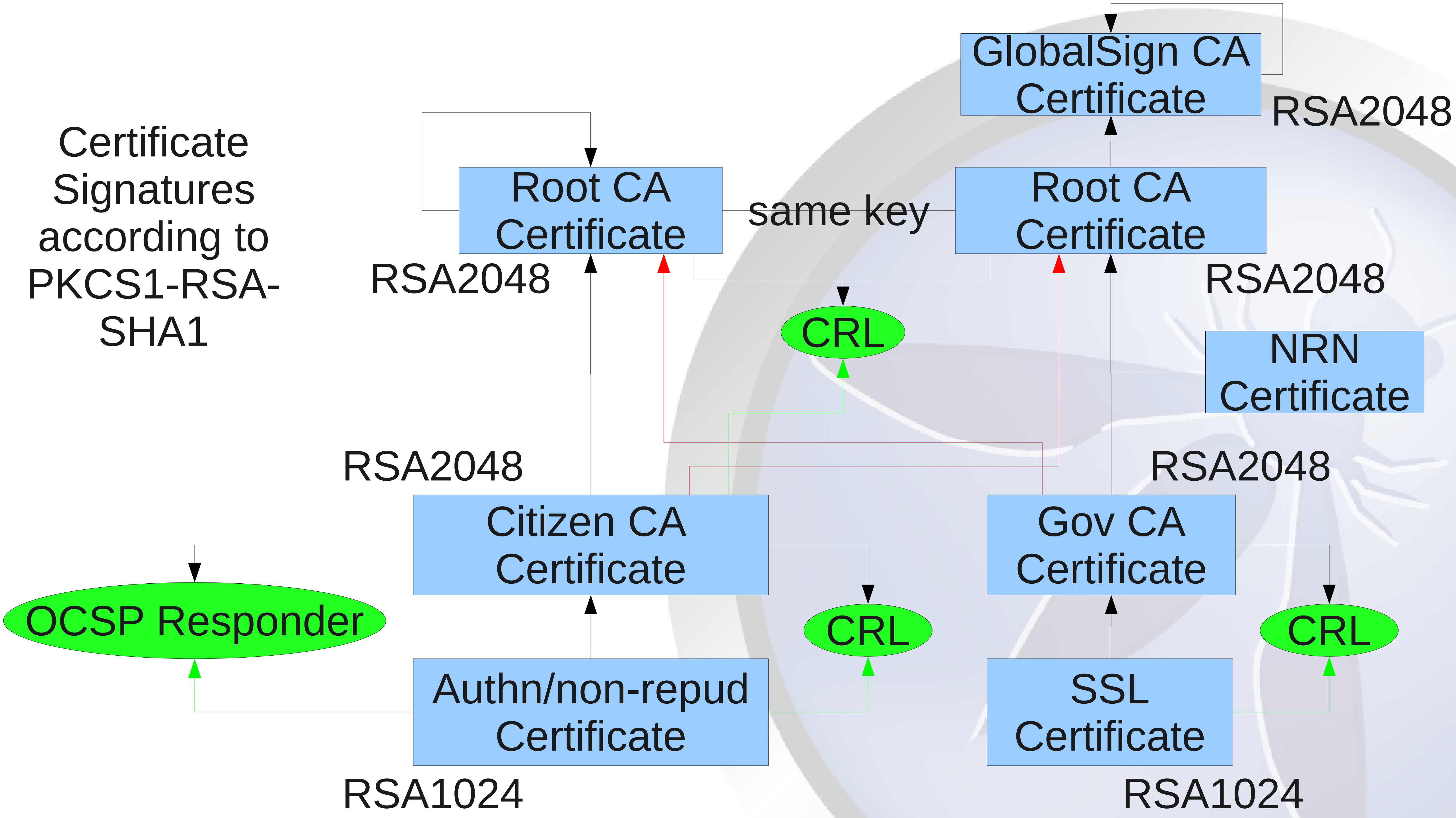
PKCS#15 file structure

eID Card Functionality

- Non-electronic functionality
  - Visible Identification
  - Visible Authentication via facial recognition and/or hand-written signature (a kind of challenge)
- Electronic functionality
  - Identification: who are you?
    - Passive eID usage
    - Privacy sensitive
  - Authentication: can you prove who you are?
    - Active challenging the eID card + eID user (2 factor)
  - Digital Signing: non-repudiation signature
    - Prove of acknowledgment at a point in time

