Flexibel beheer van gedistribueerde sociale netwerksites met FOAF+SSL

Kris Van den Bergh

December 2009

Samenvatting

In dit artikel wordt FOAF+SSL besproken, een eenvoudig protocol dat toelaat om zich te authenticeren aan de hand van slechts een URI in bestaande browsers — zonder dat de eindgebruiker zijn accountgegevens hoeft te onthouden. Het is opgebouwd rond op grote schaal ingezette technologieën zoals HTTP, SSL/TLS certificaten en Semantisch Web gebaseerde FOAF netwerken. Deze zullen samengevat worden, gevolgd door een gedetailleerde beschrijving van het authenticatie mechanisme. Tenslotte worden beperkingen op het niveau van beveiliging aangehaald en wordt er gekeken naar gelijkaardige initiatieven, waarbij de vergelijking wordt gemaakt met OpenID in het bijzonder.

1 Inleiding

Hoewel menig applicatie op het web sociaal aan het worden is, zijn deze nog steeds zogenaamde "walled gardens" [1]. De gebruiker is genoodzaakt om een afzonderlijke account en bijhorende login gegevens te hebben voor elke organisatie waarmee interactie is. Deze web diensten vereisen authenticatie aan de hand van gecentraliseerde systemen. Hierdoor wordt de gebruiker genoodzaakt om zich telkens weer opnieuw te registreren. Bijgevolg moet men voor elke sociale netwerksite informatie bijhouden en bijwerken op verschillende plekken.

Het inherente probleem van het huidige wereldwijde web is dat informatie niet met elkaar verbonden is, met data silo's als gevolg en een gefragmenteerde identiteit op het web [2]. Immers, als men nu op Facebook is kan men niet naar een vriend verwijzen op LinkedIn. Het Semantisch Web, en het Linked Data initiatief in het bijzonder, wenst hier verandering in te brengen. De beveiligingsaspecten bij interactie op een gelinkt web zijn hier niet te overzien [3, 4, 5].

FOAF+SSL probeert dit op te lossen. Het eenvoudige protocol is gebouwd rond een aantal gangbare web technologieën, en laat toe om op basis van een betrouwbare one-click sign-on toegang te krijgen tot websites. SSL technologie wordt ingezet om beveiligde wachtwoordloze authenticatie te bewerkstelligen en FOAF/RDF technologie wordt aangewend om gebruikersprofielen te beheren. Het resultaat is een veilig, open en gedistribueerd authenticatie mechanisme, hetwelk de mogelijkheid biedt om een gebruiker te authenticeren op basis van een URI, maar evengoed in staat is om de autorisatie af te handelen op basis van gedistribueerde eigenschappen van de gebruiker, zoals bijvoorbeeld zijn positie in een sociaal netwerk [6, 7].

In Sectie 2 worden de onderliggende technologieën van FOAF+SSL verduidelijkt. Sectie 3 doorloopt het eigenlijke protocol in detail, beschrijft beperkingen en maakt de vergelijking met gerelateerde protocollen. Tenslotte worden er in Sectie 4 nog enkele bedenkingen geformuleerd.

2 Onderliggende technologieën en concepten

In deze sectie worden de onderliggende technologieën en beveiligingstechnieken beschreven die FOAF+SSL vormen.

2.1 Restful Web Architecture

Representational State Transfer (REST) is een architectuur voor het bouwen van grootschalige gedistribueerde informatie netwerken, waarvan het Wereldwijd Web de meest gekende toepassing is. Om tot zo'n netwerk te komen moeten alle afzonderlijke deeltjes onafhankelijk van elkander kunnen bestaan, zonder al te veel centrale coördinatie, en elk resource/object moet eenvoudig kunnen refereren naar de andere. Hierbij worden er universele namen gebruikt, ook gekend als Universal Resource Identifiers (URI's). Deze gebruiken het HTTP protocol als dereferentie mechanisme. Via canonieke methodes kunnen de resources gemanipuleerd worden. Door het zenden van een GET verzoek naar een object op een gegeven URL wordt er een *representatie* van de resource teruggegeven. Een resource kan gecreëerd, gewijzigd en verwijderd worden door respectievelijk de POST-, PUT- en DELETE methoden.

2.2 Semantisch Web

Het traditionele web bestaat uit documenten waarin hyperlinks slechts refereren naar andere documenten. Het Semantisch Web daarentegen is een uitbreiding op het huidige World Wide Web (WWW), waar semantische betekenis gegeven wordt aan de inhoud van documenten aanwezig op het web, wat het uitwisselen en interpreteren van informatie vereenvoudigd. In het Semantisch Web wordt het namelijk mogelijk om URL's te koppelen en referenties te leggen op eender welk criteria.

