Universitat Oberta de Catalunya

TFC

NoFeud: privacitat per a xarxes socials

memòria del projecte

Gener 2014

Adrià Massanet Bienvenido
adria.massanet@gmail.com

Consultora
Cristina Pérez Solà (perezsola@uoc.edu)

Resum

Paraules clau: javascript, criptografia, privacitat, xarxes socials

Aquest treball fi de carrera és un estudi-prototip de un model de privacitat per a xarxes socials on els missatges son visibles només per a qui comparteix un cert nombre d'amics en comú, fent servir els algorismes RSA i la compartició de secrets de Shamir. Per a realitzar el prototip s'ha creat un pugin de Google Chrome per la part de client i per la part de servidor un servei en NodeJS i l'emmagatzematge en una base de dades NoSQL mongoDB, que son tecnologies emergents dintre del desenvolupament web ; per tant, tot el codi ha sigut escrit en javascript.

ÍNDEX

Introducció

Feudal security consolidates power in the hands of the few. These companies [like Google, Apple, Microsoft, Facebook etc.] act in their own self-interest. They use their relationship with us to increase their profits, sometimes at our expense. They act arbitrarily. They make mistakes. They're deliberately changing social norms. Medieval feudalism gave the lords vast powers over the landless peasants; we're seeing the same thing on the Internet.

Bruce Schneier

Justificació i context del projecte

Avui en dia, pràcticament tota la informació que utilitzem per comunicar-nos socialment es realitza mitjançant plataformes que basen, principalment, el seu model de negoci en generar extractes d'aquesta informació per a personalitzar anuncis o directament per vendre-la a tercers. Per això, aquestes empreses poden oferir els seus serveis sense que comporti un pagament per part dels seus usuaris.

Aquest contracte, però, presenta una sèrie de problemes importants respecte a la retenció d'informació:

- La ètica d'aquestes empreses depenen directament dels seus canviants consells d' administració i no són sota cap control per part dels seus usuaris, per tant es difícil assegurar una continuïtat en els aspectes relacionats amb la privacitat. [1]
- No hi ha cap regulació al respecte de com es gestiona aquesta privacitat, si bé hi ha en esborrador una regulació a nivell europeu[2][3]

[1] http://articles.washingtonpost.com/2013-09-05/business/41788623_1_user-privacy-data-use-policy-facebook-s

[2] http://www.theguardian.com/news/datablog/2013/aug/12/europe-data-protection-directive-eu

[3] http://www.europarl.europa.eu/meetdocs/2009_2014/documents/libe/dv/briefingnote_/briefingnote_en.pdf

- Les ultimes revelacions d'Edward Snowden, respecte a la capacitat d'EEUU, tant de demanar dades directament al aquestes empreses, com de perpetrar violacions de l' intimitat de qualsevol ciutadà del món, irrompent en sistemes de manera no gaire ètica i intervenint en els processos d'estandardització per introduir backdoors en els protocols de seguretat, fan que la protecció de la informació estigui menys garantida que mai.[4]
- La teoria del "present continu" diu que la informació ja no es borra i el present actual serà accessible per sempre en el futur, de manera que podem parlar d'una "nova història". Això fa que les futures eines d'explotació (com ara comença amb el Big Data) possiblement podran explotar tota la informació guardada i treure perfils dels ciutadans sense que aquests puguin protegir la seva intimitat mai més.
- Si bé és cert, que les xarxes socials permeten la creació d'un usuari que no té perquè correspondre a la realitat, el fet que la identitat és un aspecte social, com més connexions tingui és més fàcil desambiguar aquest anonimat.
- Actualment sembla que no hi ha masses alternatives per fer servir les xarxes socials de manera segura per part del proveïdor. La implementació de criptografia feta pels proveïdors està implementada en javascript i HTML5 i això no es segur en el sentit que si el proveïdor modifica els scripts que envia el client, pot recuperar-ne les claus, sense cap advertència a l'usuari.

El que proposa aquesta TFC és una prova de concepte per intentar dotar mecanismes al navegador web perquè tant l'usuari com els websites pugin gestionar informació xifrada.

Estat de l'art

S'ha fet una cerca d'informació per la web sobre l'estat respecte a l'ús de la criptografia en els navegadors:

- Sobre CryptoAPI, actualment hi ha en marxa un esborrador de la WebCryptoAPI[5] () per a navegadors. Sembla que s'està començant a implementar per a Firefox[6] (), així com una implementació en javascript[7]. Sembla força interessant i hauria de ser estudiat.
- Dintre del món del cloud, hi ha un nou model de criptografia anomenat xifrat homomòrfic[8], que permet realitzar operacions amb dades xifrades, de manera que el

[4] http://www.propublica.org/article/the-nsas-secret-campaign-to-crack-undermine-internet-encryption
[5] http://www.w3.org/TR/WebCryptoAPI/
[6] https://bugzilla.mozilla.org/show_bug.cgi?id=865789
[7] http://badassjs.com/post/40101764862/polycrypt-a-webcrypto-api-polyfill-in-javascript
[8] http://en.wikipedia.org/wiki/Homomorphic_encryption

resultat xifrat és equivalent a l' operació feta en clar. Actualment es considera un dels futurs pilars de la privacitat a Internet, i s'està en una etapa d' expansió.[9]

- Respecte a la gestió de claus, hi ha alguns sistemes que proposen fer servir models PGP com PGfb[10], si bé altres adopten la postura de posar les claus de xifrat en un sistema diferent, ala DLP (Data Loss Prevention) com ara bé ho fa Prot-On[11], però en general, aposten per una gestió pròpia de les claus[12] [13] [14] [15]
- La web de MEGA aporta una visió molt interessant de com gestionar el xifrat de les dades fent servir storage HTML5 y oferint a nivell d'usuari un filesystem segur amb crides POSIX.[16]

Per altre banda, també s'han mirat mecanismes de xifrat per les xares socials:

- NOYB[17] es un sistema per a Facebook on es barregen les dades personals dels usuaris entre ells per a dificultar la identificació d'aquests.
- FaceCloak[18] es un sistema per a Facebook que fa servir una clau simètrica per a xifrar la informació i aquesta es guarda a un servidor central, posant un text extret de wikipedia com a índex de referència a aquesta informació xifrada. Aquest sistema té el problema que el text extret de wikipedia es arbitrari i pot portar a confusió al resta d'usuaris.
- Diaspora[19] es un sistema que permet compartir la informació fent servir el esquema OpenPGP però amb seva pròpia xarxa d'emmagatzematge de claus.
- flyByNight[20] es una aplicació per a Facebook que permet xifrar les dades, però que la desa les claus en el propi servidor de Facebook.
- Scramble![21] es una extensió per a Firefox que utilitzant el model OpenPGP permet tindre grups destí de la informació xifrada mitjançant l'ús d'una clau simètrica per grup, on

[9] http://phys.org/news/2013-06-cloud-algorithm-major-problem-homomorphic.html
[10] https://github.com/m0hit/PGfb
[11] http://www.prot-on.com/
[12] http://www.spacenext.com/encrypt-facebook.php
[13] https://chrome.google.com/webstore/detail/safe-secure-and-private-f/nmeknaoicphndgilljmajkhkoheceico
[14] https://chrome.google.com/webstore/detail/secure-gmail-by-streak/jngdnjdobadbdemillgljnnbpomnfokn
[15] http://www.securitybydefault.com/2008/09/como-sincronizar-un-comando-terrorista.html
[16] https://mega.co.nz/#doc
[17] Saikat Guha, Kevin Tang, and Paul Francis. Noyb: privacy in online social net- works. In WOSN '08: Proceedings of the first workshop on Online social networks, pages 49–54, New York, NY, USA, 2008. ACM
[18] Matthew M. Lucas and Nikita Borisov. Flybynight: mitigating the privacy risks of social networking. In Proceedings of the 7th ACM workshop on Privacy in the electronic society (WPES), pages 1–8, New York, NY, USA, 2008. ACM
[19] https://joindiaspora.com/
[20] Matthew M. Lucas and Nikita Borisov. Flybynight: mitigating the privacy risks of social networking. In Proceedings of the 7th ACM workshop on Privacy in the electronic society (WPES), pages 1–8, New York, NY, USA, 2008. ACM

aquesta es compartida als membres del grup mitjançant un sobre digital individual. A part, permet signar els missatges i aquells que siguin massa llargs, enlloc de posar el missatge, posar-hi una URL amb un enllaç al missatge xifrat.

[21] http://www.freehaven.net/anonbib/papers/pets2011/p12-beato.pdf

Objectius inicials

Es van establir els següent objectius per el TFC

1. Entendre els mecanismes actuals i futurs de seguretat que implementa un navegador
2. Conèixer l'estat de l'art de la criptografia en javascript i en HTML5.
3. Oferir una visió de com podrien resoldre's certs aspectes de la privacitat des de la perspectiva del navegador com a sistema operatiu.
4. Implementar un petit prototip que permeti comprovar el seu funcionament

Així doncs, es van plantejar inicialment tres mecanismes a implementar a nivell de navegador:

- Fer que el navegador ofereixi un mecanisme de obtenció protegida d'informació per part de l'usuari. Això vol dir que el proveïdor de serveis, mitjançant javascript podrà obtenir informació xifrada o en el cas que sigui desxifrada amb un avís. Això inclou la capacitat de que en la etapa de rendering del text, si aquest conté informació xifrada, es visualitzi en clar.
- Fer que el navegador ofereixi una CryptoAPI mínima perquè el proveïdor d'informació pugui gestionar y utilitzar (sense conèixer en sí) les claus de xifrat que fa servir l'usuari, així com habilitar mecanismes per a la distribució de les claus (possiblement PGP)
- Si hi ha temps, afegir la possibilitat d'habilitat càrrega de JavaScript signats en el navegador

Es va plantejar dues possibilitats per fer el desenvolupament: Chrome i FirefoxOS.

Metodologia de desenvolupament

La metodologia de desenvolupament d'aquest projecte ha sigut la pròpia d'un prototip o de prova de concepte, per tant:

- No s'han realitzat testos unitaris ni d'integració
- No s'ha documentat el codi
- No hi ha control de versions centralitzat, només versió local amb git
- No hi ha incidències ni tracking d'aquestes, ni es prioritzen tasques
- No s'està obligat a seguir les convencions de codificació de plataforma (si bé s'ha intentat)
- S'intentarà tindre solucions visibles de cara a cada PAC

Planificació

Es van establir els següents passos per a la realització del TFC

- Examinar internament en quin estat son els projectes Chrome i FirefoxOS i seleccionar-ne un.
- Estudiar l'arquitectura del navegador
- Crear l'entorn de desenvolupament per poder compilar/desenvolupar plugins
- Estudiar com dotar al DOM per afegir un mecanisme d' introducció i visualització de dades segura
- Implementar el mecanisme d'introducció i visualització de dades segura
- Estudiar una CryptoAPI mínima que permeti la gestió i la distribució de les claus
- Implementar la CryptoAPI
- Construir una demo
- Generar documentació sobre la feina realitzada

I es va elaborar la següent planificació temporal tenint en compte que les jornades són de 4 hores.

Resultat

El resultat d'aquest TFC ha sigut ha aconseguit de manera satisfactòria:

- Un model criptogràfic de privacitat en xarxes socials amb la novetat que es possible compartir un secret amb els amics dels amics.
- Un plugin funcional per a Google Chome i un servidor en nodeJS com a suport del plugin per a la persistència compartida, que implementen el model criptogràfic descrit.
- Coneixement de la base de dades NoSQL i del servidor nodeJS
- Coneixement de desenvolupament de plugins en Google Chrome
- Coneixement de l'estat de l'ús de la criptografia des de navegadors web.

- Aquesta documentació

Si bé, finalment, no s'ha pogut implementar ni el mecanisme d'obtenció i visualització d'informació segura ja que l'única manera era modificar i recompilar el navegador web i per altre banda no ha sigut necessària la introducció d'escripts signats ja que el model de seguretat per a plug-ins de Goolge Chrome es prou bo com perquè no calgués una protecció extraordinària d'escripts.

Disseny

Una mica basat en la idea de les identitats tenen un fort component social, arribant al punt que algunes xarxes socials com facebook permet recuperar la contrasenya a través dels teus propis amics [22], s'ha dissenyat un protocol que permet enviar missatges a la xarxa i que arribin a gent que podrien ser amics teus, es a dir, que comparteixen un nombre mínim d'amics en comú. Si bé aquest fet no té perquè ser sempre útil, no deixa de ser una bona excusa per intentar fer una implementació una mica diferent de les habituals.