eID PKI Topology

Certificate Signatures according to PKCS1-RSA-SHA1



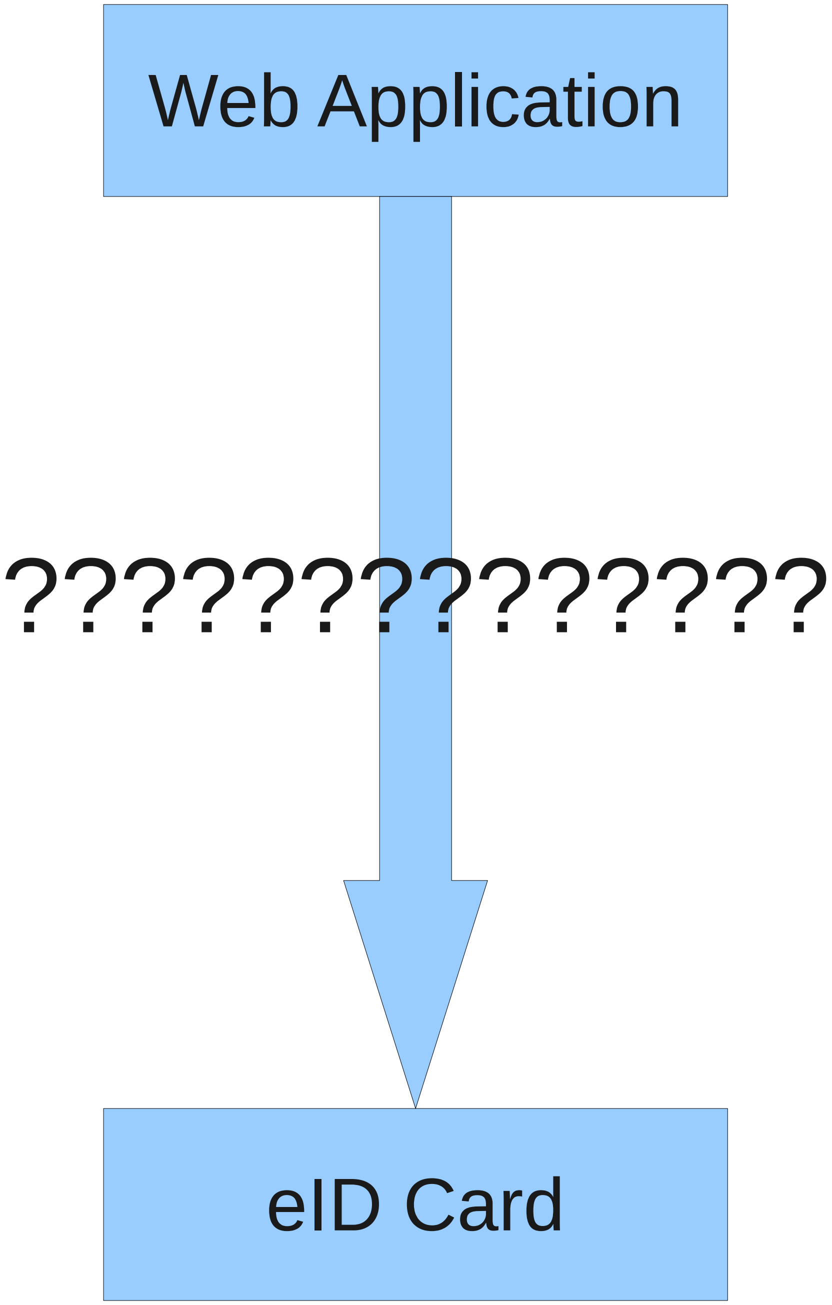
eID Card Integration

eID Card Interfacing

- Via APDU messages
  - Application Protocol Data Unit
  - Command: from the reader to the card
  - Response: from the card to the reader
- Example: Creation of a signature
  - Set APDU: select the key. 0x82 = authn key
  - Prepare DigestInfo PKCS1 DER sequence
  - Verify PIN APDU: (PIN BCD encoded)
  - Compute Digital Signature APDU
  - Retrieve signature data
- Doing the APDU interface is crazy

eID Card
APDU Demo

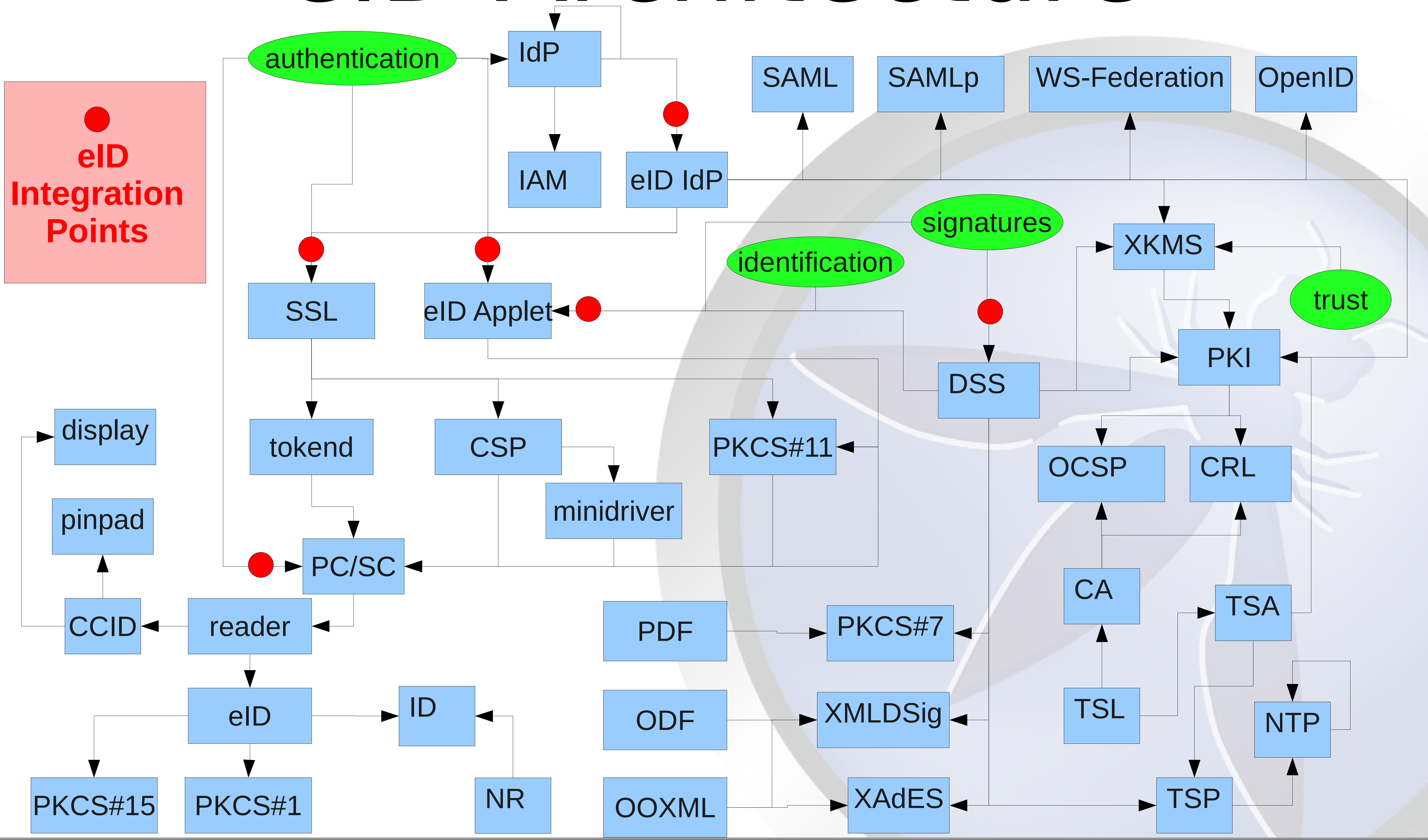
eID Card Integration



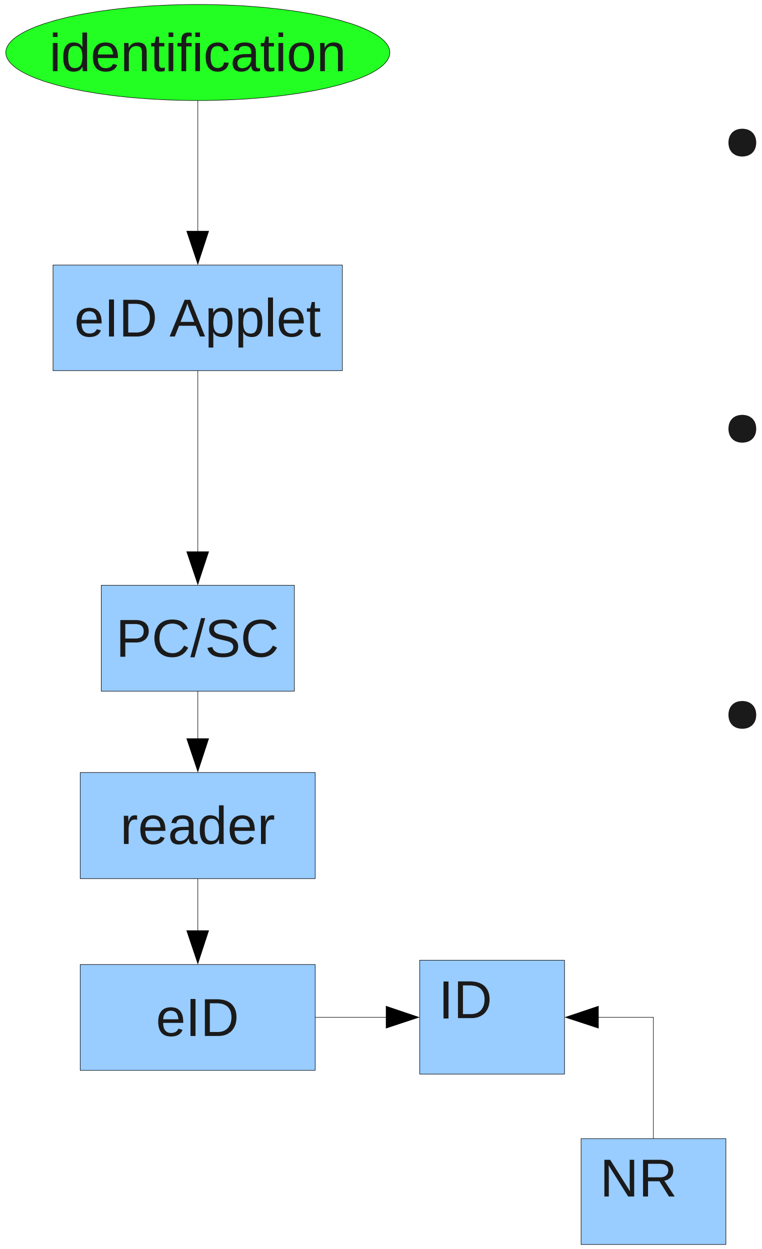
Important aspects when integrating:

