Web Infrastructures

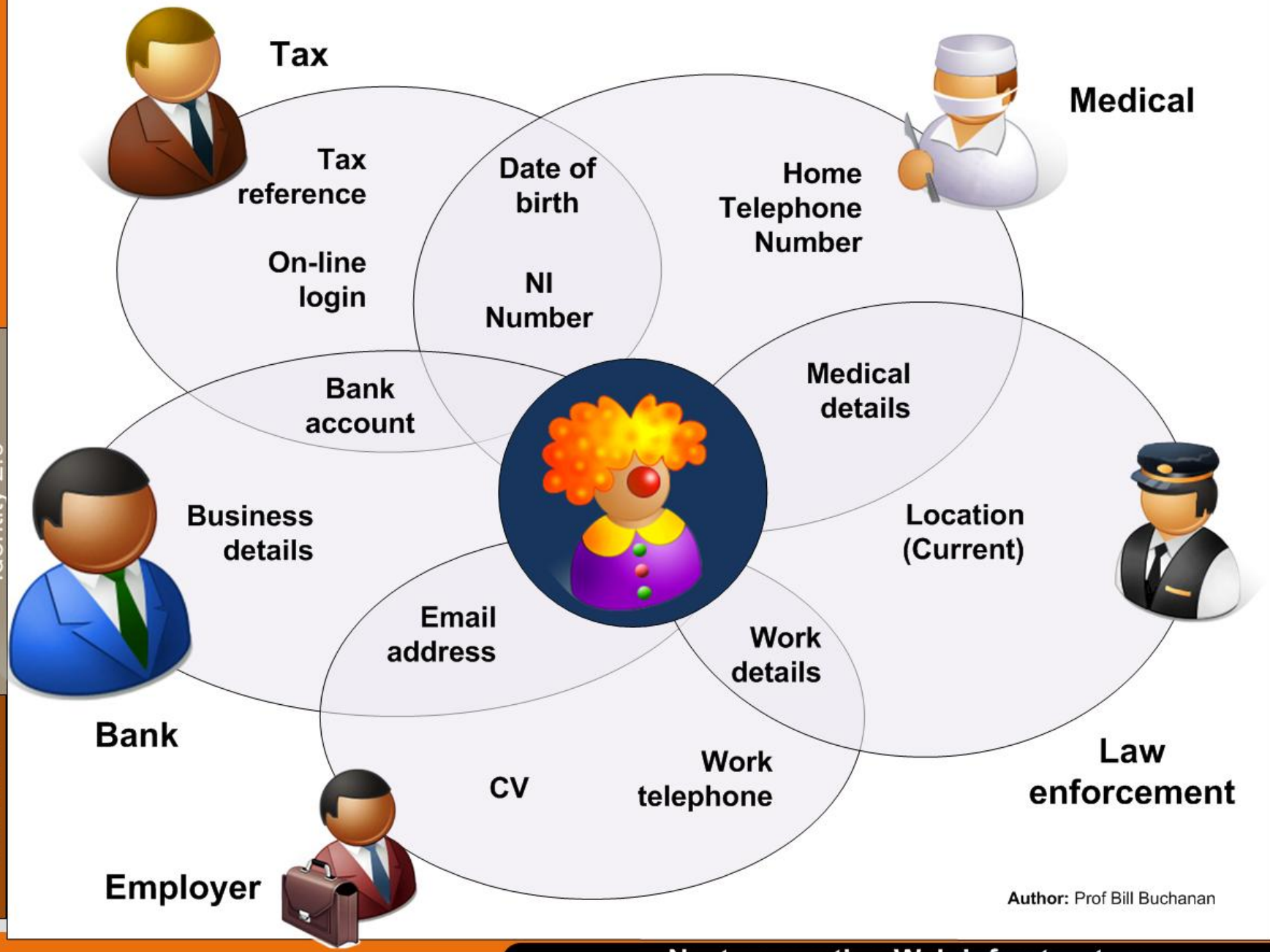
- Provide an overview of Web-based architectures, especially in authentication and access control.
- Define key protocols involved in next generation Web-based infrastructures, such as Kerberos and SOAP over HTTP.
- Define scalable authentication infrastructures and protocols.
- Investigate scaleable and extensible architectures, including using LDAP.

Author: Prof Bill Buchanan

Web Infrastructure

Identity 2.0

Author: Prof Bill Buchanan



Author: Prof Bill Buchanan

Web Infrastructure

SOAP over HTTP

Author: Prof Bill Buchanan

```
<soap:Envelope
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://
schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <CalcRootResponse xmlns="http://MyMath.com/maths">
      <CalcRootResult>9</CalcRootResult>
    </CalcRootResponse>
  </soap:Body>
</soap:Envelope>
```

Server (Windows) - Web Service

Client (Windows)

SOAP supports the encapsulation of messages and objects between different system types

Server (Linux)

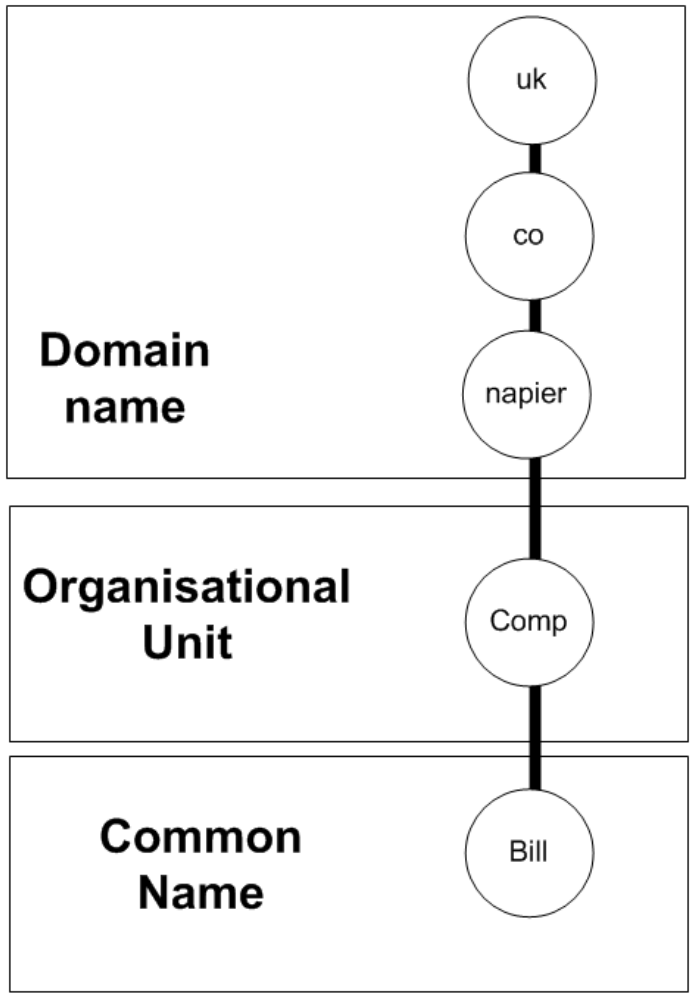
```
<double xmlns="http://MyMath.com/maths">3</double>
```