Criptografia

La base d'aquest model consisteix en que el usuari que deixa xifrat un missatge a la xarxa, ho fa de manera que els amics dels seus amics puguin llegir els seus missatges, tal i com s'ha comentat. La manera de fer-ho es fent ús de la criptografia asimètrica RSA com de l'algorisme de Shamir.

Base del sistema

Quan un usuari envia missatge xifrat, la clau de xifrat que ha utilitzat la parteix en fragments reconstruïbles mitjançant Shamir, en tants fragments com amics tingui. Cada amic seu ha d'emetre un certificat (Key certificate) i compartir amb ell la clau privada, i ell xifrarà cada fragment amb la clau publica d'un dels seus amics. Així els Key certificates amb les seves claus privades son compartits entre tots els amics en comú. Per desxifrar el missatge només cal que algú que comparteixi tants amics en comú com el paràmetre de reconstrucció de Shamir que s'ha utilitzat.

Vegem un exemple:

Una xarxa social conté els usuaris A, B, C, D, Z
{A,B,C,D} son amics entre ells
{B,C,D,Z} son amics entre ells

Com hem dit cada usuari genera el seu certificat amb clau privada (Key certificate), que anomenarem com K_{id} i el compartirà amb els seus amics, per tant:

A tindrà K_a,K_b,K_c,K_d

[22] Facebook's Trusted Contacts Sends Password Reset Codes to Your Friends
http://lifehacker.com/facebooks-trusted-contacts-sends-password-reset-codes-487340748

B tindrà K_b,K_a,K_c,K_d,K_z
C tindrà K_c,K_a,K_b,K_d,K_z
D tindrà K_d,K_a,K_b,K_c,K_z
Z tindrà K_z,K_b,K_c,K_d

Ara A genera una clau simètrica cs per xifrar els seus missatges, escull que almenys son necessaris que es comparteixin dos dels seus amics per a llegir el missatge, per tant, amb l'ajuda de l'algorisme partirà la clau en tants amics com tingui, en aquest cas 3,

Shamir(cs,3,2)= {cs_1,cs_2,cs_3}

fent que sigui reconstruïble amb dos, es a dir, que amb {cs_1,cs_2},{cs_1,cs_3} o {cs_2,cs_3} es pot reconstruir el secret ca.

Ara xifra amb la clau privada dels certificats dels seus amics, cada fragment. En fa un paquet i ho comparteix de manera publica:

{ RSA clau publica K_b de cs_1
RSA clau publica K_c de cs_2
RSA clau publica K_d de cs_3 }

Ara Z, que NO es amic de A, com té les claus publiques privades de K_b i de K_c (perquè n'és amic) podrà obtenir cs_1 i cs_2 i reconstruir el secret cs que xifrava el missatge enviat per A.

Així veiem que aquest sistema requereix, addicionalment d'un repositori comú, que pot ser centralitzat o bé distribuït (com ara ho és el blockchain de bittorrent) i que es necessita confiança explicita en els Key certificates ja que al ser autosignats no venen avalats per cap autoritat central (cosa que es podria solucionar fent que els key certificates tinguessin una signatura provinent d'un DNI electrònic o d'una clau PGP, per exemple).

Millora del sistema

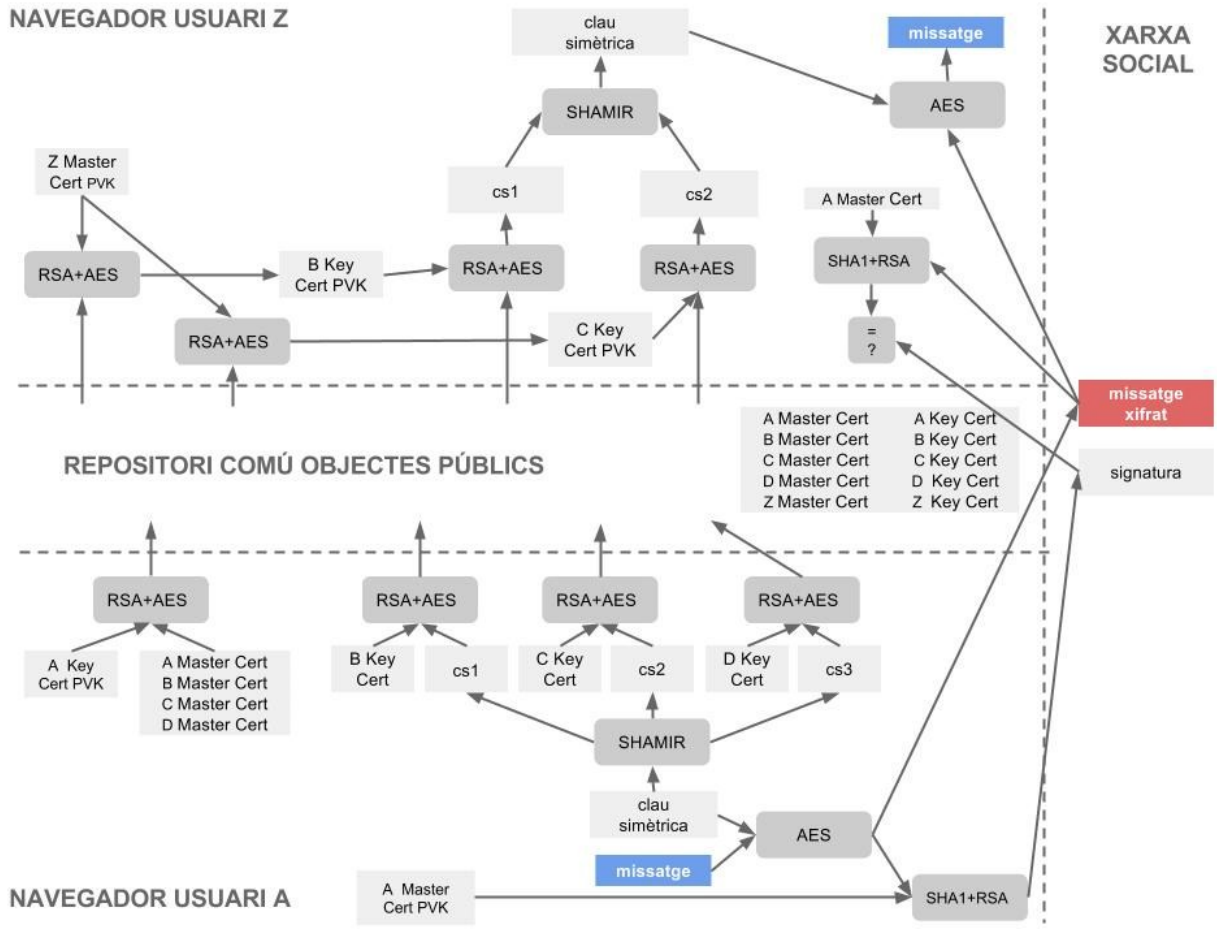
Tal com s'ha descrit el sistema té el problema que si es deixa de ser amic d'algú aquest amic te la clau privada del certificat, per tant deixa de ser útil. Per solucionar-ho es crea un certificat jeràrquicament superior anomenat Master certificate (en el mòn de la PKI podríem dir-li un certificat de CA) que signarà el Key certificate, de manera que aquest deixa de ser autosignat, passant-ho a ser el Master certificate.