- Ease of integration
- Secure usage of eID
- Platform independent solution:
  - Windows
  - Linux
  - Mac OS X
- Multiple browser support:
  - Firefox
  - MS IE
  - Safari
  - Chrome
- Integration point abstraction level
- Idiot proof eID components

eID Architecture



eID Identification

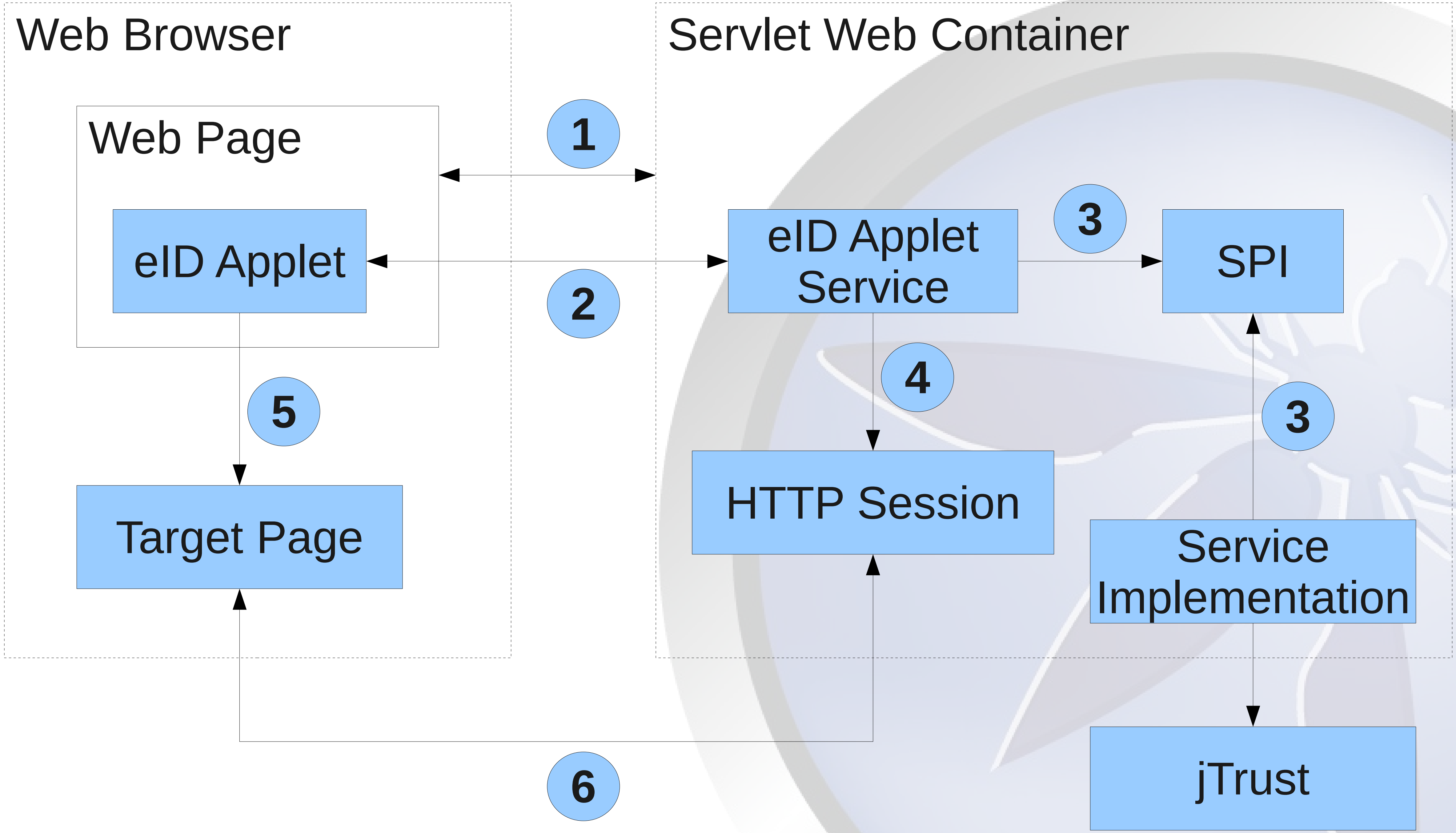


- Identification: who are you?
- Readout of the eID Identity, Address and Photo files.
- Server-side identity file parsing
- Server-side integrity validation of all identity files is possible via NRN signature

eID Applet

- Java 6 Web Browser eID Component
- Supports multiple web browsers:
  - Firefox, Chrome, Internet Explorer, Safari
- Platform-independent:
  - Windows, Linux, Mac OS X
- Interactive eID card handling
- Support for secure CCID pinpad readers
- Web developer friendly
- Open Source Software: GNU LGPL 3
  - http://code.google.com/p/eid-applet/

eID Applet Architecture



SPI design pattern is key to enabling eID Applet features

eID Applet Identification

identify-the-user.html

```
<script src="https://www.java.com/js/deployJava.js"></script>
<script>
    var attributes = {
        code :'be.fedict.eid.applet.Applet.class',
        archive :'eid-applet.jar',
        width :600,
        height :300
    };
    var parameters = {
        TargetPage :'identification-result-page.jsp',
        AppletService :'applet-service',
    };
    var version = '1.6';
    deployJava.runApplet(attributes, parameters, version);
</script>
```

identification-result-page.jsp

```
<%@page import="be.fedict.eid.applet.service.Identity"%>
<html>
<body>
    <%=((Identity) session.getAttribute("eid.identity")).name%>
</body>
</html>
```

web.xml

```
<servlet>
    <servlet-name>AppletServiceServlet</servlet-name>
    <servlet-class>be.fedict.eid.applet.service.AppletServiceServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>AppletServiceServlet</servlet-name>
    <url-pattern>/applet-service</url-pattern>
</servlet-mapping>
```

eID Identification Demo

eID Identity Integrity

- eID Applet reads the identity files and NRN signatures, posts to eID Applet Service
- Server-side verification of the NRN signatures over the identity, photo and address files by the eID Applet Service
- Verification of the NRN certificate chain via the IdentityIntegrityService SPI