Author: Prof Bill Buchanan

Web Infrastructure

LDAP

Author: Prof Bill Buchanan

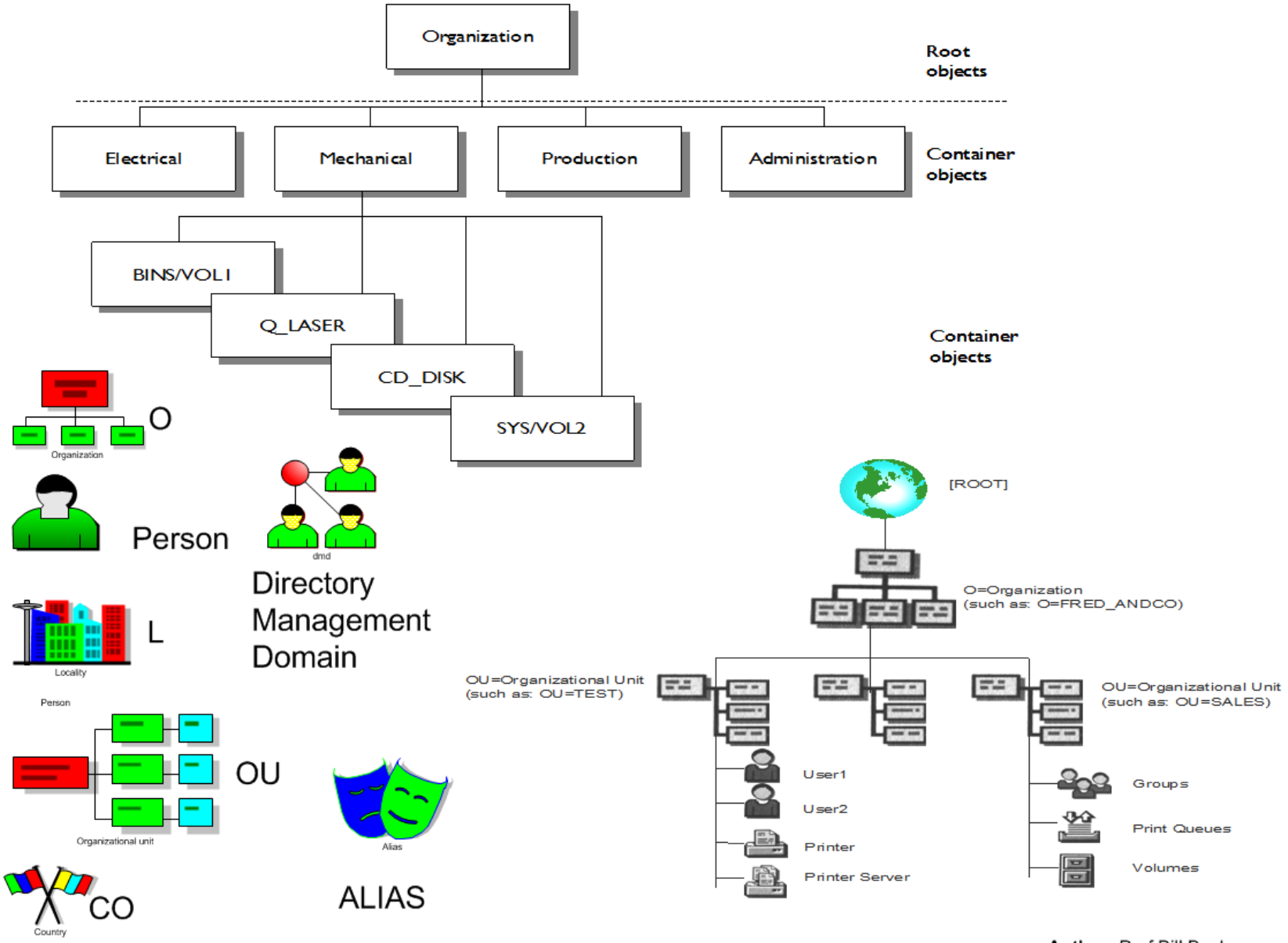


dn: dc=napier,dc=ac,dc=uk
ou: Comp
cn: Bill

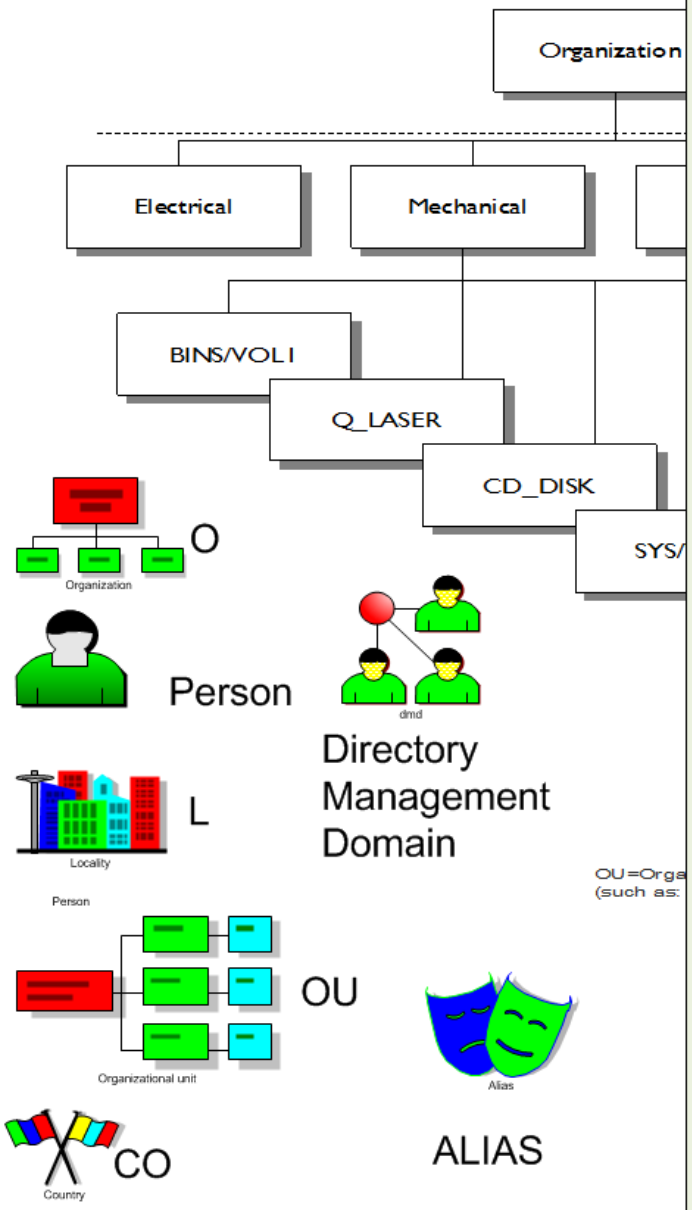
Access to Fred's folder	**cn=Fred Folder,ou=people,dc=fake,dc=com**
Identifier for Fred login	**uid=fred,ou=people,dc=fake,dc=com**
Identifier for Fred	**cn=fred,ou=people,dc=fake,dc=com**

Author: Prof Bill Buchanan

```
ldap://ldap.example.com/cn=Bill,dc=napier,dc=ac,dc=uk
```



Author: Prof Bill Buchanan

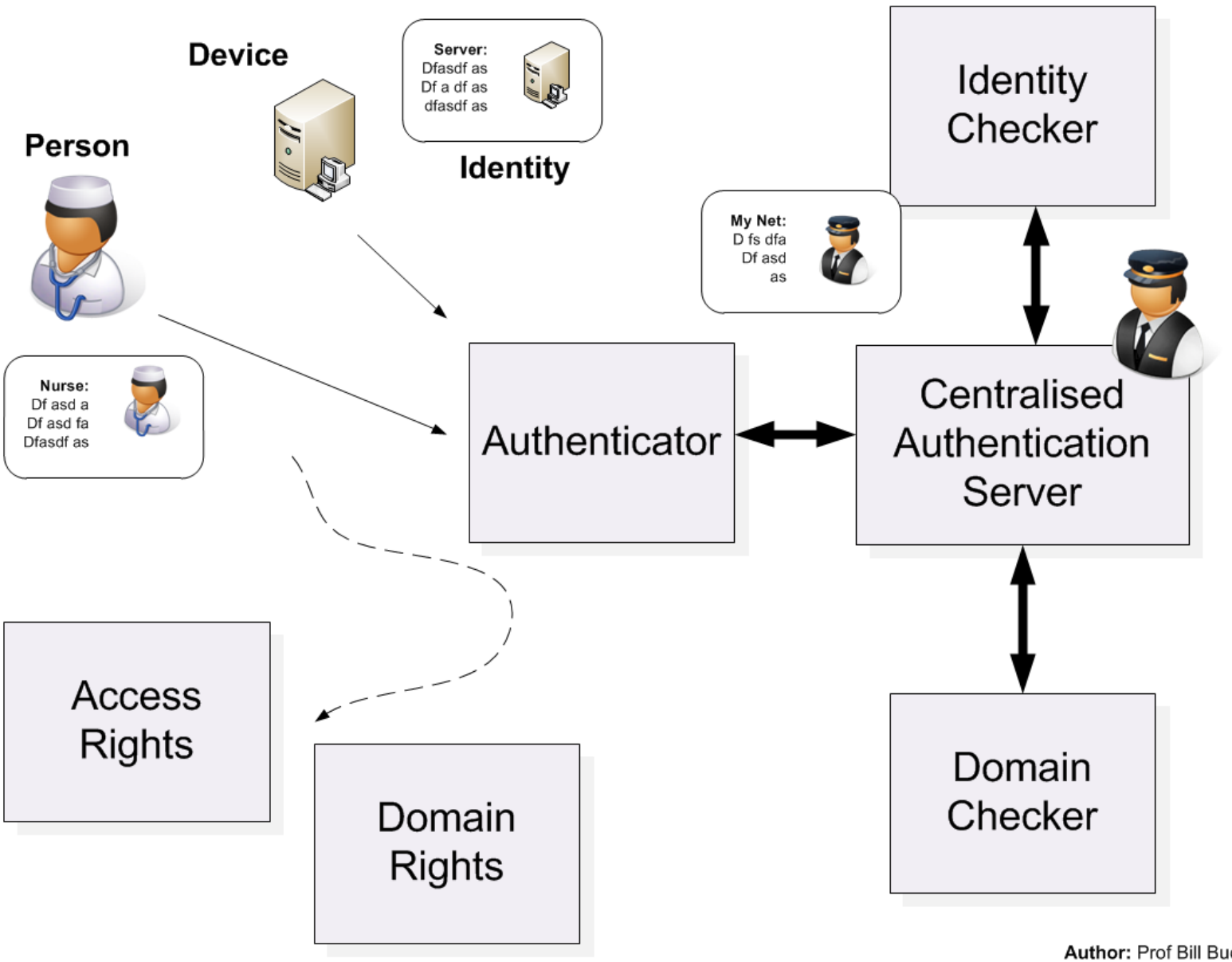



```
dn: ou=people,dc=fake,dc=com
  objectClass: organizationalUnit
  ou: people

dn: ou=groups,dc=fake,dc=com
  objectClass: organizationalUnit
  ou: groups

dn: uid=fred, ou= people, dc=fake, dc=com
  objectClass: inetOrgPerson
  objectClass: posixAccount
  objectClass: shadowAccount
  uid: fred
   givenname: Fred
   sn: Fredaldo
   cn: Freddy Fredaldo
   telephonenumber: 45511332
   roomnumber: C.63
   o: Fake Inc
   mailRoutingAddress: f.smith@fake.com
   mailhost: smtp.fake.com
   userpassword: {crypt}ggHi99x
   uidnumber: 5555
   gidnumber: 4321
   homedirectory: /user/fred
   loginshell: /usr/local/bin/bash

dn: cn=example,ou=groups, dc=fake,dc=com
  objectClass: posixGroup
  cn: example
  gidNumber: 10000
```