Això té avantatges:

- Si es canvia d'amics, es genera un nou Key certificate, i es xifra contra els Master certificates dels amics en un repositori comú, de manera que no cal tornar a confiar explícitament en el Key certificate, només caldrà fer-ho en el Master certificate.
- Permet afegir una signatura electrònica als missatges enviats, ja que es posseeix una clau privada que no es comparteix amb ningú.

Per altre banda, els xifrats han de ser amb sobre digital no fent servir un xifrat RSA pur, per prevenir atacs.

Així, podríem veure com quedaria l'exemple anterior amb la millora proposada



Especificació del protocol

El disseny que s'ha seleccionat per a realitzar la pràctica es pot descriure de la següent manera:

Sigui

- A(id): el conjunt d'amics de id en aquesta xarxa
- M_{id} : un certificat autosignat emès per id
- K_{id} : un certificat emès per M_{id}
- C[pvk]: clau privada del certificat C
- C[pbk]: clau publica del certificat C
- AES(m,cs) : el xifrat amb l'algorisme AES[23] del missatge m amb la clau asimètrica cs
- RAND(b): un nombre aleatori de b bits
- RSA_C(m,ca) : xifrat/desxifrat amb sobre digital[24] utilitzant AES/RSA[25] del missatge m amb la clau asimètrica ca
- RSA_S(m,ca) : signatura SHA1/RSA del missatge m amb la clau asimètrica ca
- SHAMIRC(S,n,nr) = { $p_1,p_2,..,p_n$} crea una partició amb l'esquema de Shamir[26] del secret S en n parts: $p_1,p_2,..,p_n$ parts, reconstruible amb nr parts.
- SHAMIRR({ $p_1,p_2,..,p_k$}) =S recostrueix el secret S a partir de, almenys, k parts
- #O: es el cardinal del conjunt O

Inicialització única del usuari u

1. Cada usuari u crea el seu propi certificat M_u amb la corresponent clau privada

Inicialització o en cas que canviï A(u)

2. A(u)∪{u} s'intercanvien els certificats M entre ells
3. Per cada usuari u
   - 3.1. Crea el certificat K_u que està emès per el seu propi M_u
   - 3.2. Publica a un repositori comú el seu certificat públic K_u
   - 3.3. Crea una clau de xifrat aleatòria $xifrat_u$ = RAND(bits)
   - 3.4. Divideix la clau aleatòria en #A(u) parts P_u= {$p_1,p_2,..,p_{\#A(u)}$} = SHAMIRC($xifrat_u$,k,#A(u))

[23] http://es.wikipedia.org/wiki/Advanced_Encryption_Standard
[24] http://www.revista.unam.mx/vol.7/num7/art55/art55-3.htm
[25] http://es.wikipedia.org/wiki/Criptograf%C3%ADa_asim%C3%A9trica
[26] http://es.wikipedia.org/wiki/Esquema_de_Shamir

3.5. Per cada amic a ∈ A(u)

3.5.1. Publica a un repositori comú la clau privada del seu certificat K_u, xifrat amb la clau publica del seu amic $RSA_C(K_u[pvk],M_a[pbk])$

3.5.2. Recupera del repositori comú el certificat i la clau privada del seu amic xifrada $RSA_C(K_a[pvk],M_u[pbk])$ i la desxifra

3.5.3. Publica al repositori la clau un element de p_u xifrat (que no s'hagi agafat abans) amb la clau publica del certificat K del amic: $RSA_C(p_a ,K_a[pbk])$

Publicació d'un missatge xifrat

1. L'usuari u publica $missatgexifrat_u = AES(missatge_u,xifrat_u)$
2. Opcionalment també pot posar la signatura al missatge $\{missatgexifrat_u,RSA_S(missatge_u,M_u[pvk])\}$

Desxifrat del missatge

1. L' usuari w recupera el missatge $missatgexifrat_u$
2. Recupera els $\{RSAC(p,K_a[pbk])\}$ que ha publicat el usuari u
3. Dels $K_a[pbk]$ que té el $K_a[pvk]$,és a dir, que també son amics seus, desxifra la part p
4. Un cop té almenys k elements dels shares, pot reconstruir el secret $SHAMIRR(p_1,p_2,..,p_k) = xifrat_u$

Persistència

Persistència local

Els usuaris u necessiten emmagatzemar localment:

- certificat M_u i M_u[pvk]
- certificat K_u i K_u[pvk]
- per cada a ∈ A(u) M_a
- $xifrat_u$ actual

Per tant, en tindrem prou en emmagatzemar parells clau-valor.

Persistència compartida

Necessitarem emmagatzemar la següent informació:

- certificats M
- certificats K
- els $RSA_C(K_u[pvk],M_a[pbk])$
- els $RSA_C(p_a ,K_a[pbk])$

Implementació

Elecció de tecnologia

La primera decisió que calia fer era decidir si la implementació es feia sobre Chrome o sobre FirefoxOS. Finalment es va escollir Google Chrome perquè es una plataforma que porta més temps, hi ha més documentació, i hi ha més exemples i tutorials, i, segurament mes estable, que no FirefoxOS, si bé no ho vaig poder comprovar directament.