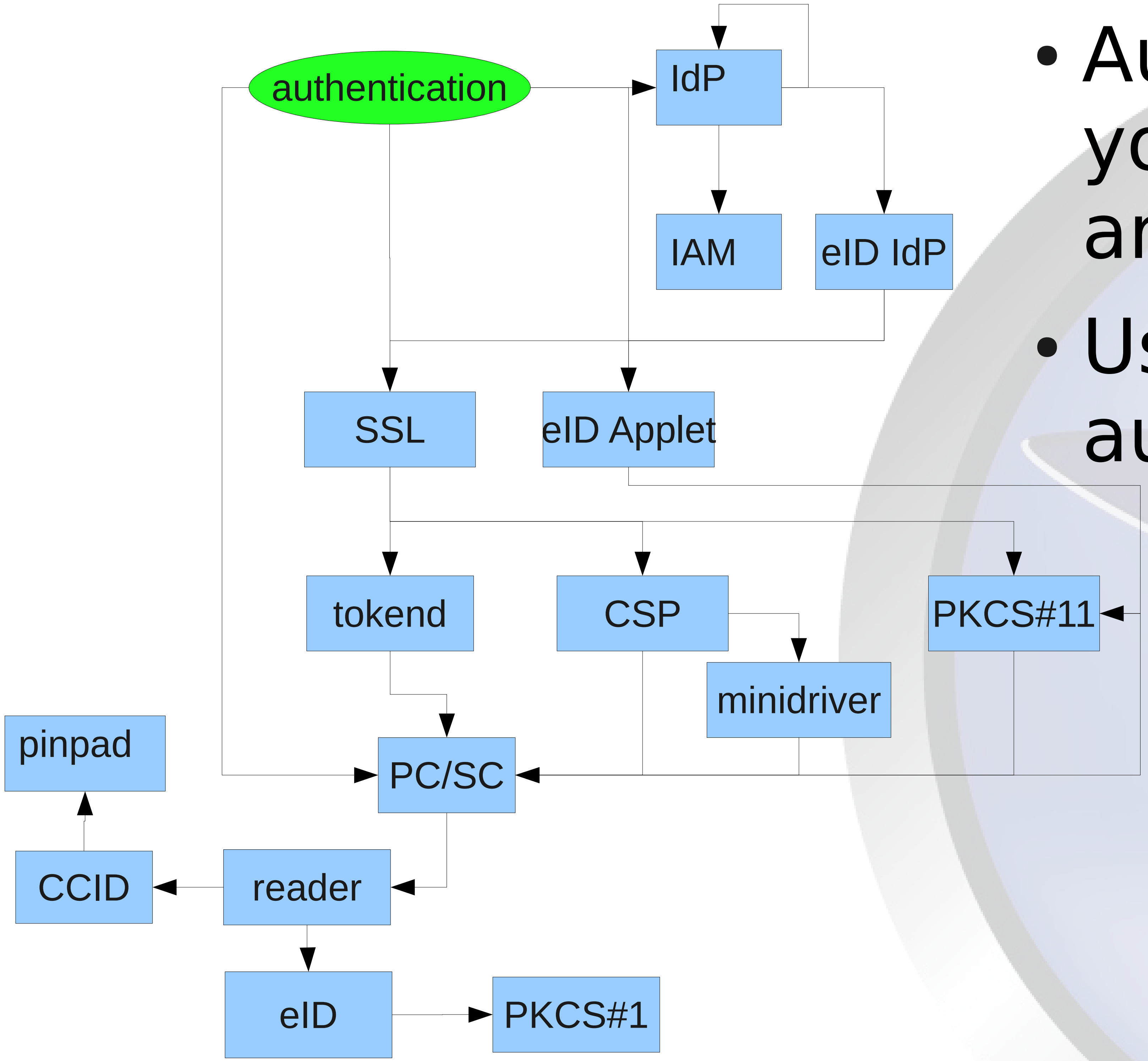
```
<servlet>
    <servlet-name>AppletIntegrityServiceServlet</servlet-name>
    <servlet-class>be.fedict.eid.applet.service.AppletServiceServlet</servlet-class>
    <init-param>
        <param-name>IdentityIntegrityService</param-name>
        <param-value>/location/in/jndi/of/the/IdentityIntegrityServiceBean</param-value>
    </init-param>
</servlet>
<servlet-mapping>
    <servlet-name>AppletIntegrityServiceServlet</servlet-name>
    <url-pattern>/applet-service-integrity</url-pattern>
</servlet-mapping>
```

eID Identification Demo Integrity Validation

eID Card Cloning

- eID has a 3th key pair: card authn key
- eID Card Authentication signature can be created without PIN verification
- eID Card Authentication allows for detection of a cloned eID card
- eID Card Authentication signature cannot be verified due to missing corresponding public key.
- National Registry needs to make the public key available as an eID file
- Would prevent identity fraude

eID Authentication



- Authentication: can you prove who you are?
- Using the eID authentication key

Entity Authentication

- Entity authentication is the process whereby one party is assured of the identity of a second party involved in a protocol, and that the second has actually participated (i.e., is active at, or immediately prior to, the time the evidence is acquired)
- Formal definition (A authenticated B if):
  - Alice A, Bob B
  - A believes freshness challenge_A
  - A believes (B recently said challenge_A)
- Authentication vs. Session Key Establishment
- How to achieve this using an eID card?

eID Authentication

- Authentication Private Key (1024 bit RSA)
  - PKCS1-RSA
  - PIN authorization for Authn Key usage
  - Card caches the authn PIN authorizations
  - Log-off instruction to reset PIN authorization
- Creation of a signature:
  - Set APDU: select the key. 0x82 = authn key
  - Prepare DigestInfo PKCS1 DER sequence
  - Verify PIN APDU: (PIN BCD encoded)
  - Compute Digital Signature APDU
  - Retrieve signature data
- eID can only sign (RSA decryption of DigestInfo)