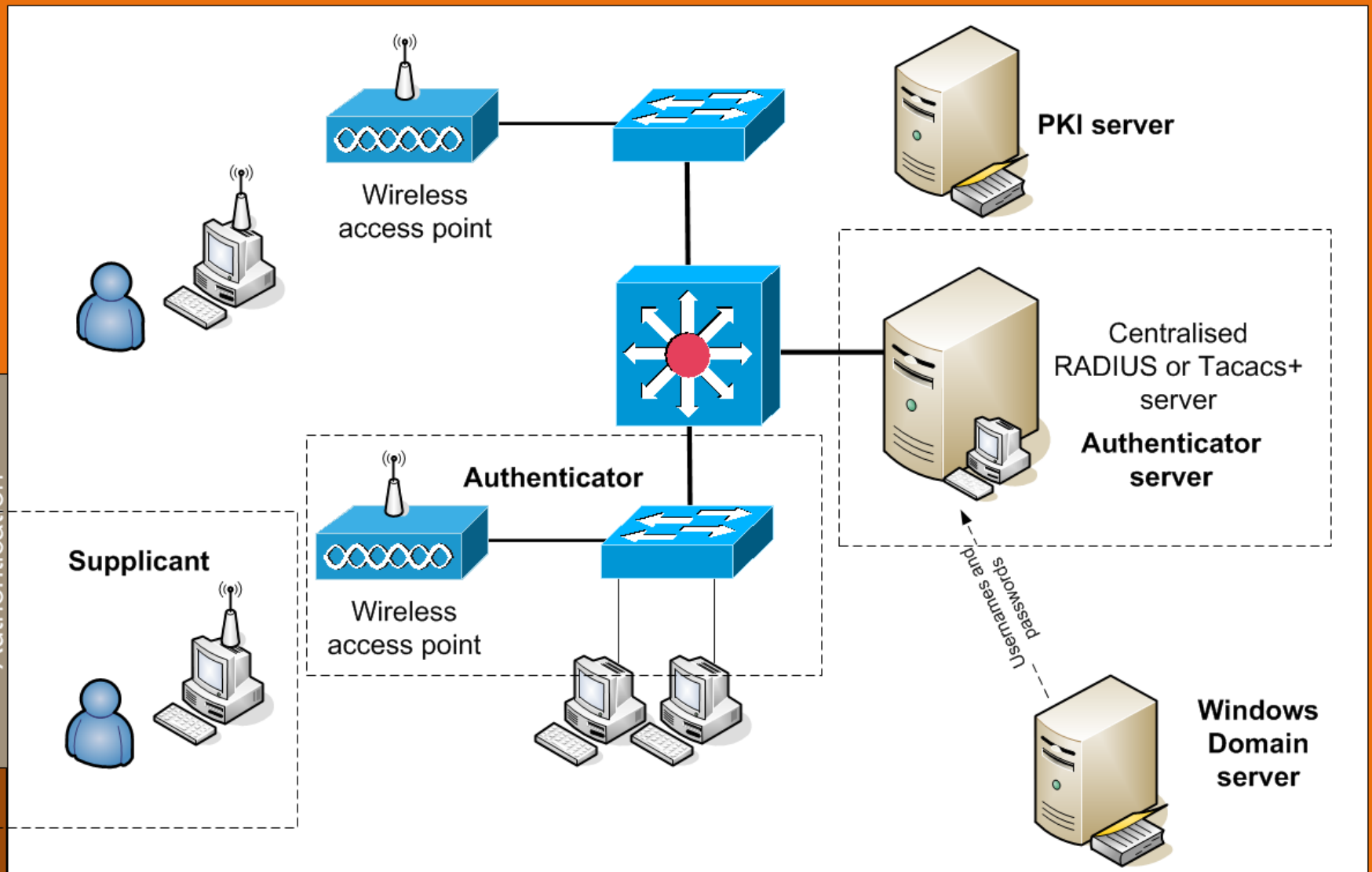
Web Infrastructure

Authentication Infrastructures

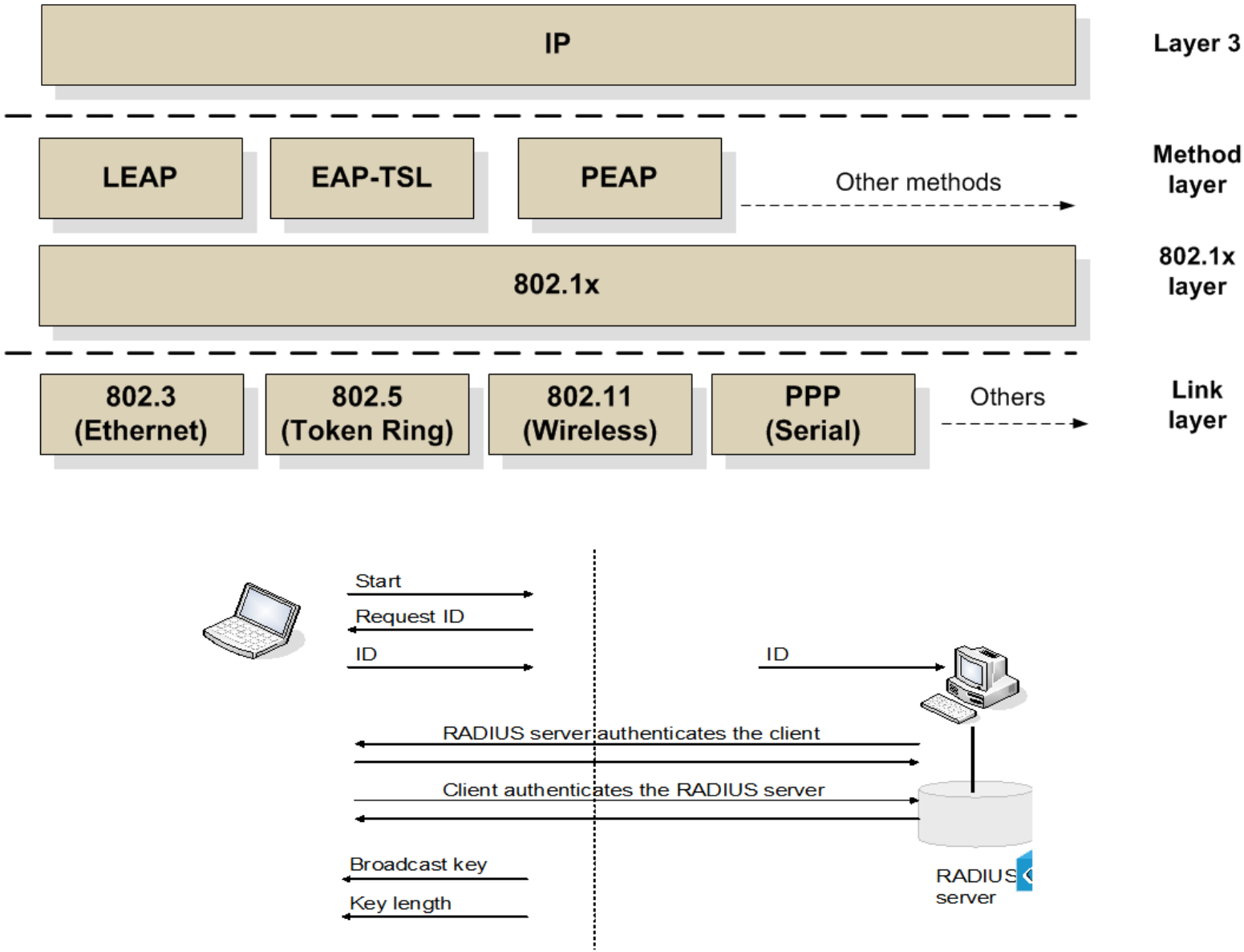
Author: Prof Bill Buchanan



Author: Prof Bill Buchanan

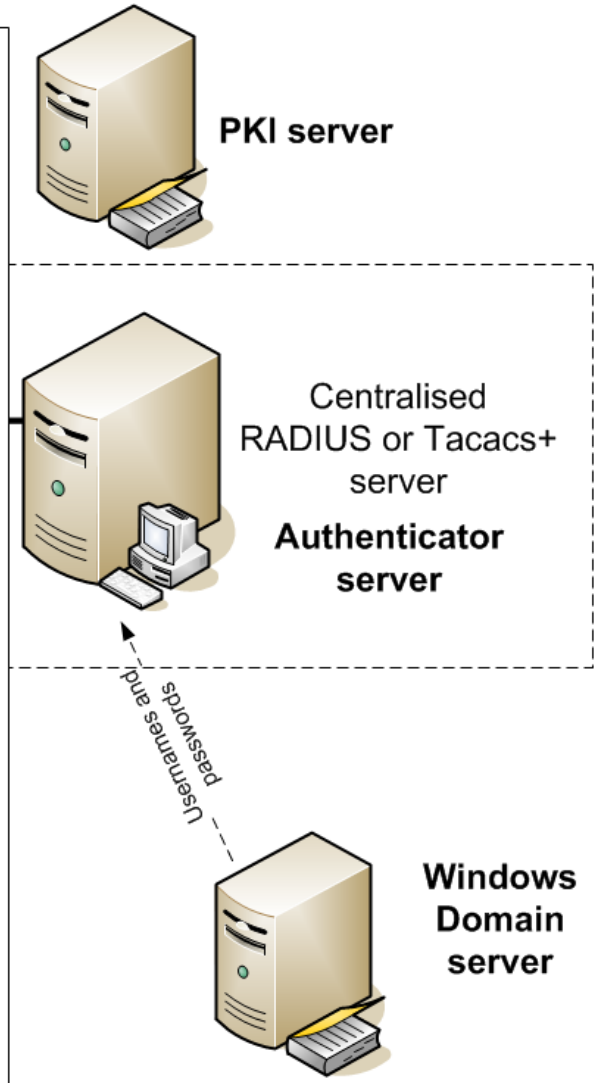


Author: Prof Bill Buchanan



Author: Prof Bill Buchanan

- **Brute-forcing of user credentials**. A malicious user can continually access the RASIUS server with a range of user ID and associated passwords, and RADIUS may eventually return a success authentication if a match is found.
- **Denial of service**. RADIUS uses UDP, which is connectionless, thus it is difficult to determine malicious from non-malicious UDP packets on ports 1812 and 1813.
- **Session replay**. There is very little authentication of the messages involved in RADIUS, thus malicious users can reply valid ones back into the next at future times.
- **Spoofed packet injection**. There is very little authentication of data packets built into RADIUS, and it can thus suffer from spoofed packet injection.
- **Response Authenticator Attack**. RADIUS uses an MD5-based hash for the Response Authenticator, thus if an intruder captures a valid Access-Request, Access-Accept, or Access-Reject packet sequence, they can launch a brute force attack on the shared secret. This is because the intruder can compute the MD5 hash for (Code+ID+Length+RequestAuth+Attributes), as most of the parts of the Authenticator are known, and can thus focus on the shared secret key.
- **Password Attribute-Based Shared Secret Attack**. Intruders can determine the share secret key but attempting to authenticate using a known password and then capturing the resulting Access-Requestpacket. After this they can then XOR the protected portion of the User-Password attribute with the password that they have used. A brute-force attack can then be done on the shared secret key
- **Shared Secret**. The basis methodology of RADIUS is that the same shared secret by many clients. Thus weakly protected clients could reveal the secret key.