Een Semantisch Web document is een serialisatie van een zogenaamde graf van directe relaties tussen objecten. Elke relatie bestaat uit drie delen en wordt uitgedrukt in de vorm van onderwerp-relatie-voorwerp. In Semantisch Web termen wordt dit ook wel *triple* genoemd. Omdat URL's niet leesbaar zijn wordt in dit artikel de N3 @prefix geïntroduceerd. De volgende voorvoegsels worden ingevoerd:

```
@prefix log:  <http://www.w3.org/2000/10/swap/log#> .
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
@prefix cert:  <http://www.w3.org/ns/auth/cert#> .
@prefix rsa:  <http://www.w3.org/ns/auth/rsa#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix romeo:  <https://romeo.example/#> .
@prefix jult:  <https://juliet.example/#> .
@prefix :  <> .  # speciale woordenschat in dit artikel
```

De stelling dat "Romeo is een persoon", kan op basis van de N3 notatie geschreven worden als volgt: `romeo:i a foaf:Person`. Elk kenmerk is beschikbaar onder de vorm van een URL. Bijkomstige informatie over een kenmerk kan bekomen worden door middel van dereferentie — een HTTP GET op `<http://xmlns.com/foaf/0.1/Person>` geeft een representatie van dit object weer hetgeen zowel door mensen als machines geïnterpreteerd kan worden.

Bovendien kan elke geretourneerde representatie geïnterpreteerd worden als een graf van relaties, dewelke in N3 geïsoleerd worden door ze tussen accolades te plaatsen `{ }`. De relatie die een resource mapt op een graf, beschreven door een document dat bekomen is op basis van de toegepaste canonische dereferentie methode, wordt gedefinieerd als de `:semantics` relatie. Het volgende kan dan verklaard worden:

(P1)
```
romeo:i :semantics { romeo:i a foaf:Person;
                         :HasPrivateKeyFor PubKey;
                     foaf:name "Romeo";
                     foaf:knows jult.me; }
```

Deze graf kan ook gebruikt worden om regels te formuleren, als we de hierboven `:semantics` relatie in termen van de `log:semantics` eigenschap definiëren, hetwelk een document relateert aan een graf, en de `:representation` dat een resource relateert aan één van haar representaties (de door `?` voorafgegane variabelen zijn universeel gekwantificeerd):

(D1)
```
?resource :representation ?doc . ?doc log:semantics ?graph .
=>  ?resource :semantics ?graph .
```

De `log:` namespace betreft logica of formules[1]. In het bijzonder linkt de `log:includes` eigenschap een onderwerp graf met een object graf door te beweren dat de ene een subset is van de andere graf; de `log:implies` eigenschap, ook geschreven als `=>`, kan beschouwd worden als de motivering gebaseerd op eerste-ordelogica.

2.3 FOAF en het Web van vertrouwen

Een veelgebruikte toepassing van het Semantisch Web is *FOAF* (Friend-of-a-Friend)[2]. Het beschrijft relaties tussen mensen en andere agenten in RDF. Op die manier bekomt men een sociaal netwerk dat de relaties vastlegt tussen iemand en zijn netwerk van vrienden. Een FOAF-profiel verwijst naar een URI (Web ID) op hetwelk een agent dan een GET kan doen. Iemand die door Romeo de `romeo:i` URL is gegeven, en de `:semantics` is gaan halen heeft gegronde redenen om te geloven dat deze informatie correct is. De gebruiker kan dan `romeo:i` aan zijn FOAF-profiel toevoegen en aangeven dat `:me foaf:knows romeo:i`. De waarde van het resulterende netwerk neemt exponentieel toe als het aantal mensen dat er een FOAF-profiel op na houden en naar elkaar linken stijgt. Deze vicieuze cirkel is ook wel bekend als de wet van Metcalfe.

De `foaf:knows` relatie is natuurlijk niets zonder vertrouwen. Er moeten daarom reputatie netwerken gecreëerd worden die dit vertrouwen beschrijven. In het geval van FOAF+SSL gebeurt dit via signatures en cryptografische sleutels om zo tot een Web van vertrouwen (Web of Trust)[3] te komen. Het Web van vertrouwen wordt gebruikt om de authenticiteit van de relatie tussen een publieke sleutel en een persoon te bepalen.

2.4 Public Key cryptografie

Public key cryptografie laat twee participanten toe om op een veilige manier te communiceren, zonder dat ze daarbij een geheim moeten delen. Dit gebeurt door gebruik te maken van een uniek paar sleutels. De ene sleutel, ook wel *publieke sleutel* genoemd, mag verspreid worden, en de andere

[1] De `log:` namespace wordt beschreven op `http://www.w3.org/DesignIssues/N3Logic`
[2] Gedefinieerd op `http://xmlns.com/foaf/0.1`
[3] Web of Trust wordt beschreven op `http://en.wikipedia.org/wiki/Web_of_trust`

sleutel, oftewel *private sleutel*, moet alleen worden bijgehouden door zijn eigenaar. Dit is in contrast met symmetrische cryptografie, alwaar beide participanten dezelfde kennis over één geheime sleutel moeten hebben, zowel voor encryptie als decryptie.