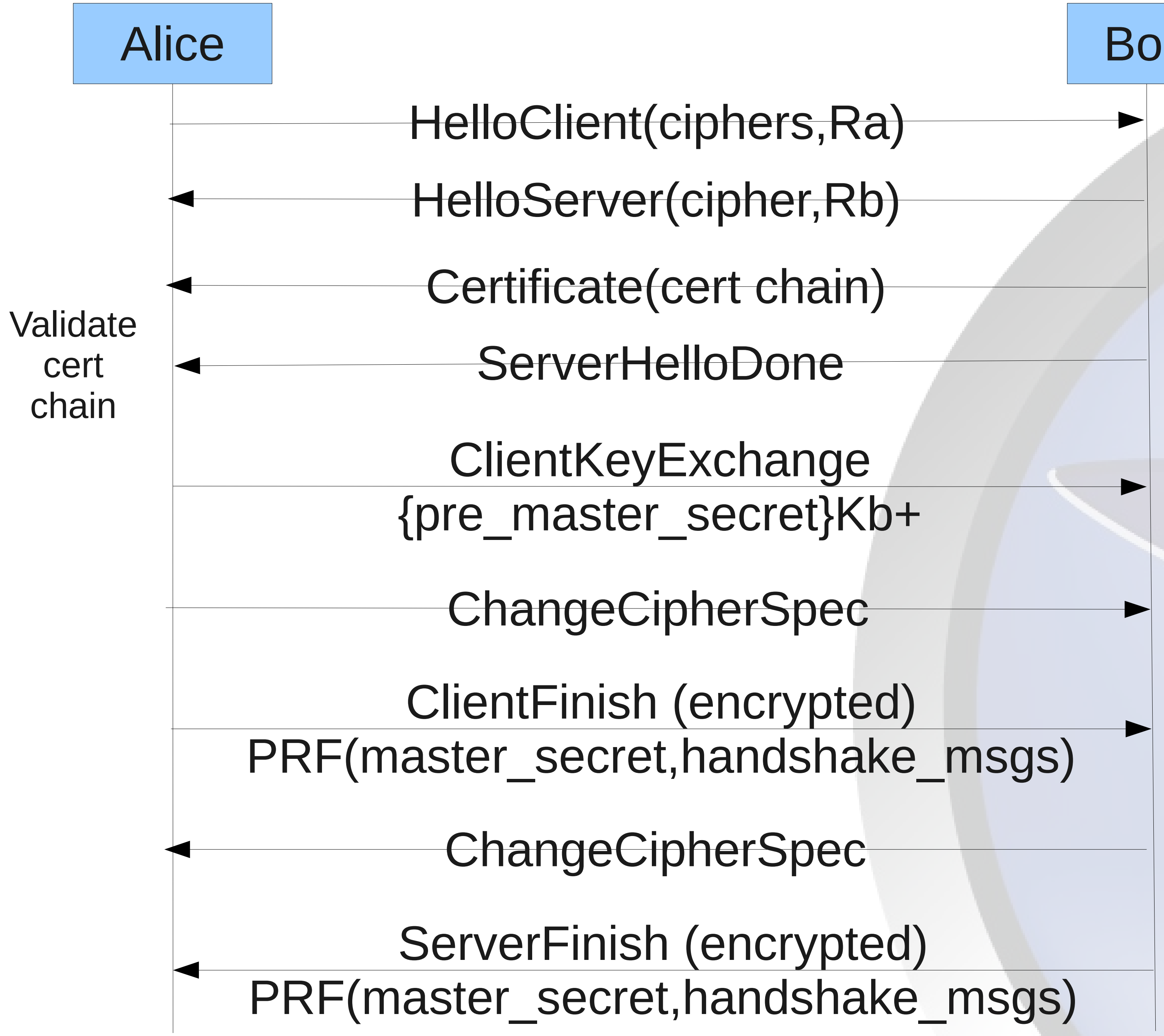
Authentication Protocol

- eID authentication by itself is useless
- Remote Entities, e.g. web application context.
- We need an Authentication Protocol
- Different Authentication Protocols are possible
- Each Entity Authentication Protocol yields its own cryptographic goals.
  - Of course Entity Authentication
  - Session key via combined Key Agreement (SSL)
- DO NOT TRY TO INVENT YOUR OWN PROTOCOL!
  - Needham-Schroeder protocol: replay attack
  - Creativity is great for non-critical applications, like music.

Candidate Protocols

- Mutual SSL
  - Browser initiated SSL handshake
  - Relies on eID PIN authorization caching feature
- Tunneled Entity Authentication
  - Uses unilateral SSL to authenticate the server
  - Based on ISO/IEC 9798-3 Authentication SASL Mechanism (RFC 3163)
  - Cryptographic channel binding to secure the channel (RFC 5056)
  - Requires an eID Applet (or browser extension)
  - Explicit eID card management possible
  - Sequential eID card access possible

Unilateral SSL



Alice

Bob

HelloClient(ciphers,Ra)

HelloServer(cipher,Rb)

Certificate(cert chain)

Validate
cert
chain

ServerHelloDone

ClientKeyExchange
{pre_master_secret}Kb+

ChangeCipherSpec

ClientFinish (encrypted)
PRF(master_secret,handshake_msgs)

ChangeCipherSpec

ServerFinish (encrypted)
PRF(master_secret,handshake_msgs)

Ra: random by A
Rb: random by B
Kb+: public key of B
pre_master_secret: random by A
PRF: pseudo-random function

Entity Authentication

•A believes freshness Ra
•A believes B recently said Ra

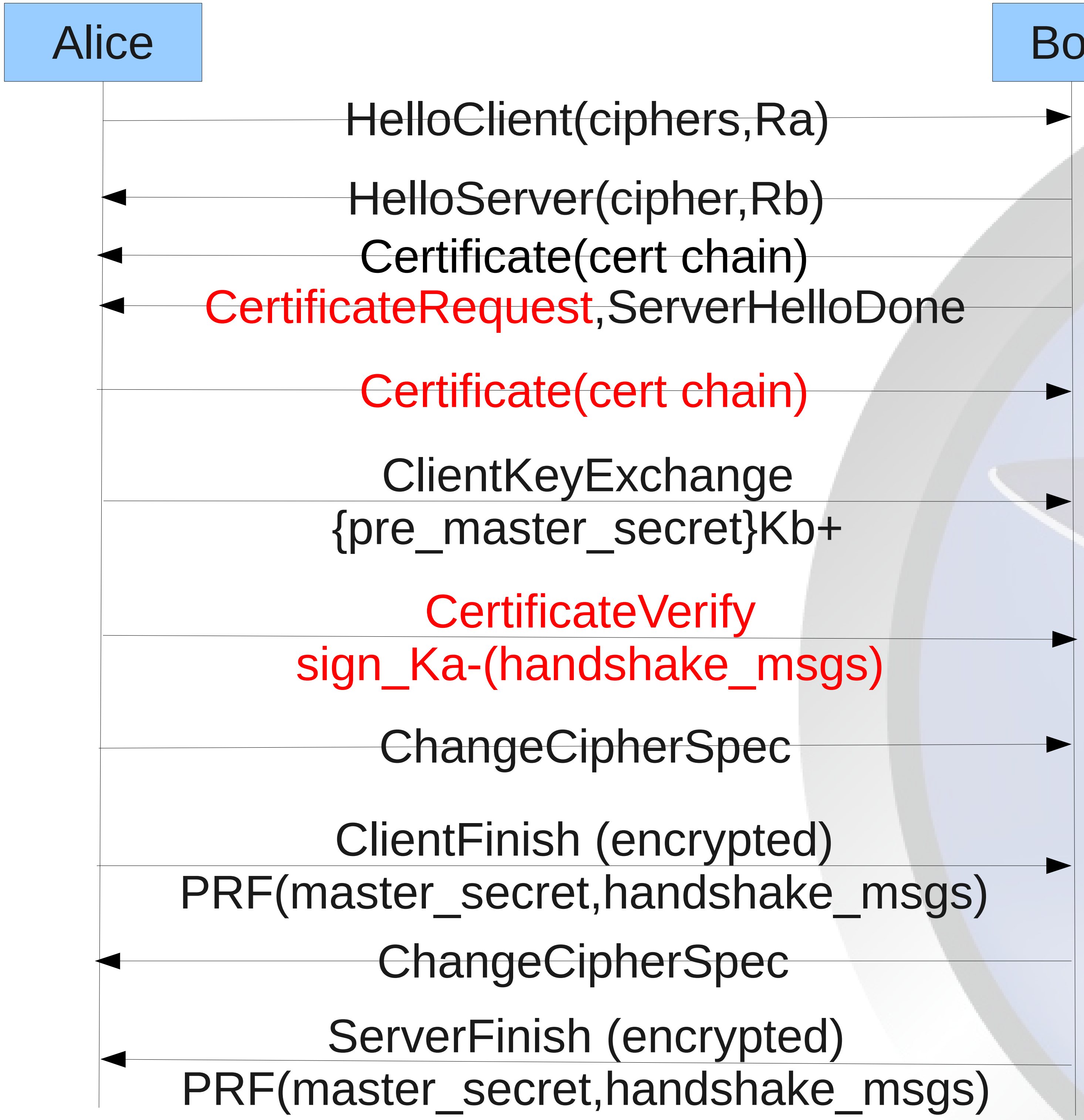
master_secret=PRF(pre_master_secret, Ra, Rb)