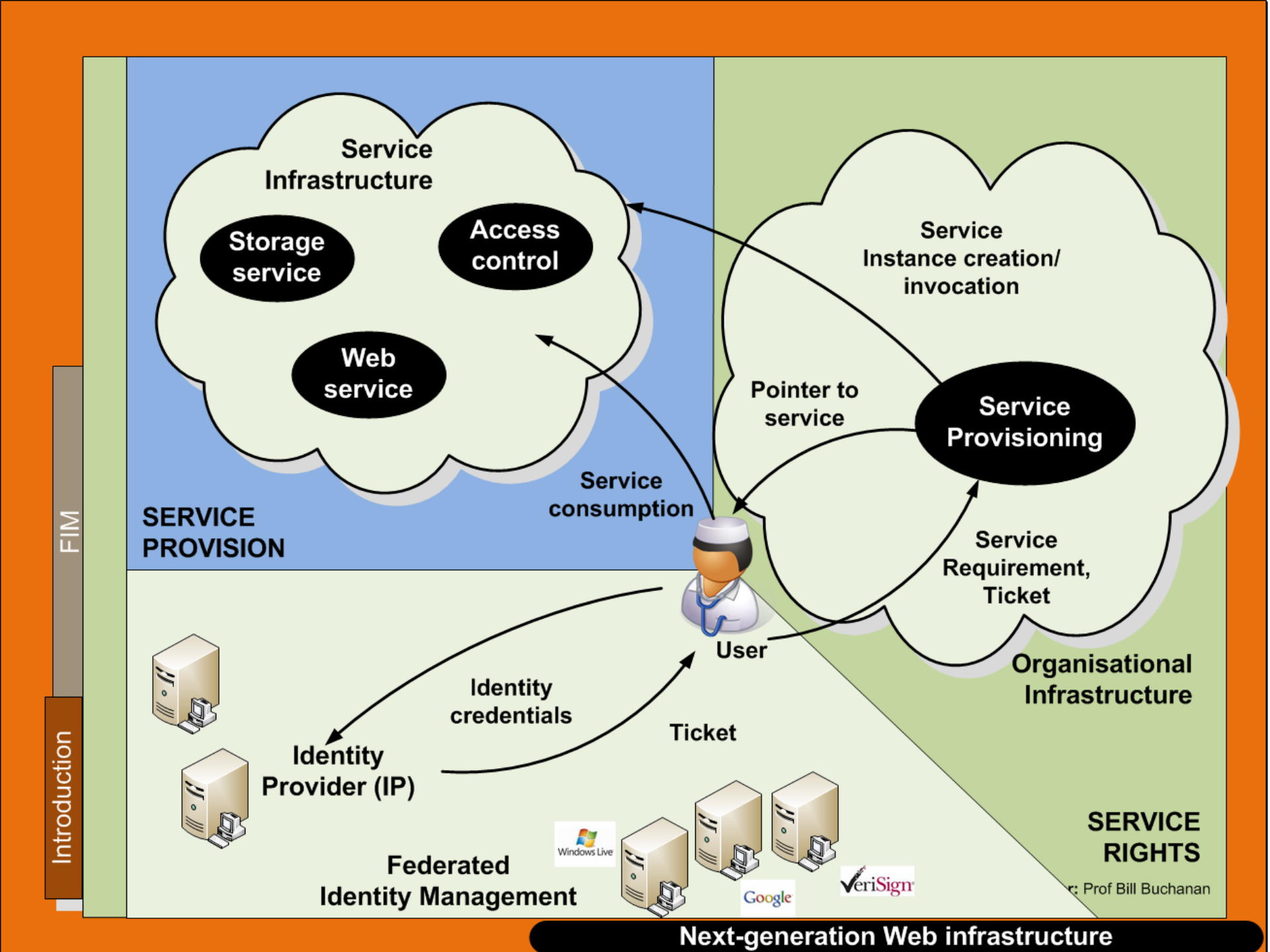


Author: Prof Bill Buchanan

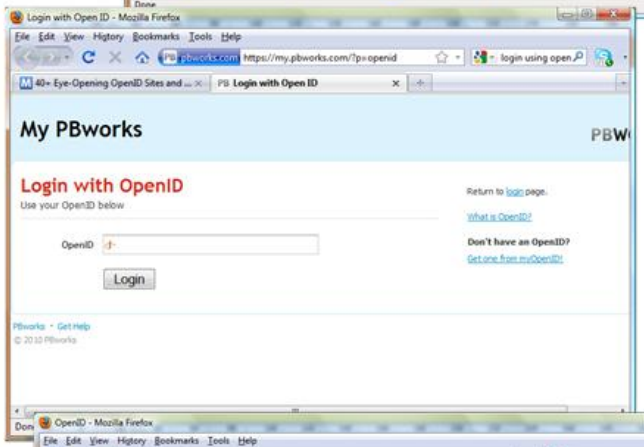
Web Infrastructure

Federated Identity Management

Author: Prof Bill Buchanan


Service Infrastructure
Storage service
Access control
Web service
SERVICE PROVISION
Service consumption
Service Instance creation/ invocation
Pointer to service
Service Provisioning
Service Requirement, Ticket
Organisational Infrastructure
User
Identity credentials
Ticket
Identity Provider (IP)
Federated Identity Management
SERVICE RIGHTS
Windows Live
Google
VeriSign
FIM
Introduction
Next-generation Web infrastructure

Author: Prof Bill Buchanan

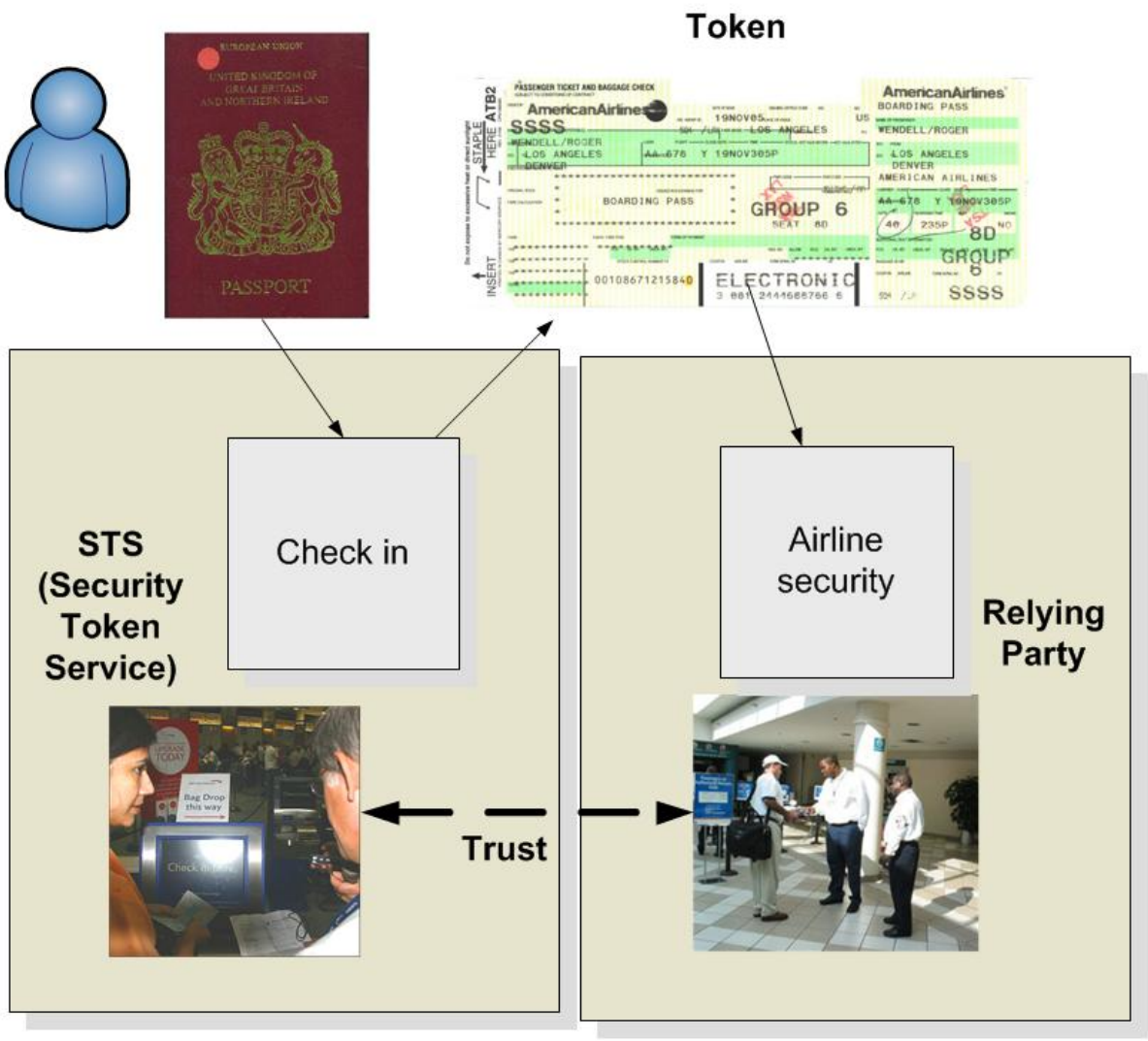


Authenticated into the OpenID infrastructure

http://billbuchanan.myopenid.com/

Others:

- AOL – openid.aol.com/screenname
- Flickr (Flickr) – www.flickr.com/photos/username
- LiveDoor profile.livedoor.com/username
- Orange (France Telecom) – http://openid.orange.fr
- Yahoo (Yahoo!) – http://openid.yahoo.com
- WordPress.com – username.wordpress.com


Token
EUROPEAN UNION
UNITED KINGDOM OF GREAT BRITAIN AND NORTHERN IRELAND
PASSPORT
PASSENGER TICKET AND BAGGAGE CHECK
AmericanAirlines
SSSS
WENDELL/ROGER
LOS ANGELES
DENVER
BOARDING PASS
GROUP 6
SEAT 8D
ELECTRONIC
STAPLE HERE ATB2
INSERT
STS (Security Token Service)
Check in
Airline security
Relying Party
Bag Drop this way
Trust

Authentication

Web Infra

Kerberos

STS

Check in

Trust Infrastructure

Token

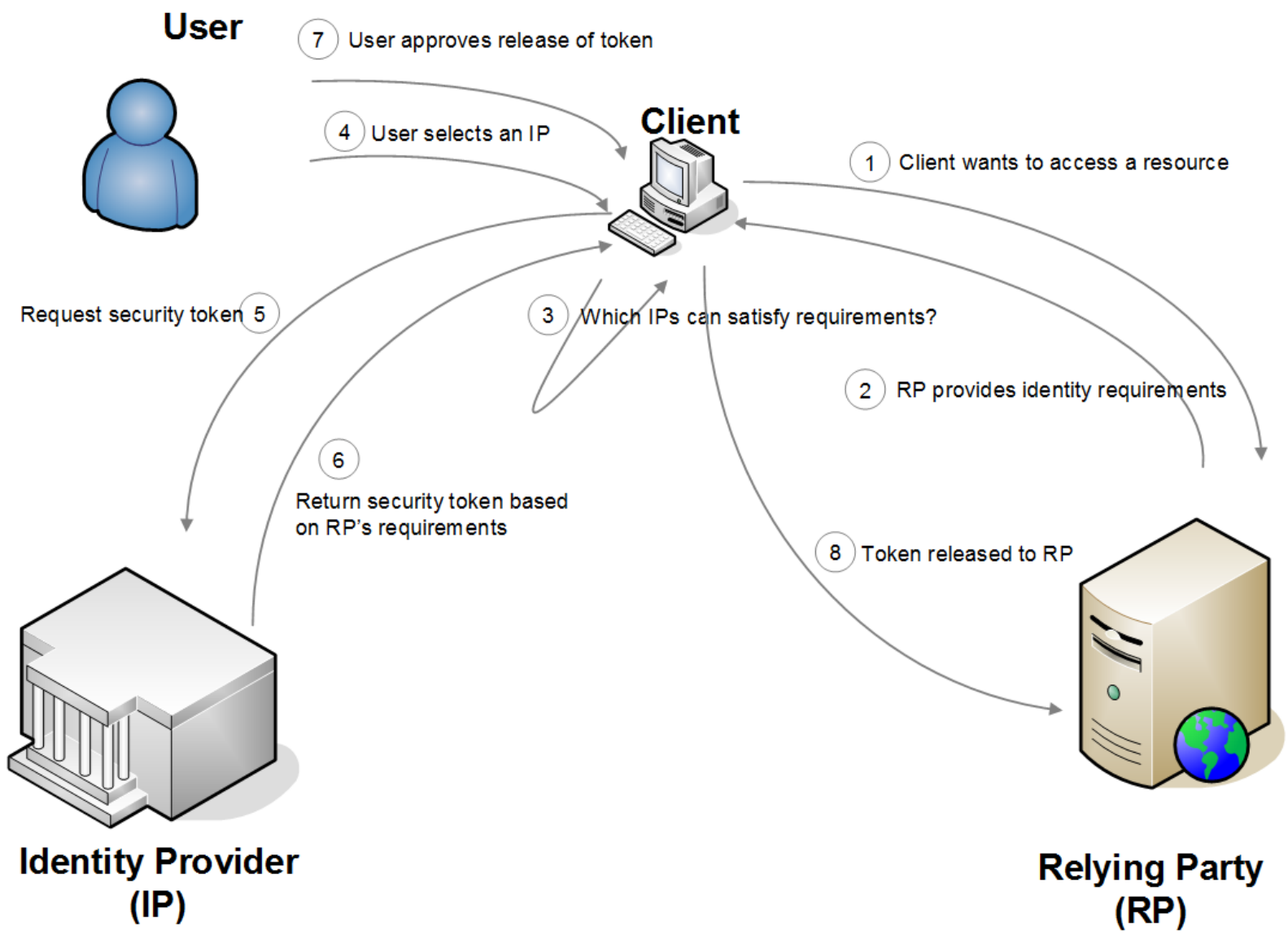
Airline security

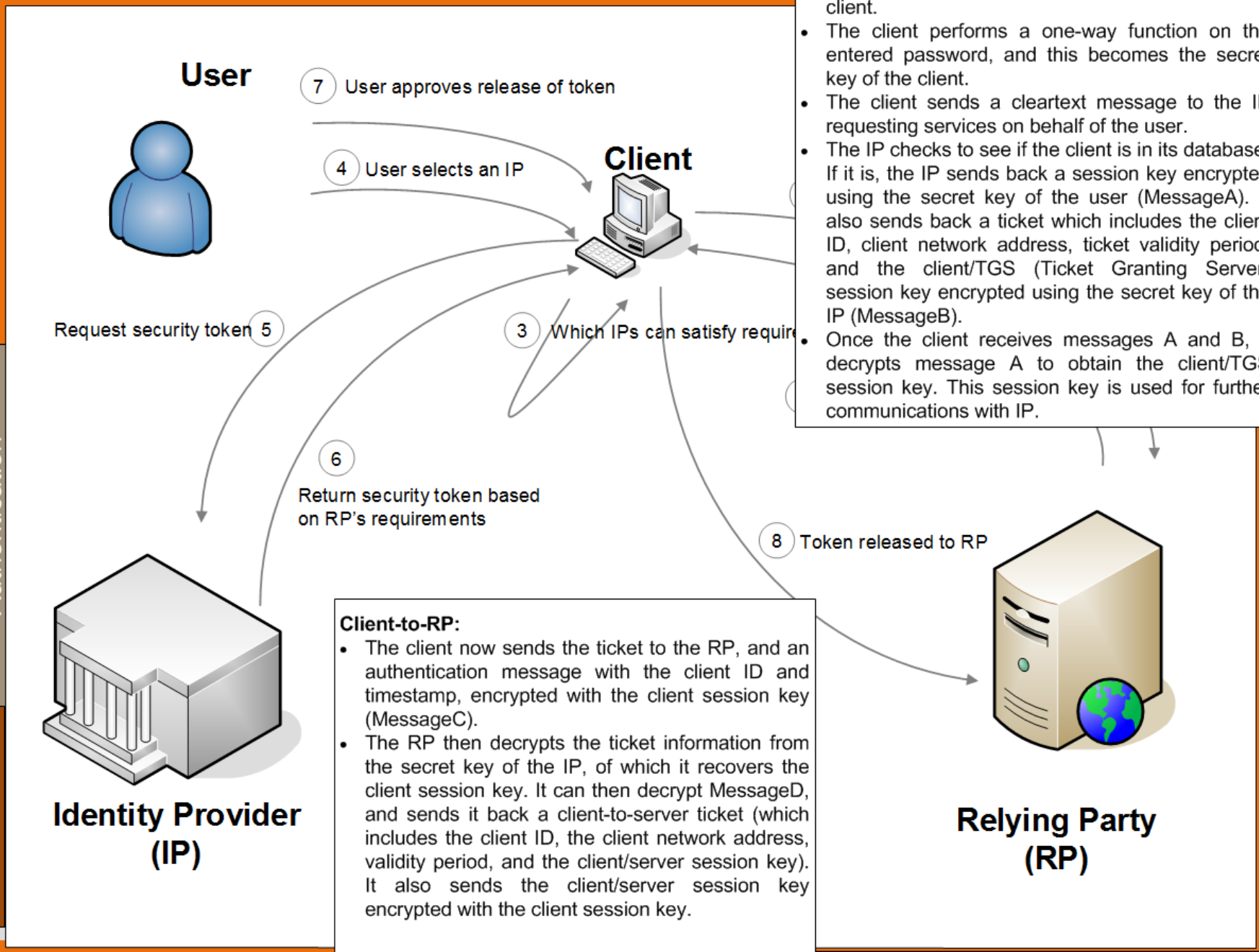
Relying Party

STS

Check in

Trust


User
7 User approves release of token
4 User selects an IP
Client
1 Client wants to access a resource
Request security token 5
3 Which IPs can satisfy requirements?
2 RP provides identity requirements
6
Return security token based on RP's requirements
8 Token released to RP
Identity Provider (IP)
Relying Party (RP)



Client to IP:

- A user enters a username and password on the client.
- The client performs a one-way function on the entered password, and this becomes the secret key of the client.
- The client sends a cleartext message to the IP requesting services on behalf of the user.
- The IP checks to see if the client is in its database. If it is, the IP sends back a session key encrypted using the secret key of the user (MessageA). It also sends back a ticket which includes the client ID, client network address, ticket validity period, and the client/TGS (Ticket Granting Server) session key encrypted using the secret key of the IP (MessageB).
- Once the client receives messages A and B, it decrypts message A to obtain the client/TGS session key. This session key is used for further communications with IP.