Public key cryptografie gaat uit van het vermoeden dat het onhaalbaar is om een private sleutel te bekomen die bij een gegeven publieke sleutel hoort, omdat deze operatie teveel rekenkracht vereist. Bovendien zullen twee individuen nooit dezelfde paar sleutels genereren. We zouden hiervoor een inverse functioneel kenmerk `:hasPrivateKeyFor` kunnen definiëren als volgt:

(D2)
```
:hasPrivateKeyFor a owl:InverseFunctionalProperty;
  rdfs:domain foaf:Agent;
  rdfs:range cert:PublicKey .
```

Dankzij de duale natuur van de publieke en private sleutel paren, zijn er twee onderscheidende acties mogelijk:

1. Bij *encryptie* wordt er een gecodeerd plat bericht gegenereerd met behulp van de publieke sleutel, zodat zij enkel gedecodeerd kan worden op basis van de corresponderende private sleutel. Dit wordt *obfuscation* genoemd.

2. Bij *signing* wordt er een digitale handtekening geassocieerd met een bericht; deze handtekening wordt gegenereerd via de private sleutel. De authenticiteit en integriteit van het bericht kan worden geverifieerd met behulp van de bijbehorende publieke sleutel.

Een *public key certificaat* is een handgetekende combinatie van een publieke sleutel en gerelateerde informatie bij deze sleutel. Een dergelijk certificaat kan zelf handgetekend zijn (door gebruik te maken van de private sleutel die overeenstemt met de aanwezige publieke sleutel) of door een derde. De partij die het certificaat tekent onderschrijft de inhoud. Het vertrouwen in deze partij kan de inhoud van het certificaat doen gelden [8].

Twee verschillende architecturen zijn ontwikkeld die gebruik maken van het handtekenen van public key certificates: de hiërarchische Public Key Infrastructure en het cryptografische Web van vertrouwen. In beide architecturen wordt een omgeving initieel geconfigureerd met een betrouwbare set van certificaten, oftewel *vertrouwensankers* (trust anchors). Als er een ongekend certificaat aangeboden wordt, zal de omgeving verifiëren of zijn authenticiteit geldig is door een pad of ketting te volgen tussen het certificaat en één van de vertrouwensankers. Een certificaat wordt betrouwbaar geacht als en alleen als het gesigneerd is door een certificaat dat reeds vertrouwd is. Indien nodig wordt de operatie herhaald om een pad te bouwen door middel van intermediaire certificaten, van het welk de vertrouwensrelatie transitief is.

Het hiërarchische Public Key Infrastructure model en het cryptografische Web van vertrouwen verschillen hoofdzakelijk op de manier waarop de certificaten worden gedistribueerd en intermediaire certificaten worden vertrouwd [9].

2.5 PKI en het hiërarchische vertrouwensmodel

Het *Internet X.509 Public Key Infrastructure* (PKI) is een hiërarchisch model voor het distribueren en vertrouwen van certificaten. In dit model worden certificaten handgetekend door een *certification authority* (CA). X.509 certificaten incorporeren een *Subject Distinghuished Name* (Subject DN), hetgeen het onderwerp van het certificaat identificeert, en een *Issuer Distinguished Name* (Issuer DN), hetgeen de uitgever van het certificaat identificeert — de entiteit die het certificaat tekent.

Deze structuur bouwt een hiërarchische boom van het root CA certificaat tot de eind-entiteit (bijv. client of server) certificaten.

De meeste web-browsers en operating systems bieden een standaard lijst aan van CA certificaten die ze de gebruikers impliciet laten vertrouwen. Deze lijst kan echter meestal door de gebruiker aangepast worden.

2.6 Cryptografische Web van vertrouwen

Het cryptografische Web van vertrouwen is een vorm van public key infrastructure waar elke participant vertrouwen kan doen gelden in eender welke andere participant, zonder dat er een specifieke hiërarchie speelt.

Het model achter Web van vertrouwen wordt gebruikt door PGP; PGP public key is in feite een public key certificate, vermits het ook additionele informatie (zoals een e-mail adres) bevat en handgetekend is om authenticiteit te doen gelden. Zo'n certificaat is zelf-handgetekend, maar kan ook additionele handtekeningen bevatten — deze door dewelke de associatie tussen de sleutel en de additionele informatie wordt vertrouwd.