session_key=F(master_secret)

SSL Features

- Resuming a TLS connection
  - HelloClient(session_id)
  - Reusing the same master_secret
  - Reduces load due to a full TLS handshake
- Renegotiating the SSL handshake
  - Over an already established SSL connection
  - Useful when client authentication is required
  - Both client and server can initiate a renegotiation
  - Not all SSL stacks support this (Java does not)
  - Security flaws in implementations

Mutual SSL via eID



Alice

Bob

HelloClient(ciphers,Ra)

HelloServer(cipher,Rb)

Certificate(cert chain)

CertificateRequest,ServerHelloDone

Certificate(cert chain)

ClientKeyExchange
{pre_master_secret}Kb+

CertificateVerify
sign_Ka-(handshake_msgs)

ChangeCipherSpec

ClientFinish (encrypted)
PRF(master_secret,handshake_msgs)

ChangeCipherSpec

ServerFinish (encrypted)
PRF(master_secret,handshake_msgs)

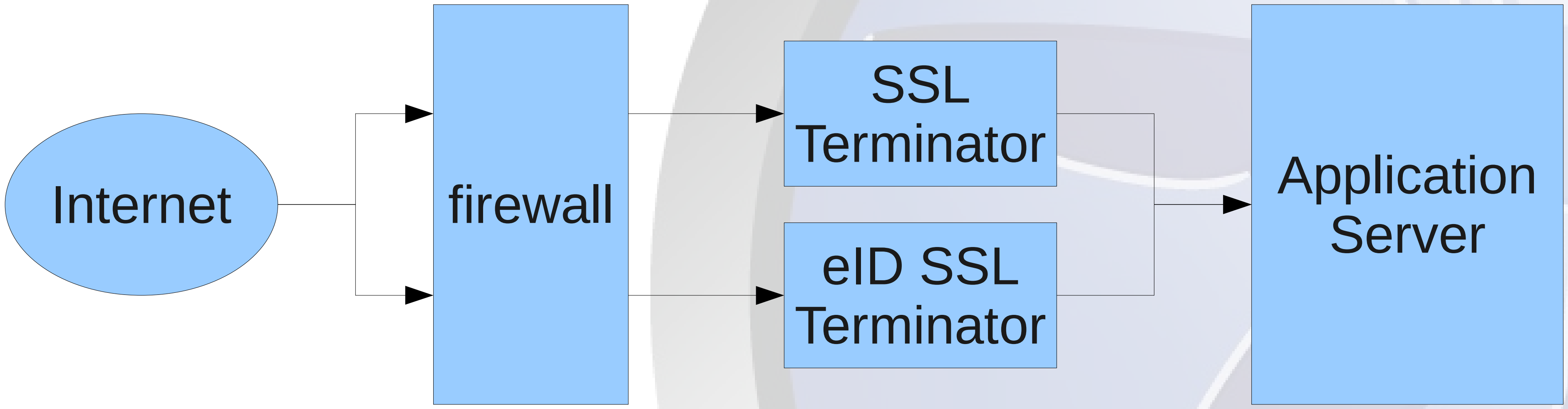
Ra: random by A
Rb: random by B
Kb+: public key of B
pre_master_secret: random by A
Ka-: private key of A
PRF: pseudo-random function

Entity Authentication

- A believes freshness Ra
- A believes (B recently said Ra)
- B believes freshness Rb
- B believes (A recently said Rb)

eID SSL Authentication

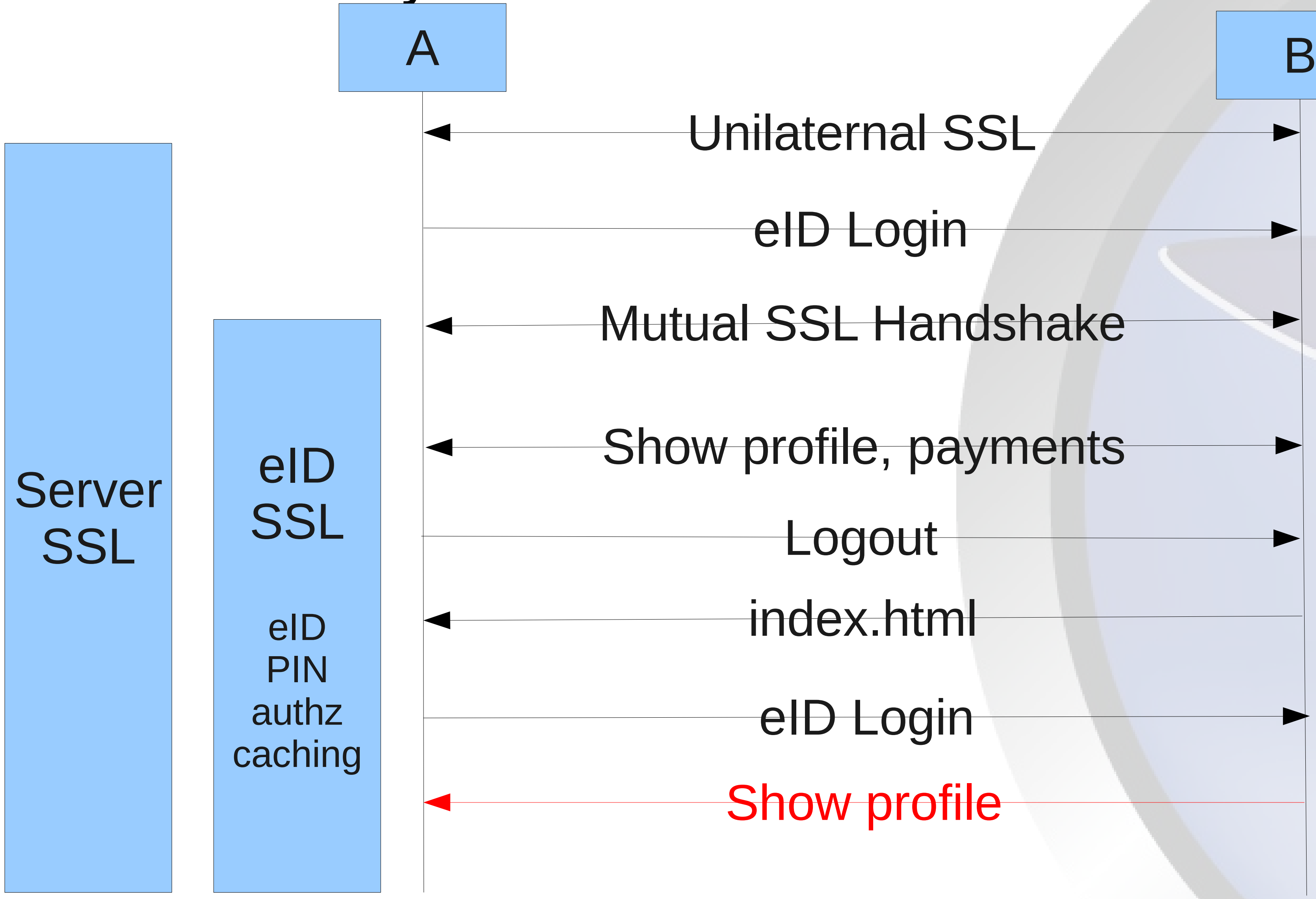
- So we need two SSL terminations:
  - One for unilateral SSL
  - One for the mutual SSL using eID
- Requires 2 IP addresses (+ DNS names), or at least 2 different ports.