Ja que aquest TFC està totalment orientat a tecnologies web, tenia sentit que a la part servidora també utilitzes el mateix model, així que s'ha escollit per a la tecnologia servidor la plataforma nodejs (servidor de javascript que fa servir el mateix core de javascript que chrome, el V8) i una base de dades NoSQL que permet emmagatzemar objectes en javascript: mongoDB.

Això fa que puguem treballar amb dades en JSON des de el navegador fins a la persistència en el servidor sense que calgui cap conversió addicional.

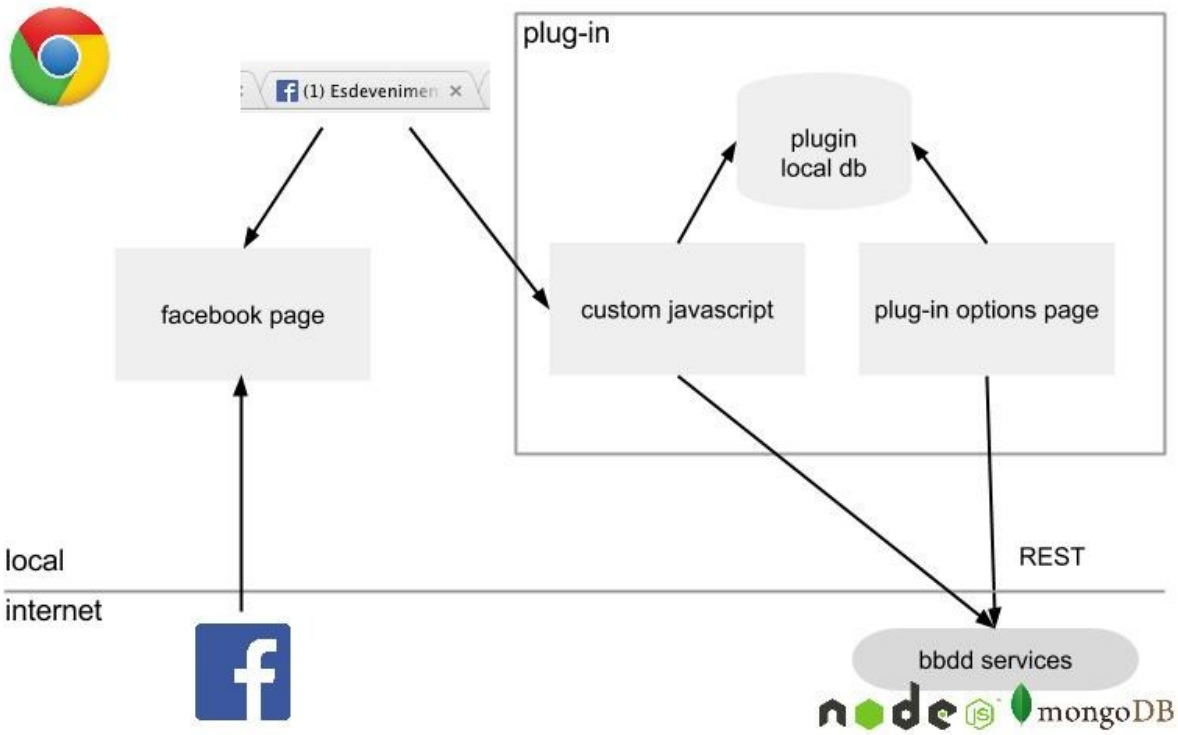
Així, l'anomenat *stack* de tecnologies que utilitza el projecte és:

- HTML5 / Javascript
- Google Chrome Plugin API
- MongoDB
- nodeJS
- jQuery
- BootStrap 3.0

Respecte a la fortalesa criptogràfica, s'han utilitzat els següents, si bé, un desplegament real se n'haurien d'incrementar els valors per fer-los més segurs

- Claus RSA de 512
- Claus de xifrat IDEA de 32 bits d'entropia
- Funció resum SHA-1

Vista general



A la implementació podrem trobar els següents components:

1. **plugin local db**: Les dades locals es desen dintre d'una petita base de dades que permet tindre chrome dintre dels plugins.
2. **plugin options page**: Tenim una pàgina d'opcions del plug-in per a configurar-lo que podem accedir desde la pàgina d'extensions de Chrome.
3. **bbdd services**: son una sèrie de serveis REST muntats sobre un servidor nodejs i una base de dades NoSQL mongoDB
4. **custom javascript** i **facebook page**: quan facebook entrega una pàgina a Chrome, aquest detectarà que s'ha d'activar el plugin, i just després de la càrrega de la pàgina, executarà el nostre javascript (custom javascript) que permetrà manipular-la.

Part client

Per la part del client s'ha decidit fer un plug-in per a Google Chrome, ja que l'alternativa de Firefox OS no s'ha trobat que fos massa estable com per a fer desenvolupaments "complicats".

El codi està fet amb Javascript i s'ha utilitzat

- HTML5 per la persistència local
- llibreria jquery per la interacció amb l'usuari i la base de dades
- llibreria CryptoJS v3.1.2 per l'AES
- llibreries jsbn per l'RSA
- llibreries de Shamir d'Alexander Stetsyuk

Interacció amb l'usuari

Per xifrar un contingut, cal seleccionar el test a xifrar i prémer control-1, per desxifrar un contingut cal seleccionar el contingut xifrat i prémer control-1.