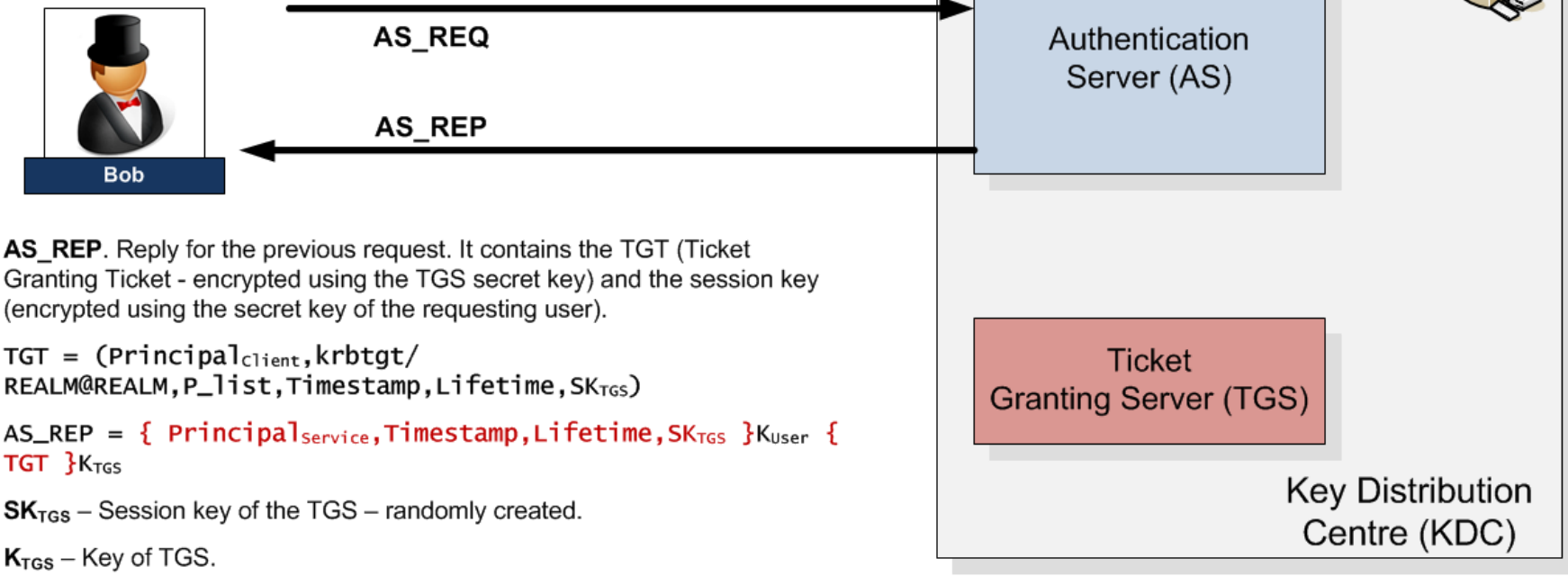
Client-to-RP:

- The client now sends the ticket to the RP, and an authentication message with the client ID and timestamp, encrypted with the client session key (MessageC).
- The RP then decrypts the ticket information from the secret key of the IP, of which it recovers the client session key. It can then decrypt MessageD, and sends it back a client-to-server ticket (which includes the client ID, the client network address, validity period, and the client/server session key). It also sends the client/server session key encrypted with the client session key.

AS_REQ is the initial user authentication request. This message is directed to the KDC component known as Authentication Server (AS).

AS_REQ = ($Principal_{Client}$, $Principal_{service}$, IP_list, Lifetime)

Eg $Principal_{Client}$ = Principal for user (such as fred@home.com), IP_list = all IP address which will use the ticket (may be null if behind NAT), lifetime = require life of the ticket.



AS_REP. Reply for the previous request. It contains the TGT (Ticket Granting Ticket - encrypted using the TGS secret key) and the session key (encrypted using the secret key of the requesting user).

TGT = ($Principal_{Client}$, krbtgt/REALM@REALM, P_list, Timestamp, Lifetime, SK_{TGS})

AS_REP = { $Principal_{service}$, Timestamp, Lifetime, SK_{TGS} }K_{User} { TGT }K_{TGS}

SK_{TGS} – Session key of the TGS – randomly created.

K_{TGS} – Key of TGS.

K_{user} – Secret key of Bob.

Note:

{ Message } – The curly brackets identify an encrypted message.

(Message) – The round brackets identify an non-encrypted message.

$AS\_REQ = (Principal_{Client}, Principal_{service}, IP\_list, Lifetime)$

AS_REQ

$TGT = (Principal_{Client}, krbtgt/REALM@REALM, P\_list, Timestamp, Lifetime, SK_{TGS})$

$AS\_REP = \{ Principal_{service}, Timestamp, Lifetime, SK_{TGS} \}K_{User} \{ TGT \}K_{TGS}$

AS_REP

Bob now needs a service ticket

TGS_REQ

$Authenticator = \{ Principal_{Client} , Timestamp \}SK_{TGS}$

$TGS\_REQ = (Principal_{Service} , Lifetime , Authenticator) \{ TGT \}K_{TGS}$

TGS_REP

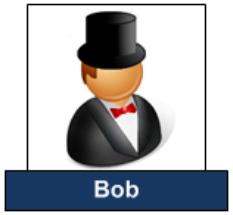
$T_{Service} = (Principal_{Client}, Principal_{service}, IP\_list, Timestamp, Lifetime, SK_{Service})$

$TGS\_REP = \{Principal_{service}, Timestamp, Lifetime, SK_{service} \}SK_{TGS} \{ T_{Service} \}K_{service}$

Authentication Server (AS)

Ticket Granting Server (TGS)

Key Distribution Centre (KDC)



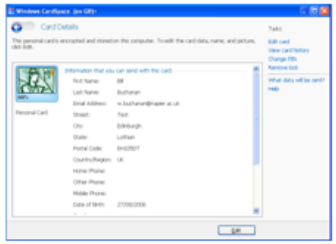
SK_{TGS} – Session key of the TGS – randomly created.

K_{TGS} – Key of TGS.

K_{user} – Secret key of Bob.

$SK_{Service}$ – Secret key of the service

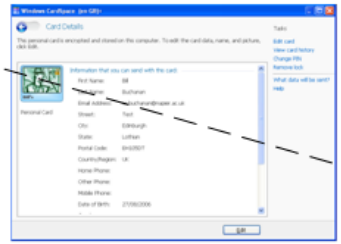
Author: Prof Bill Buchanan



Roaming details

Personal Card

Secure communication of details

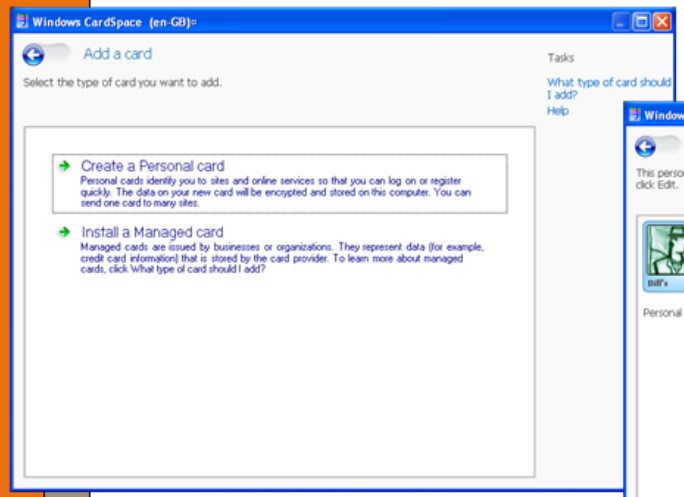


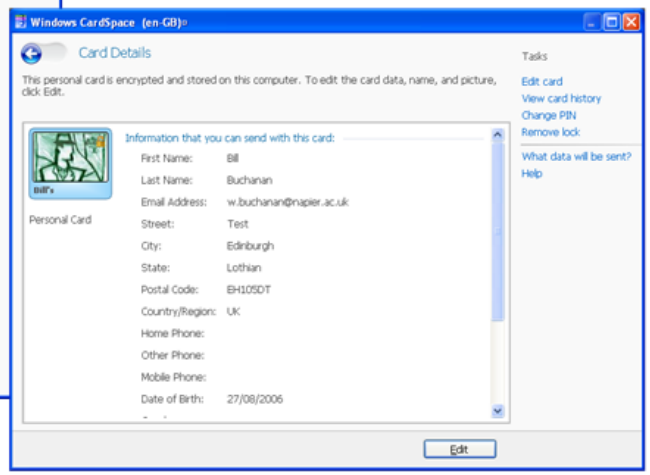
Managed Card
(for on-line purchases, managed logins, and so on

Verification of the user

Off machine storage

Storage of sensitive details
(such as credit card details, passwords, and so on)

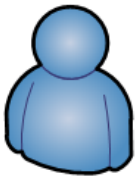


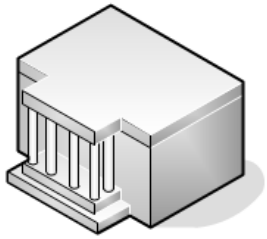


Personal cards:
- Created by the person.
- Encrypted.