- Problem: how to properly link the SSL sessions?
  - If same IP address, via session cookies
  - If different IP address, via signed SAML tickets

eID SSL Authentication

- How about session life cycles?
- The Application Server cannot inform SSL to terminate that easily...



Magical SSL Session Linking

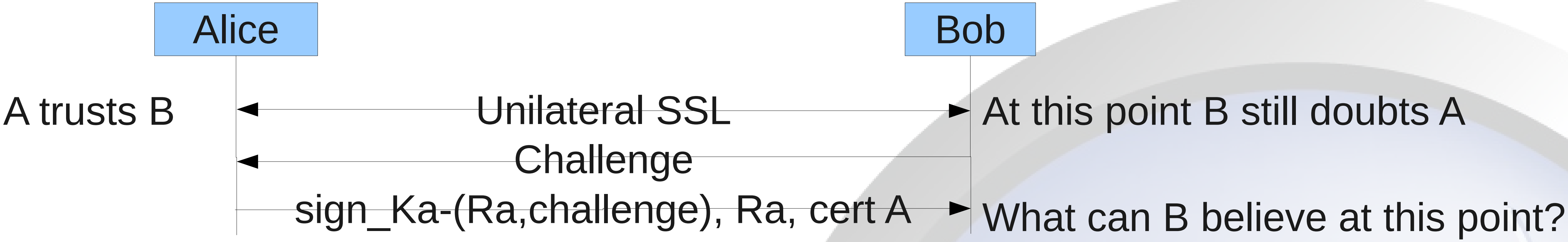
What if the web application wants to sign a contract with eID of citizen C???

eID SSL session still alive!!!

Short SSL sessions? :)

eID SSL Authentication Demo

Tunneled Entity Authn



Alice

Bob

A trusts B

Unilateral SSL

At this point B still doubts A

Challenge

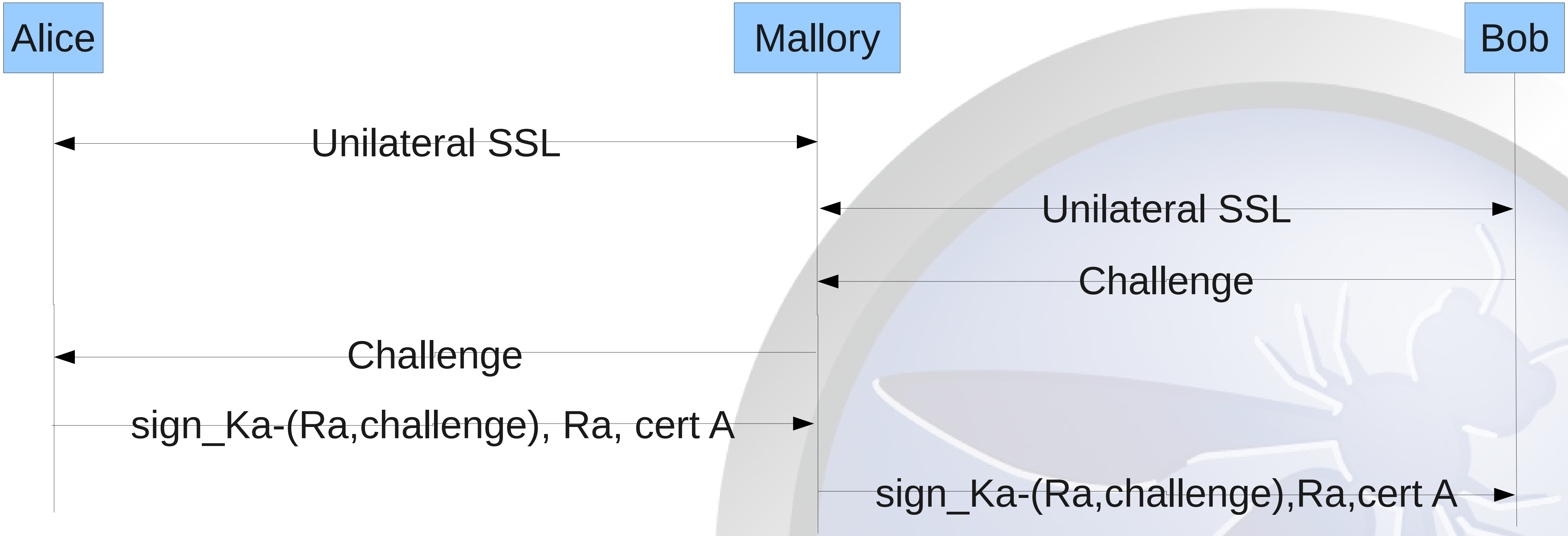
sign_Ka-(Ra,challenge), Ra, cert A

What can B believe at this point?