In PGP zijn de vertrouwensankers de gebruiker zijn eigen certificaat en de certificaten die de gebruiker vertrouwt, sommige afkomstig van zogenaamde *trusted introducers* (personen met wie vertrouwen transitief is). Het aantal handtekeningen dat een certificaat heeft, reflecteert de connectiviteit met andere partijen. Hoe meer handtekeningen een certificaat heeft, hoe waarschijnlijker het is dat een derde in staat zal zijn om een pad te vinden via een trusted introducer.

2.7 SSL authenticatie

Het meest ingezette protocol voor beveiligde communicatie tussen een user agent en een web server is Transport Layer Security (TLS)[4]; in HTTP applicaties wordt het aangeduid door de `https` prefix in URL's.

Tijdens de SSL handshake in het begin van een SSL connectie verkrijgt de client een X.509 certificaat van de server. Op dit moment is de client afhankelijk van zijn vertrouwensankers om het te verifiëren. Als de handshake beëindigd is, kan de communicatie op een beveiligde manier verder worden gezet; de enige andere partij die in staat is om de communicatie te lezen moet de corresponderende private sleutel hebben voor het server certificaat.

Er bestaat een variant op de handshake procedure in hetwelk de client verzocht wordt of genoodzaakt is om de server een certificaat te presenteren, zodat de server de client kan authenticeren op basis van dezelfde verificatie methode als hierboven beschreven. Dit is de paradigma shift die FOAF+SSL hanteert [10].

Vanuit een Semantisch Web perspectief beschrijven we nu hoe vertrouwen in een certificaat geëvalueerd wordt. We beschrijven de redenatie van een server, `S`, voor het authenticeren van een client, `:client`, die verzoekt. Server `S` beschikt over een set van vertrouwde CA's. `S` zou graag verklaren dat `issuerDN` een vertrouwde CA is:

(P2)
```
issuerDN a :TrustedCA
     :hasPrivateKeyFor CAKey .
```

[4]Het TLS protocol wordt beschreven op `http://tools.ietf.org/html/rfc5246`

De `CAKey` is een `cert:PublicKey` en wordt meestal geïdentificeerd door een aantal inverse functionele eigenschappen, dewelke een OWL2 sleutel[5] vormen. CA sleutels kunnen uniek geïdentificeerd worden.

`S` verzoekt dat de `:client` een handgetekend certificaat presenteert waarvan CA weet van heeft. `S` ontvangt `_:certDoc` met de volgende semantics:

(P3)
```
_:certDoc :semantics _:certSemantics .
_:certSemantics =  <> dc:created issuerDN;
           foaf:primaryTopic subjectDN .
               subjectDN :hasPrivateKeyFor pubKey .
               issuerDN :hasPrivateKeyFor CAKey .
```

De SSL handshake heeft verzekerd dat de client over de private sleutel beschikt:

(P4)
```
:client :hasPrivateKeyFor pubKey .
```

De client laat de inhoud van het certificaat gelden en bevestigt dat het handgetekend is door de issuer:

(P5)
```
:client :claims  _:certDoc :signature _:certSig;
                  _:certSig :signedWith CAKey
                        :sigString "XYZ SIG"
```

`S` kan laten gelden dat, na de verificatie, `_:certDoc` handgetekend is door de private sleutel te gebruiken corresponderende met `CAKey`:

(P6)
```
_:certDoc :signature [ :signedWith CAKey ] .
```

Het bewijs dat een document handgetekend is door `P`, is laten gelden dat `P` aanspraak maakt op de inhoud:

(D3)
```
 ?P :hasPrivateKeyFor ?key .
?doc :signature [ :signedwith ?key ]
 =>  ?P :claims [ is :semantics of ?doc ]  .
```

Gegeven de verificatie van de handtekening P6, de inhoud van het certificaat P3, en het bewijs dat een document handgetekend is en aanspraak maakt op de inhoud D3, kan `S` laten gelden dat:

(P7)
```
issuer :claims _:certSemantics
```

Iemand vertrouwen betekent vertrouwen wat ze beweren. `S` vertrouwt TrustedCA's, vandaar:

(D4)
```
?ca :claims ?s . ?ca a :TrustedCA  =>  ?s a log:Truth  .
```

Uit P2, P7 en D4 kan `S` concluderen dat:

(P8)
```
subjectDN :hasPrivateKeyFor pubKey
```

[5] http://www.w3.org/TR/owl2-syntax/#Keys

Uit P4, vekregen door de SSL handshake, P8 en de definitie D2 van `:hasPrivateKeyGot` als een `owl:inverseFunctionalProperty` kunnen we afleiden dat:

(P9) `client owl:sameAs subjectDN`

Vanaf dan heeft de server `S` de `:client` geauthenticeerd als zijnde Distinguished Name (DN). Als de `:client` dan verzoekt om toegang te krijgen tot resource `R`, kan de server uitzoeken of dat deze DN geautoriseerde toegang heeft tot `R`.