Personal information:
Name, addresses, phone numbers, date of birth, and gender.

Additional:
Card name, card picture, and card creation date and a history of the sites where this card was used.

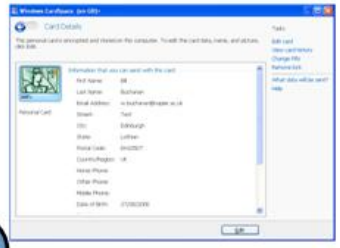




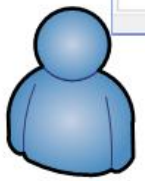
Managed Cards:
- Created by identity provider.
- Encrypted.

Information:
Maintained by IP that provides card.
Stored at site.
Some info on local machine (Card name, when installed, Valid until date, History of card)

Author: Prof Bill Buchanan



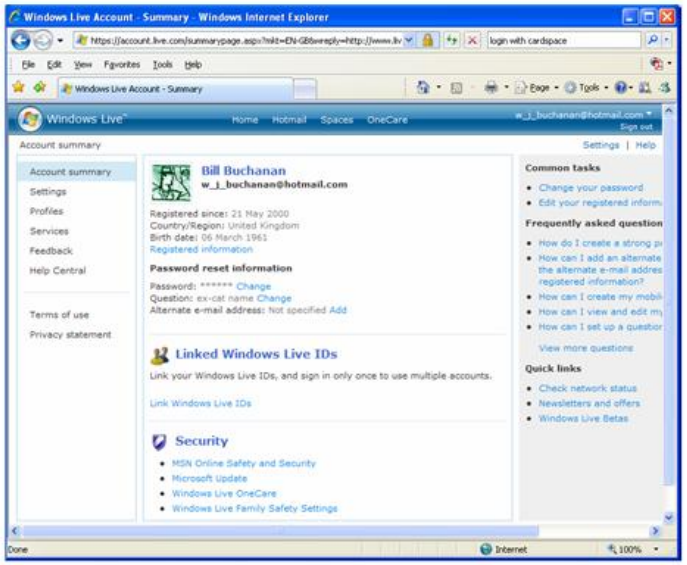
Personal Card

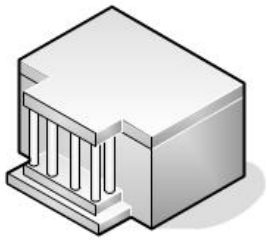


User

Name, addresses, phone numbers, date of birth, and gender.

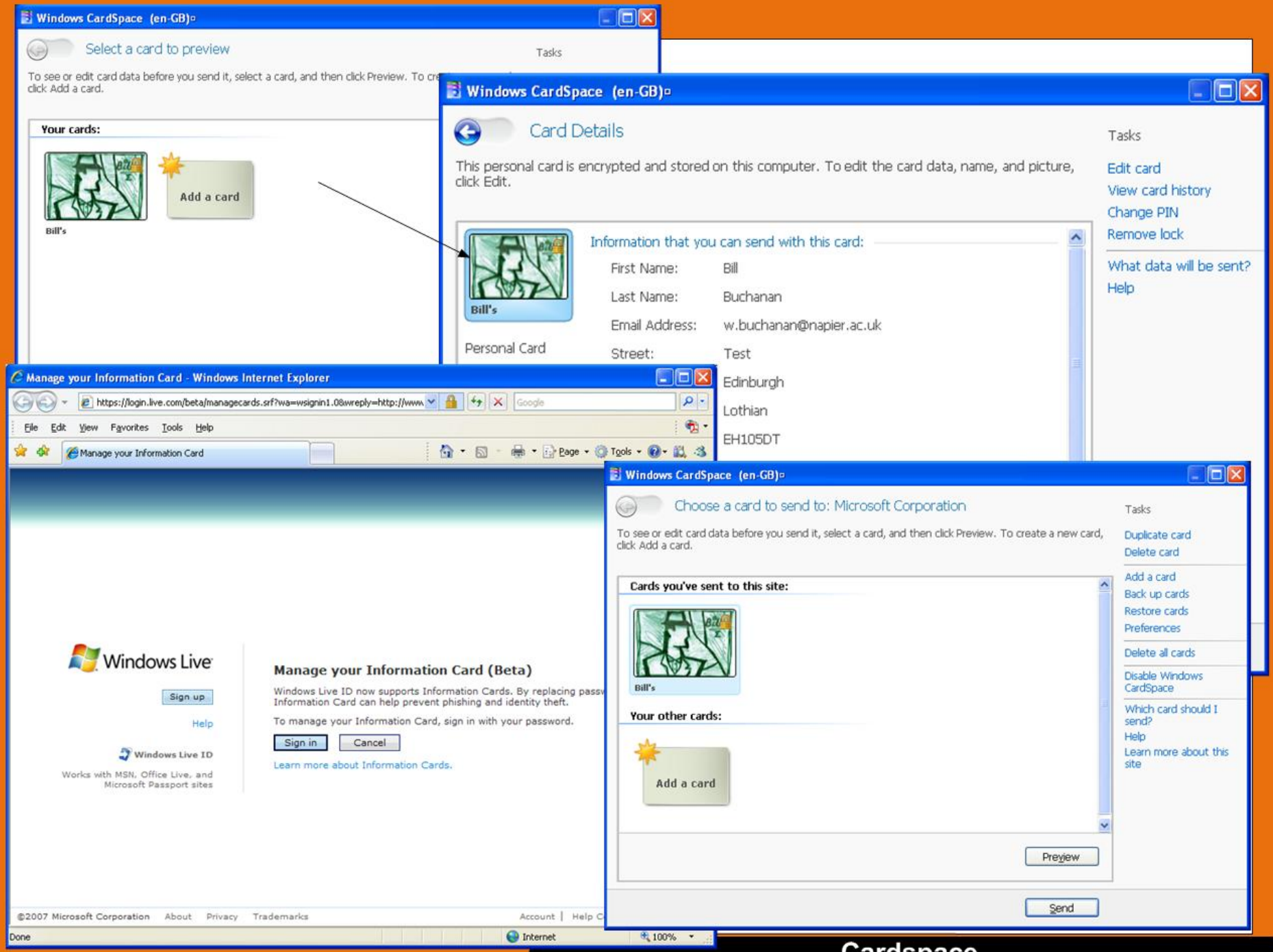
Relying Parity (RP)





Identity Provider (IP)

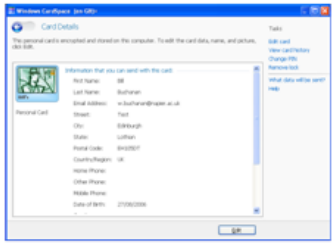
Author: Prof Bill Buchanan


Windows CardSpace (en-GB)
Select a card to preview
Tasks
To see or edit card data before you send it, select a card, and then click Preview. To cr
click Add a card.
Your cards:
Add a card
Bill's
Windows CardSpace (en-GB)
Card Details
This personal card is encrypted and stored on this computer. To edit the card data, name, and picture, click Edit.
Tasks
Edit card
View card history
Change PIN
Remove lock
What data will be sent?
Help
Information that you can send with this card:
First Name: Bill
Last Name: Buchanan
Email Address: w.buchanan@napier.ac.uk
Street: Test
Edinburgh
Lothian
EH105DT
Bill's
Personal Card
Manage your Information Card - Windows Internet Explorer
https://login.live.com/beta/managecards.srf?wa=wsignin1.0&wreply=http://www
File Edit View Favorites Tools Help
Manage your Information Card
Page Tools
Windows Live
Sign up
Help
Windows Live ID
Works with MSN, Office Live, and Microsoft Passport sites
Manage your Information Card (Beta)
Windows Live ID now supports Information Cards. By replacing passw
Information Card can help prevent phishing and identity theft.
To manage your Information Card, sign in with your password.
Sign in
Cancel
Learn more about Information Cards.
©2007 Microsoft Corporation About Privacy Trademarks
Account | Help C
Done
Internet
100%
Windows CardSpace (en-GB)
Choose a card to send to: Microsoft Corporation
To see or edit card data before you send it, select a card, and then click Preview. To create a new card, click Add a card.
Tasks
Duplicate card
Delete card
Add a card
Back up cards
Restore cards
Preferences
Delete all cards
Disable Windows CardSpace
Which card should I send?
Help
Learn more about this site
Cards you've sent to this site:
Bill's
Your other cards:
Add a card
Preview
Send
Cardspace

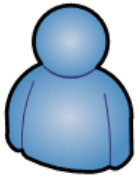
Web Infrastructure

WS-*

Author: Prof Bill Buchanan



Identity



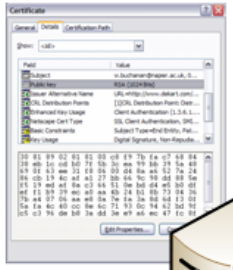
User

Identity selector

SAML (Security Assertion Markup Language)
Or Custom

WS-Security Policy
WS-Security

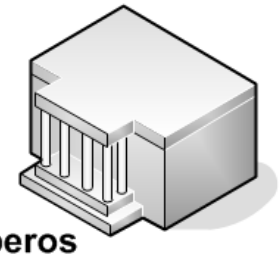
Relying Parity (RP)



Security Token Service (STS)

Identity Provider (IP)

X509 Certificate



Kerberos

PKI Server
Digital Certificate Granter (Verisign)