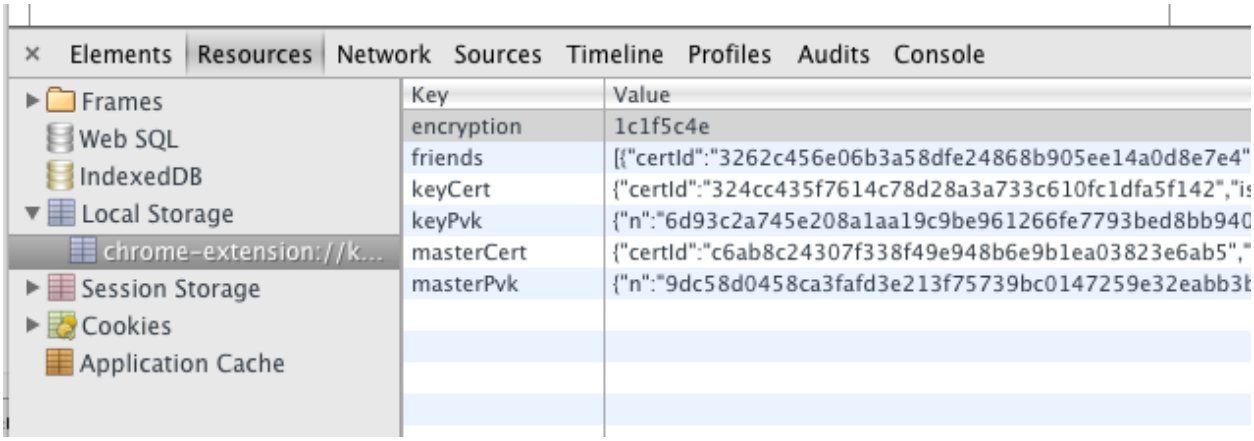
S'han fet una quantitat considerable d'esforç en fer que la interacció amb l'usuari sigui el més fàcil possible, intentant que el xifrat i el desxifrat es fes de manera automàtica mitjançant la manipulació dels callbacks del DOM amb jquery però no s'ha aconseguit cap resultat satisfactori.

Persistència local

Per la persistència local, chrome permet desar les dades mitjançant la API HTML5. És molt senzilla, ja que només cal accedir a la variable localStorage, i escriure-hi valors:

```
localStorage[key] = value;
value = localStorage[key];
```

Aquestes dades després es poden veure fàcilment desde la consola de desenvolupadors de Chrome:



API

S'han implementat les següents funcions per a poder implementar la API:

Extensions de jsbn (`tfccrypto.js`)

- mètode `isPrivate` : retorna true si conté la clau privada
- mètode `serialize`: serialitza a un objecte pla
- mètode `unserialize` : desserialitza desde un objecte pla
- mètode `signSHA1` : fa una signatura amb SHA1
- mètode `verifySHA1` : verifica una signatura feta amb signSHA1
- mètode `envelope` : xifra amb AES
- mètode `unenvelope` : desxifra amb AES

Classe `Certificate` per gestionar certificats (`tfccrypto.js`)

- mètode `getToBeSigned` : retorna les dades que son signades
- mètode `generateSelfSigned` : genera un certificat autosignat
- mètode `issueCertificate :` crea i signa un certificat
- mètode `verify :` verifica la signatura d'un certificat
- mètode `serialize:` serialitza a un objecte pla
- mètode `unserialize :` desserialitza desde un objecte pla
- mètode `encrypt :` xifra RSA
- mètode `decrypt :` desxifra RSA
- mètode `id :` torna l'identificador del certificat
- mètode `issuerId :` emissor del certificat
- propietat `idProof :` prova de validesa del certificat

- propietat `issuerKey :` clau publica del emissor del certificat
- propietat `subject :` propietari del certificat
- propietat `signature :` signatura
- propietat `key :` clau RSA (pot ser publica o privada)

Classe SCryptoV2 per gestionar criptografia (`tfccrypto.js`)

- mètode `setMinShare :` estableix el nombre d'amics necesari per recompondre un share
- mètode `setMyKeyCertificate :` estableix la clau K_u (amb clau privada)
- mètode `addFriendMasterPbkCertificate :` afegir M_a (sense clau privada) d'un amic
- mètode `addFriendKeyPvkCertificate :` afegir K_a (amb clau privada) d'un amic
- mètode `createNewSecretAndGetSharesVectorInfo :` crea un nou $xifrat_u$ i torna P_u
- mètode `cipherMessage :` xifra un missatge
- mètode `decipherMessage :` desxifra un missatge amb l'ajuda d'un P_a
- mètode `selftest :` fa un self-test
- propietat `cert :` K_u (amb clau privada)
- propietat `fiendsMasterCertificates` : els M_a
- propietat `fiendsKeyCertificates` : els K_a
- propietat `minShare` : k (nombre d'amics necessari per recompondre un share)
- propietat `currentSecret` : $xifrat_u$

L'API pública en si és la següent (`tfccontroller.js`)
(no s'ha fet una classe perquè hi havia problemes amb crides múltiples ajax en la llibreria jquery):

- funció `ctl_doinit` : inicialitza la API
- funció `ctl_newMasterCertificate` : crea un nou M_u i el publica al repositori
- funció `ctl_newKeyCertificate` : crea un nou K_u i el publica al repositori la pvk xifrada
- funció `ctl_newEncryptionKey` : crea un nou K_u i el publica al repositori P_u
- funció `ctl_addFriendMasterCertificate` : cerca un M_a al repositori i l'afegeix a la llista d'M_a
- funció `ctl_removeAllFriends` : esborra la llista d'M_a
- funció `ctl_encodeMessage` : xifra un missatge
- funció `ctl_decodeMessage` : desxifra un missatge