Het probleem met DN's is dat, hoewel ze in de vorm van een URI kunnen worden gegoten, het dereferentie mechanisme voor DN's niet zo globaal is als de manier waarop URL's dat zijn. Voor ongeveer dezelfde reden kan data in LDAP servers geen velden hebben die refereren naar resource in eender welke andere LDAP server. Bijgevolg wordt het huidig gebruik van client certificaten gelimiteerd tot het gebruik over enkele domeinen.

Daarom, als er toegang tot `R` wordt toegekend op basis van een aantal regels, waar er meer informatie moet ontdekt worden om een verdere beslissing te maken, dan kan de DN geen globale oplossing bieden.

Het vermogen om globaal te linken is een essentiële bouwsteen dat vereist is om een globaal sociaal netwerk te construeren. De volgende sectie toont aan hoe FOAF+SSL dit probleem oplost.

3 Het FOAF+SSL protocol

Het FOAF+SSL protocol maakt gebruik van SSL, maar gebruikt een ander model van vertrouwen dan PKI om certificaten te verifiëren.

Bij het beschermen van een service is het belangrijk om onderscheid te maken tussen authenticatie en autorisatie. Authenticatie is de actie die de identiteit van de eindgebruiker verifieert. Autorisatie bestaat uit het toegang verlenen of weigeren om een operatie uit te voeren op een gegeven resource, gebaseerd op de identiteit vastgelegd tijdens authenticatie.

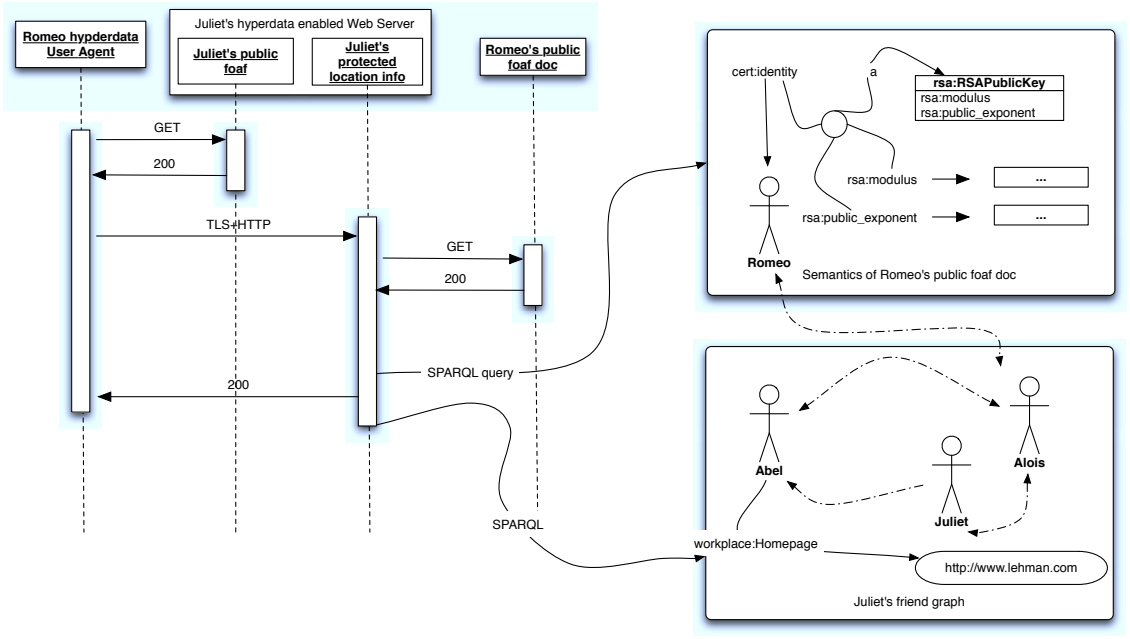
FOAF+SSL stelt een server in staat om een client te authenticeren op basis van een simpele URL [12]. Deze URL kan verder ook direct gebruikt worden voor autorisatie, of om op zoek te gaan naar verdere informatie in het web van Linked Data[6], ten einde te beslissen of de URL referent voldoet aan de beperkingen die opgelegd zijn om toegang te krijgen tot een bepaalde resource [11].

Het FOAF+SSL authenticatie protocol bestaat uit de volgende stappen, zoals geïllustreerd in Figuur 1:

1. Een client raadpleegt een publieke HTTP resource dewelke refereert naar een beschermde resource, bijvoorbeeld `<https://juliet.example/location>`

2. De client, `romeo:i`, doet aan dereferentie van deze URL

[6] Meer informatie over Linked Data op `http://www.linkeddata.org/`



Figuur 1: Het FOAF+SSL sequentie diagram.