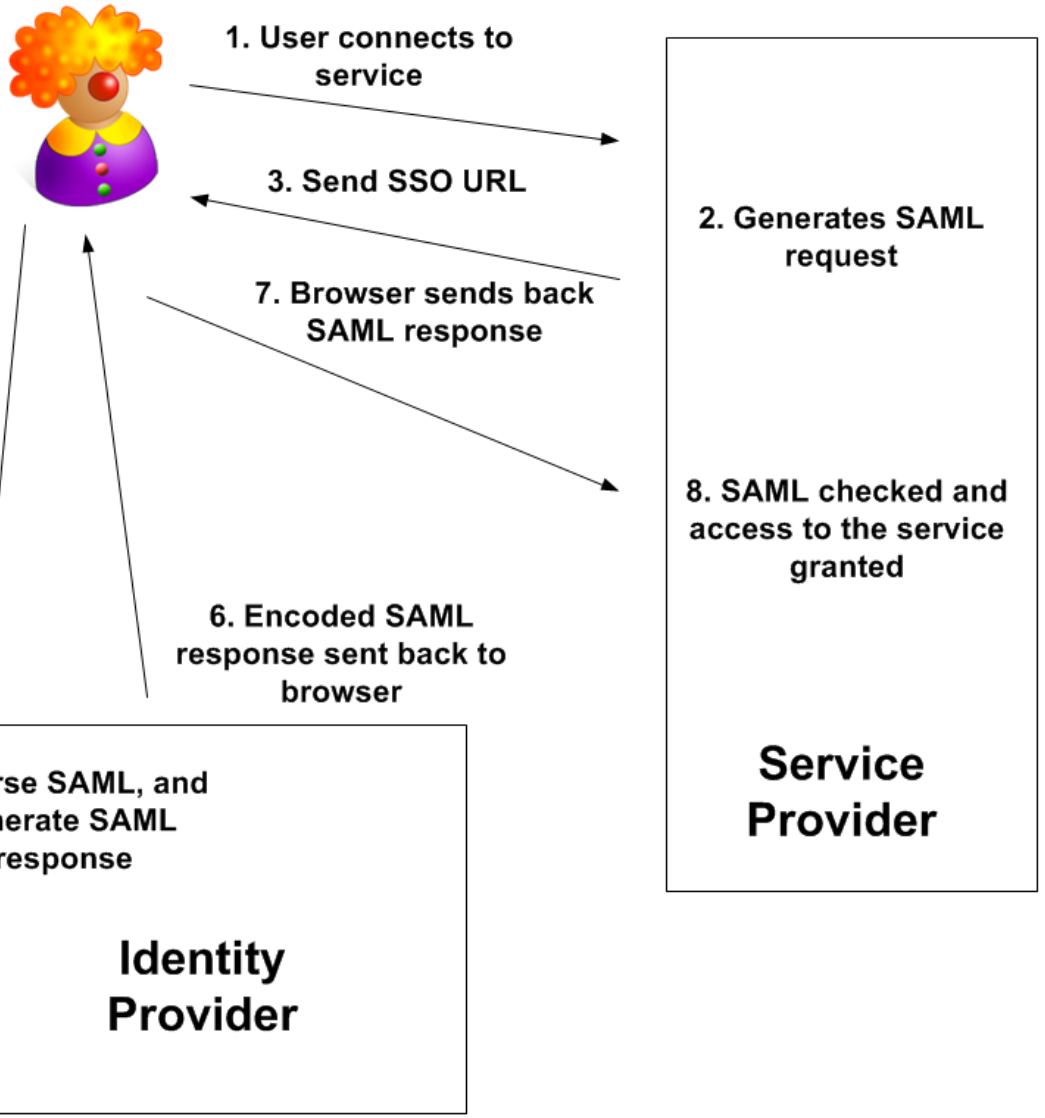
Open XML standards:
WS-*:-
WS-Trust, WS-Metadata Exchange Framework

Author: Prof Bill Buchanan

```
1 <Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
 2    IssueInstant="2003-04-17T00:46:02Z" Version="2.0"
 3    xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
 4    <Issuer>
 5       example.com
 6    </Issuer>
 7    <Subject>
 8       <NameID
 9         Format=
10         "urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
11         Alice@example.com
12       </NameID>
13       <SubjectConfirmation
14         Method="urn:oasis:names:tc:SAML:2.0:cm:sender-vouches"/>
15    </Subject>
16    <Conditions NotBefore="2003-04-17T00:46:02Z"
17                NotOnOrAfter="2003-04-17T00:51:02Z">
18       <AudienceRestriction>
19          <Audience>
20             example2.com
21          </Audience>
22       </AudienceRestriction>
23    </Conditions>
24    <AttributeStatement>
25       <saml:Attribute
26    xmlns:x500=
27      "urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
28    NameFormat=
29      "urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
30    Name="urn:oid:2.5.4.20"
31    FriendlyName="telephoneNumber">
32          <saml:AttributeValue xsi:type="xs:string">
33                +1-888-555-1212
34          </saml:AttributeValue>
35       </saml:Attribute>
36    </AttributeStatement>
37 </Assertion>
```

- **SAML Assertions**. These are: **Authentication** assertions (which assert that the user have proven their identity); **Attribute** assertions (which contains information about the user, such as when their limits are); and **Authorization** decision assertions (these define when the user can actually do).
- **Protocol**. This defines method that SAML uses to get gets assertions, such as using SOAP over HTTP (which is the most common method at the present).
- **Binding**. This defines how SAML message are exchanged, such as with SOAP messages.

Author: Prof Bill Buchanan

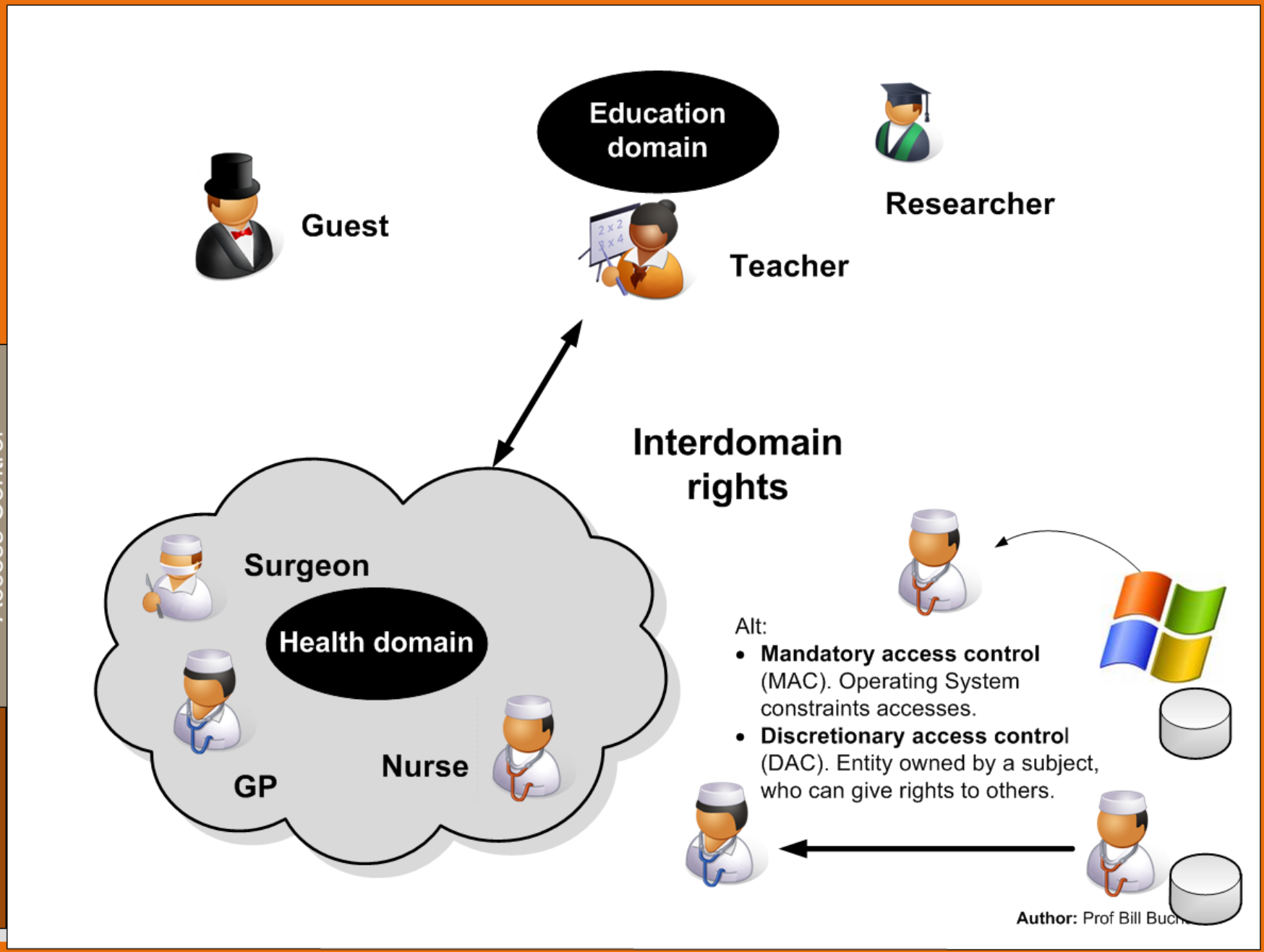


Author: Prof Bill Buchanan

Web Infrastructure

Access Control

Author: Prof Bill Buchanan


Education domain
Guest
Researcher
Teacher
Interdomain rights
Surgeon
Health domain
GP
Nurse
Alt:
• Mandatory access control (MAC). Operating System constraints accesses.
• Discretionary access control (DAC). Entity owned by a subject, who can give rights to others.
Author: Prof Bill Buch

Web Infrastructures

- Provide an overview of Web-based architectures, especially in authentication and access control.
- Define key protocols involved in next generation Web-based infrastructures, such as Kerberos and SOAP over HTTP.
- Define scalable authentication infrastructures and protocols.
- Investigate scaleable and extensible architectures, including using LDAP.

Author: Prof Bill Buchanan