- ISO/IEC 9798-3 Authentication SASL Mechanism
  - RFC 3163
  - Unilateral client authentication: server already authenticated via unilateral SSL connection
- Did we achieve the same effect as mutual SSL?
- What if challenge actually is SHA1(contract)?
- B can abuse A's challenge signature.

eID Tunneled Entity Authentication Demo

MITM attack on SASL



Alice

Mallory

Bob

Unilateral SSL

Unilateral SSL

Challenge

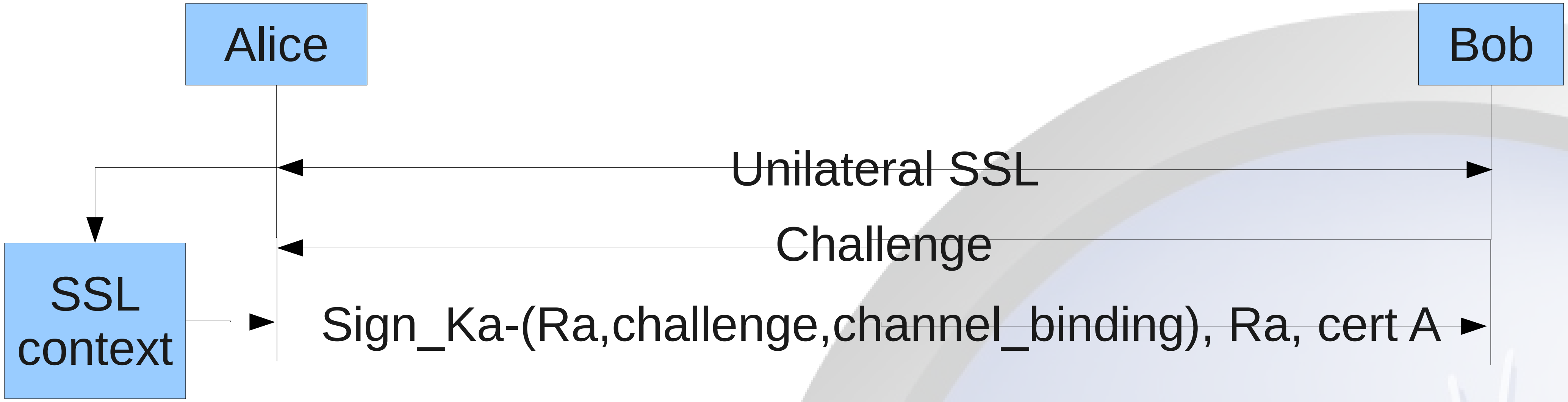
Challenge

sign_Ka-(Ra,challenge), Ra, cert A

sign_Ka-(Ra,challenge),Ra,cert A

- Mallory can abuse the authentication token of Alice
- Why is this going wrong?
  - SSL: sign_Ka-(handshake_msgs) so the signature digests parts of the secure channel's “identity”
  - SASL: sign_Ka-(Ra,challenge) does not digest any part of the secure channel's “identity”.

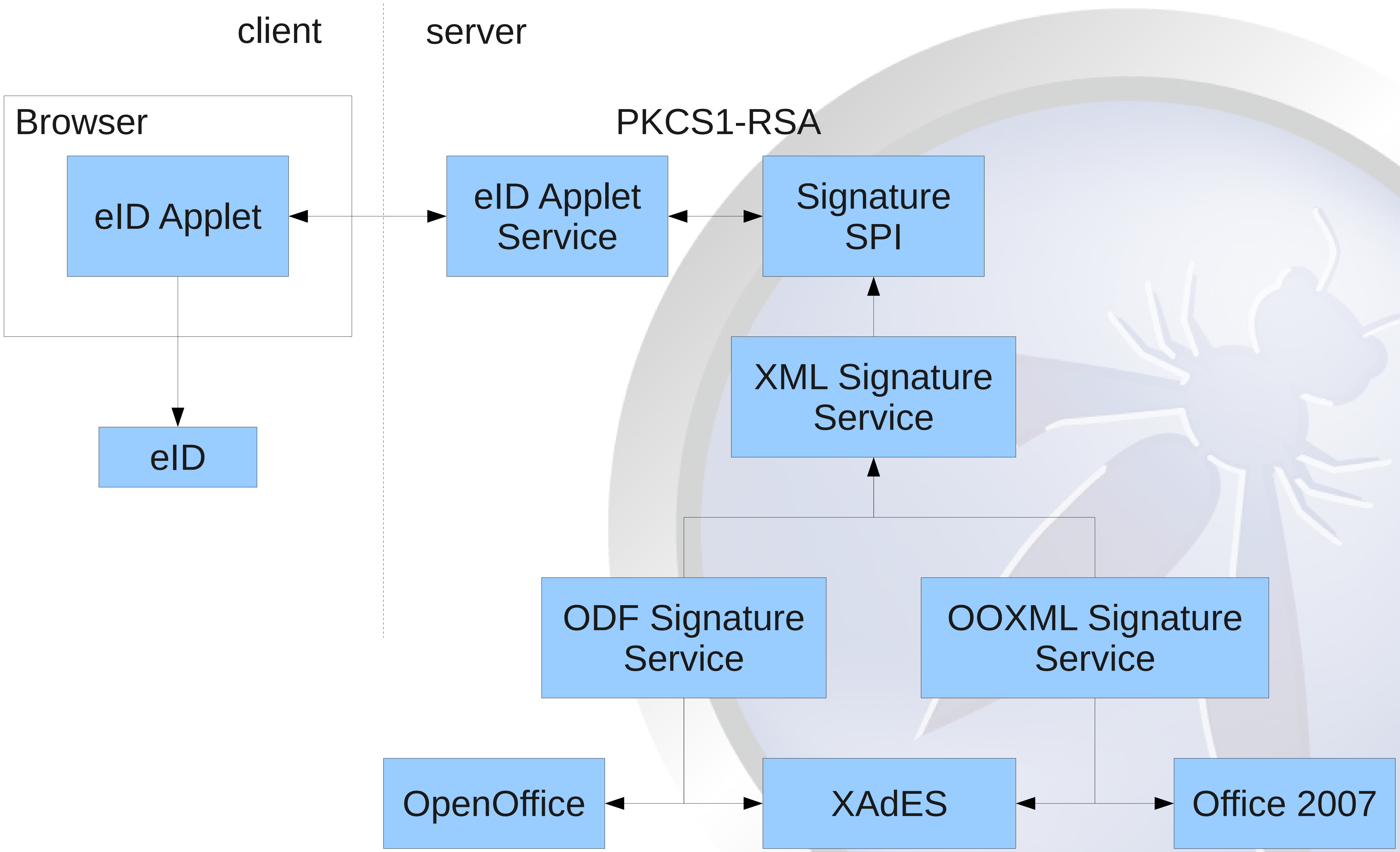
Secure Channel Binding



- RFC 5056: cryptographic binding
- channel_binding =
  - Hostname B (nice try)
  - Inet address B (nice try)
  - SSL certificate B (OK)
  - SHA1(master_key) (even better)
- A channel binding should really digest part of the channel's “identity”. Alice's SSL stack must support this.

Secure Channel Binding Demo

eID Applet Signatures



eID Applet Signature Demo

Examples of
Bad Implementations

Examples of bad implementations

- Identification is not authentication
- Unsecure trust in a third party
- After successful authentication abuse HTTP
- Web vulnerabilities in an eID site

Identification is not authentication

- Using a Java applet to retrieve first name and last name and use this for authentication => bad
- Without using HTTPS
  - Sniffing
  - MITM
  - XSRF

Unsecure trust in a third party

- Authentication occurs using a redirect or intercept by reverse proxy
- To forward credentials to web application, an insecure mechanism is used
- HTTP Header contains NRN: can be spoofed, hijacked

After successful authentication abuse HTTP

- Session cookie is used to maintain state
- Redirect to HTTP
  - Sniff cookie and spoof authentication

Web vulnerabilities in an eID site

- Cross-site-scripting to steal session cookie
- SQL Injection ㅆ
- XSRF
- Broken authorization

Questions?