3. Gedurende de SSL handshake verzoekt de server om een certificaat van de client. Omdat FOAF+SSL niet beroep doet op CA's, mag het vragen naar eender welk certificaat. De client zendt Romeo's certificaat (hetwelk zelg-gesigneerd kan zijn) dat de publiek sleutel (zie "Subject Public Key Info" in Lijst 1) en een Subjective Alternative Name URI (zie "X509v3 extensions" in Lijst 1") bevat. Omdat de SSL handshake succesvol is geweest, weet Juliet's server dat Romeo's client over de private sleutel beschikt die correspondeert met de publieke sleutel van het certificaat.

Listing 1: Uitreksel van een tekst representatie van een FOAF+SSL certificaat

```
Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        RSA Public Key: (1024 bit)
                Modulus (1024 bit)
                        00:b6:bd:6c:e1:a5:ef:51:aa:a6:97:52:c6:af:2e:
                        71:94:8a:b6:da:9e:5a:5f:08:6d:ba:75:48:d8:b8:
                        [...]
                Exponent: 65537 (0x10001)
X509v3 extensions:
        X509v3 Subject Alternative Name:
                        URI:https://romeo.example/#i
```

4. Juliet's server derefereert het Subject Alternative Name URI gevonden in het certificaat en eindigt met een subset van D1.

5. De `log:semantics` van het document worden opgevraagd voor informatie aangaande de publieke sleutel aanwezig in het X.509 certificaat. Dit kan gerealiseerd worden op basis van een

SPARQL query zoals in Lijst 2. Als de publieke sleutel van het certificaat overeenkomt met deze van het FOAF file, dan is `romeo:i` geauthenticeerd.

Listing 2: SPARQL query om public key informatie te bekomen uit een FOAF profiel. Elke set certificaat data bestaat uit een agent en een RSA sleutel met een exponent en modulus.

```
SELECT ?modulus ?exp WHERE {
        ?key cert:identity <https://romeo.example/#i>;
        a rsa:RSAPublicKey;
        rsa:modulus [ cert:hex ?modulus; ];
        rsa:public_exponent [ cert:decimal ?exp ] . }
```

6. Als de authenticatie voltooid is, kan de positie van `romeo:i` in Juliet haar social graph bepaald worden. Juliet's server kan deze informatie bekomen door over het web te navigeren vertrekkende van haar FOAF file, of langs andere wegen.

7. Authenticatie is afgehandeld, autorisatie kan nu plaatsvinden.

3.1 Authenticatie Logica

Deze sectie volgt dezelfde redenatie als Sectie 2.7, waarbij een web server `S` zich probeert te authenticeren met een `:client`.

Op het einde van stage 3 in het FOAF+SSL sequentie diagram heeft `S` het client certificaat veilig ontvangen. Daar het zelf-gesigneerd is (of handgetekend door een onbekende partij), is de semantiek gedeeltelijk anders. `S` is echt uitsluitend geïnteresseerd in de URI identificaties die naar het onderwerp refereren — de beperkingen van de DN's omzeilend. Daarnaast, vermits de client het laat doen gelden, weet `S` dat:

(P10)
```
:client :claims  <> dc:created romeo:i;
         foaf:primaryTopic romeo:i .
      romeo:i :hasPrivateKeyFor pubKey  .
```

`S` mag slechts weten van `romeo:i` wat het verzameld heeft op basis van de SSL handshake:

(P11)
```
:client :hasPrivateKeyFor pubKey .
```

Wanneer er iemand een bewering maakt moet men ook gevolg geven aan de logische consequenties van de bewering, inclusief deze afkomstig van nieuwe feiten. Vandaar, iemand die een bewering maakt `:mustAgree` met de conclusies uit de unie van wat we betrouwbaar achten, wat de client gelooft, en de redenatie regels:

(D5)
```
?client :claims ?clientGraph .
( ?clientGrph secureFactGraph owlReasoningRules )
   log:conjunction  [ log:conclusion ?C ]
=>  ?client :mustAgree ?C
```

Vandaar, vertrekkende van P10, P11, en D2, mag `S` concluderen dat `:client` het eens zou moeten zijn dat het `romeo:i` betreft. Dit zou geen verrassing mogen zijn, vermits dit inderdaad is wat men aanneemt als iemand de intentie heeft om een certificaat te zenden.

(P12)
```
:client :mustAgree [ log:includes  romeo:i = :client  ] .
```

Vermits `:client` zegt dat het `romeo:i` is, accepteert het om geïnspecteerd te worden via `romeo:i`. Daar `romeo:i` een URL is, kan `S` aan derefentie doen en zo P1 ontdekken. Dan, op basis van P1, P11, D2, en D5:

(P13) `romeo:i :mustAgree [ log:includes  romeo:i = :client  ] .`

Met andere woorden, zowel `romeo:i` als `:client` moeten het erover eens zijn dat, gegeven wat `S` weet, dat `romeo:i owl:sameAs :client`. In het bijzonder kan `romeo:i` deze stelling niet verwerpen vermits `romeo:i` zelf P1 heeft opgelegd. Bijgevolg, als `S` geautoriseerd is om `R` aan te bieden aan `romeo:i`, kan `S` dit eveneens aanbieden aan `:client` .

3.2 Beperkingen en risico's

Een voorname beperking is dat client side public key certificaten trager zijn dan geanticipeerd [13]. Een andere mogelijke tekortkoming is integratie met de enterprise wereld. Certifcaten omvatten immers veel meer dan beveiligde communicatie alleen. Op enterprise niveau betekent dit dat integratie met onder andere IPSec, EAP-TLS voor WiFi en LAN switching, EFS voor confidentiële bestanden en DRM voor MS Office bestanden mogelijk moet zijn. [14].

FOAF+SSL mitigeert een aantal beveiligingsrisico's die zich kunnen voordoen bij een gedistribueerd systeem en dewelke technisch van aard zijn. Vermits er meer dan één server gebruikt wordt met FOAF profielen, zou bij een *Denial of Service* aanval van één server het netwerk niet stabiliseren. *ARP/TCP Spoofing* wordt opgelost door het gebruik van certificaten die geverifieerd worden. Het risico van het kraken de encryptie wordt geminimaliseerd door het gebruik van 1024 bit sleutels [15]. Hoewel SSL/TLS ingezet wordt om een beveiligde verbinding te bewerkstellingen, is het momenteel nog steeds mogelijk om een *man-in-the-middle* aanval uit te voeren [16].

3.3 Gerelateerd werk

Huidige *federated identity management systems* (e.g. SAML, Liberty ID-FF) missen controle door de eindgebruiker en het is daar dat FOAF+SSL op inspeelt. Nochtans, de afgelopen jaren hebben OpenID, LID, CardSpace, Higgins en anderen dit reeds ingevuld. Het voornaamste verschil is dat FOAF+SSL uitgaat van een graf — uiteraard dient de gefragmenteerde identiteit afkomstig uit verschillende silo's gedecentraliseerd en geaggregeerd te worden zoals [4]. Deze zogenaamde *user-centric identity schemes* baseren zich op core World Wide Web specificaties, maar adresseren ook andere vereisten zoals anti-phishing- en privacy bescherming terwijl traditionele federatie systemen minder deze B2C items discussieerden. Hoewel grote spelers als Microsoft, IBM en Novell reeds *user-centric* oplossingen aanbieden, is er echter nog veel werk betreffende interoperabiliteit vooraleer dit een mainstream praktijk wordt in de business wereld [17].

Vermits OpenID[7] ook gebruik maakt van URL's als identificatie methode, net zoals FOAF+SSL vertrouwt op URI's die FOAF data dragen dewelke gederefereerd kunnen worden, wordt hier de vergelijking gemaakt. OpenID maakt ten eerste geen verder gebruik van de informatie in de OpenID resource; het wordt enkel gebruikt om de autorisatie service te vinden. Bijgevolg vereist OpenID veel meer connecties om een identiteit vast te stellen — 6 in tegenstelling tot 2 — en verzaakt aan een RESTful design, waardoor de voordelen van een netwerk architectuur afwezig zijn [18]. FOAF+SSL heeft het grote voordeel dat het direct linkt naar een FOAF-profiel van een gebruiker.

[7] http://www.openid.net/

Bijgevolg kan de informatie in de file gebruikt worden om een rule-based authenticatie te doen [11]. OpenID is slechts een gedecentraliseerde ID en heeft niet noodzakelijk linked data met zich geassocieerd.

Table 1: Vergelijking tussen FOAF+SSL en OpenID

	FOAF+SSL	OpenID
identificatie	browser onthoudt URL	gebruiker onthoudt URL
aantal connecties	minimum 1 – 2	6
informatie uitwisseling	elk attribuut is een URI	OpenID Attribute Exchange
uitbreidbaarheid	eenvoudig	ingewikkeld
vertrouwen	direct	indirect (via OpenID provider)
controle punt	gebruiker	service

De twee technologieën zijn niet incompatibel, en recent is reeds de brug gemaakt [19].

4 Conclusie

Hoewel FOAF+SSL zich nog in een immattuur stadium bevindt, biedt het een beveiligd en flexibel globaal authenticatie systeem. Dankzij de public key cryptografie, profiteert het van alle beveiligde voordelen die deze technologie met zich mee brengt. Daar FOAF+SSL Restful is en integreert met het Semantisch Web, kan het meer informatie ontdekken over een bepaalde entiteit door de Linked Data cloud te exploreren. In vergelijking met PKI, verwijdert FOAF+SSL de nood aan hiërarchische autoriteiten om identiteit te doen gelden, wat het veel flexibeler maakt. Vandaar dat dit mechanisme zich perfect leent voor de formatie en expansie van virtuele organisaties en gedistribueerde sociale netwerksites.

Referenties

[1] Vogelstein, F. (22.06.2009), "Great Wall of Facebook: The Social Network's Plan to Dominate the Internet — and Keep Google Out", *Wired Magazine*. Geraadpleegd op 5 december 2009 uit: `http://www.wired.com/techbiz/it/magazine/17-07/ff_facebookwall`

[2] Habib, M. C. (2007), "Managing Your Identity Online", *netConnect*, 15 oktober 2007.

[3] Au Yeung, C. & Liccardi, I. & Lu, K. & Seneviratne, O. & Berners-Lee, T. (2009), "Decentralization: The Future of Online Social Networking," In: *W3C Workshop on the Future of Social Networking*.

[4] Rowe, M. (2009), "Interlinking Distributed Social Graphs," In: *Proceedings of Linked Data on the Web Workshop 2009*.

[5] De Backer, C. & Moons, J. (2009), "Facebook en privacy"

[6] Story, H. & Harbulot, B. & Jacobi, I. & Jones, M. (2009), "FOAF+SSL: RESTful Authentication for the Social Web," In: *Trust and Privacy on the Social and Semantic Web workshop at ESWC 2009*.

[7] Jones, M. & Harbulot, P., & Story, H. (2009), "Flexible management of Virtual Organisations using FOAF+SSL", In: *UK e-Science All Hands Meeting 2009*.

[8] Harbulot, B. (07.03.2009), "Signing FOAF files: FOAF files as certificates", *Distributed Matter - Blog*. Geraadpleegd op 5 december 2009 uit: `http://blog.distributedmatter.net/post/2009/03/07/Signing-FOAF-files:-FOAF-files-as-certificates`

[9] Harbulot, B. (14.10.2008), "Comparing Web of Trust and Hierarchical PKI for FOAF+SSL", *Distributed Matter - Blog*. Geraadpleegd op 5 december 2009 uit: `http://blog.distributedmatter.net/post/2008/10/14/Comparing-Web-of-Trust-and-Hierarchical-PKI-for-FOAFSSL`

[10] Story, H. (03.03.2009), "The foaf+ssl paradigm shift", *The Sun Babelfish Blog*. Geraadpleegd op 20 november 2009 uit: `http://blogs.sun.com/bblfish/entry/the_foaf_ssl_paradigm_shift`

[11] Hollenbach, J. & Presbey, J. & Berners-Lee, T. (2009), "Using RDF Metadata To Enable Access Control on the Social Semantic Web," In: *Workshop on Collaborative Construction, Management and Linking of Structured Knowledge*.

[12] Story, H. (02.12.2008) "foaf+ssl: adding security to open distributed social networks", *The Sun Babelfish Blog*. Geraadpleegd op 20 november 2009 uit: `http://blogs.sun.com/bblfish/entry/foaf_ssl_adding_security_to`

[13] Javier, L. & Oppliger, R. & Pernul, G. (2005), "Why have public key infrastructure failed so far?", *Internet Research*, nr. 15, blz. 544 – 556

[14] Williams, P (2009). "B2B communication". *foaf-protocols – FOAF and protocols discussion*. Geraadpleegd op 12 december 2009 uit: `http://lists.foaf-project.org/pipermail/foaf-protocols/2009-December/001160.html`

[15] Grzonkowski, S et al. (2005), "D-FOAF-Security Aspects in Distributed User Managment System", In: *IEEE International Conference on Technologies for Homeland Security and Safety.*

[16] Zoller, T. (2009). "TLS & SSLv3 Vulnerabilities Explained", *G-SEC.* Geraadpleegd op 12 december uit: `http://www.security.nl/artikel/31551/1/TLSSSLv3_Renegotiation_Vulnerability.html`

[17] Kobielus, J. (2007), "User-Centric Identity Still In Early Development", *Business Communications Review*

[18] Story, H. (19.12.2008), "what does foaf+ssl give you that openid does not?", *The Sun BabelFish Blog.* Geraadpleegd op 3 november 2009 uit: `http://blogs.sun.com/bblfish/entry/what_does_foaf_ssl_give`

[19] Story, H. (19.11.2009), "OpenID hearts foaf+ssl", *The Sun Babelfish Blog.* Geraadpleegd op 20 november 2009 uit: `http://blogs.sun.com/bblfish/entry/http_openid4_me_openid_foaf`