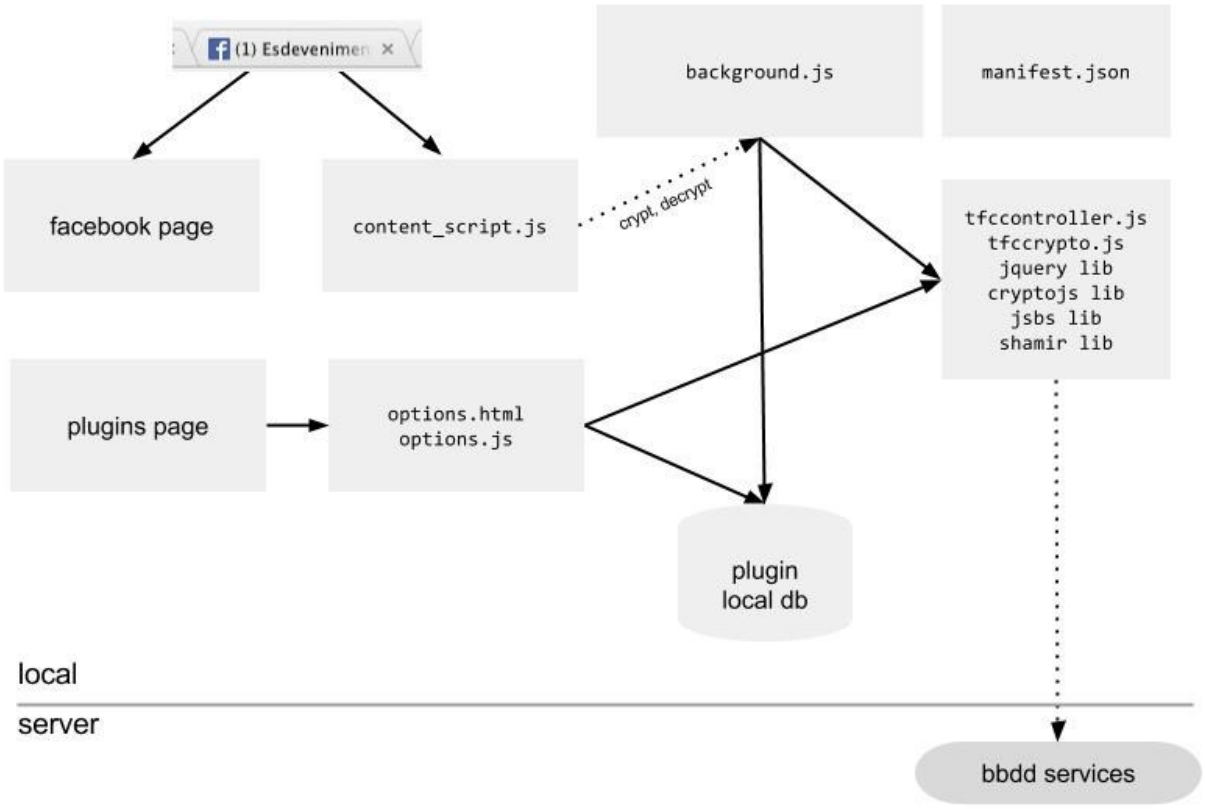
perquè aquesta api funcioni necessita que estiguin definides les següents funcions:

- funció `ctl_storageget` : recuperar un {clau,valor} del emmagatzemant local
- funció `ctl_storageset` : establir un {clau,valor} del emmagatzemant local
- funció `ctl_storageexist` : mirar si existeix una clau,valor en el emmagatzemant local
- funció `ctl_storageremove` : esborrar un {clau,valor} del emmagatzemant local

- funció `ctl_log` : gestionar una traça

Plug-in de Google Chrome

Arquitectura del plugin



1. **content_script.js**: És l'script que s'executa just després de carregar la pàgina de facebook. Aquest script modifica, mitjançant jquery, els callbacks dels elements del DOM de facebook (keypress) per a que es pugui treballar amb ells. Aquesta pàgina no pot accedir a les dades locals (M_u,K_u,$xifrat_u$...), per això es fa una crida asíncrona per xifrar i desxifrar a un servei publicat a **background.js**, ja que aquest script si que té permisos per accedir a aquestes dades (anomenada *backgroud page*).
2. **plug-in options page**: Tenim una pàgina d'opcions del plug-in per a configurar-lo que podem accedir desde la pàgina d'extensions de Chrome.

3. **bbdd services**: són una sèrie de serveis REST muntats sobre un servidor nodejs i una base de dades NoSQL mongoDB
4. **custom javascript** i **facebook page**: quan facebook entrega una pàgina a Chrome, aquest detectarà que s'ha d'activar el plugin, i just després de la càrrega de la pàgina, executarà el nostre javascript (custom javascript) que permetrà manipular-la.

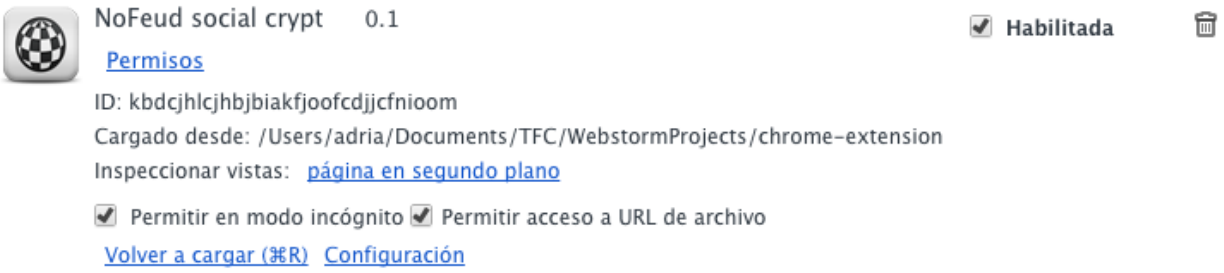
Arxiu `manifest.json`

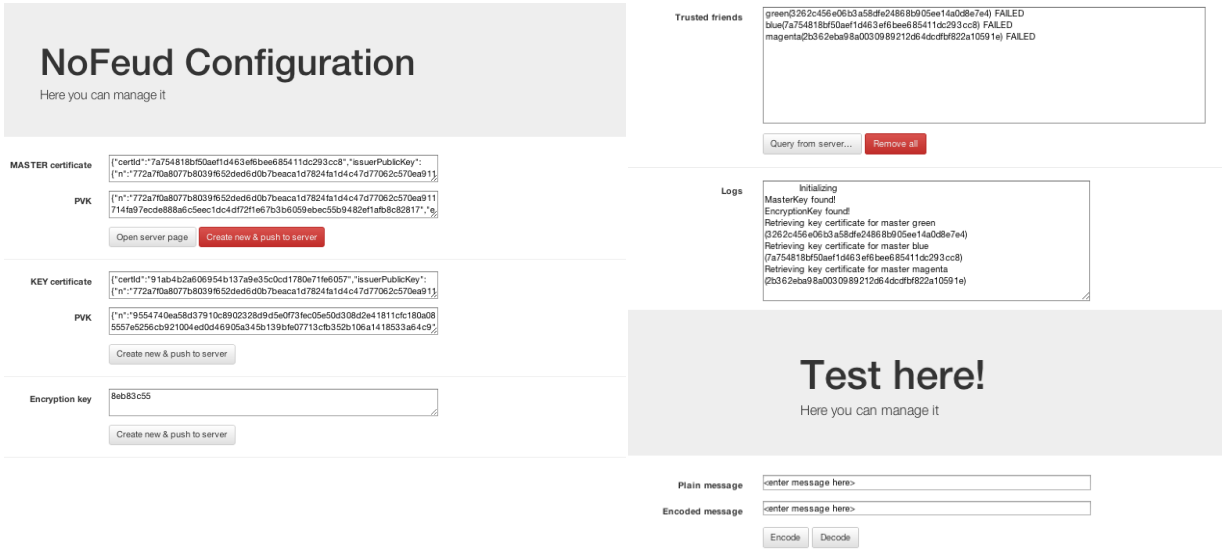
Aquest arxiu és necessari per definir les característiques generals del plug-in i estableix que:

- Es té accés a les pàgines "https://twitter.com/*", "https://www.facebook.com/*"
- Es pot fer servir emmagatzematge local
- Es poden executar scripts dintre les pàgines
- Hi ha una pàgina d'opcions
- Els scripts que cal executar quan s'accedeix a les pàgines "https://twitter.com/*", "https://www.facebook.com/*"
- Hi ha una pàgina d'opcions

Pàgina d'opcions

Els arxius options.html, options.js: correspon a la pàgina "Configuración", que es pot accedir mitjançant la pàgina d'extensions:





En aquesta pàgina podem trobar el següent:

- *MASTER certificate*: M_u i M_u[pvk]
  - El botó *Create new & push* pregunta per la meva identitat (subject), en crea el certificat M_u i la publica al servidor

- *KEY certificate*: K_u i K_u[pvk]
  - El botó *Create new & push* crea un nou K_u i el publica al repositori la pvk xifrada

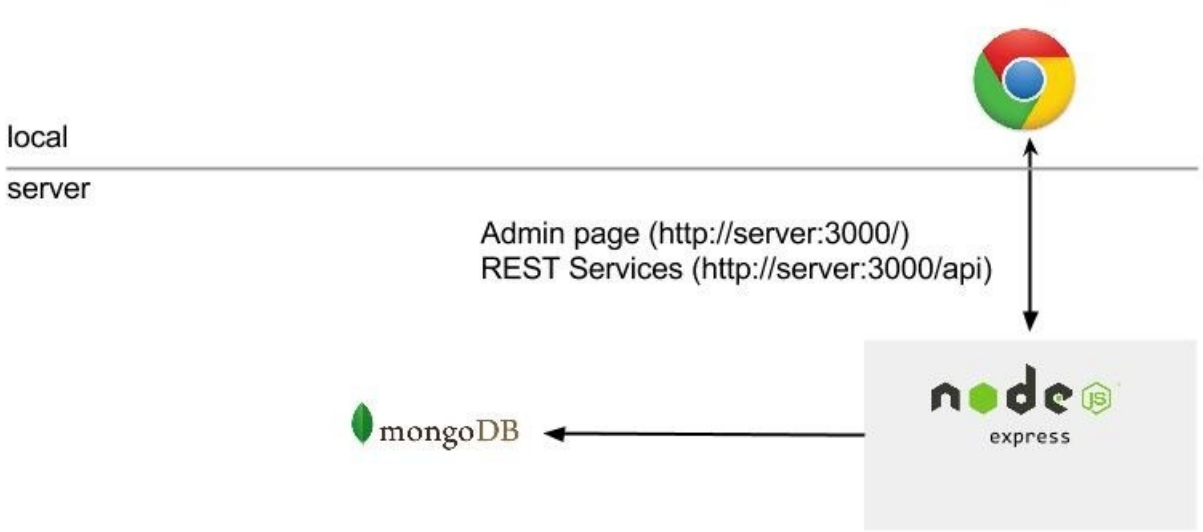
- *Encryption key* : la clau de xifrat actual.
  - El botó *Create new & push* crea un nou K_u i el publica al repositori P_u

- *Trusted friends*: la llista dels M_a
  - El botó *Query from server*, pregunta el subject del certificat M_a que es vol incorporar, el busca al repositori i l'afegeix a la llista.
  - El botó Remove all, esborra la llista d'amics

- Àrea de logs

- *Test:* serveix per fer proves de xifrat i desxifrat. En el cas de desxifrar preguntarà la identitat del publicador del missatge.

Part servidora



Per la part del servidor hem utilitzat un servidor NodsJS amb una base de dades NoSql mongoDB. El servei ofereix tant una pàgina de configuració del servei (ara és de consulta) i la capa de serveis REST.

Dir, per això, que aquest hauria de ser el servei que el proveïdor de xarxa social hauria de proporcionar no és correcte, ja que obriria la porta a molts forats de seguretat en el cas que el proveïdor volgués fer un atac dirigit. En el capítol final, proposem un model de persistència compartida, que seria més aplicable que no pas la que s'ha implementat, si bé és molt més complexa i escapa a càrrega del TFC.

Model de dades

El model de dades es força senzill, existeixen 3 col·leccions:

- master: per desar els MASTER certificates M_u
- key: per desar els KEY certificates. $RSA_C(K_a[pvk],M_u[pbk])$
- share: per desar els secrets compartits P_u

Un exemple de les dades emmagatzemades són les següents:

master (certificat amb subject "blue")


```
{
  "id": "blue",
  "cert": {
    "certId": "7a754818bf50aef1d463ef6bee685411dc293cc8",
    "issuerPublicKey": {
      "n":
"772a7f0a8077b8039f652ded6d0b7beaca1d7824fa1d4c47d77062c570ea911714fa97ecde888a6c5eec1dc4df72f1e67b3b6059e
bec55b9482ef1afb8c82817",
      "e": "10001"
    },
    "subject": "blue",
    "publicKey": {
      "n":
"772a7f0a8077b8039f652ded6d0b7beaca1d7824fa1d4c47d77062c570ea911714fa97ecde888a6c5eec1dc4df72f1e67b3b6059e
bec55b9482ef1afb8c82817",
      "e": "10001"
    },
    "idProof": "",
    "signature":
"2fa39479b089fc2f6009ef026e3a60bb0ac1638bf79b4aaf8ceaad3ca791c00b1fbaf7c0b1faa9ac17904cf76807f62019b1c0389
837489d22ef7680a0a7e1af"
  }
}
```


key (que ha emès el certificat "blue" contra un altre master amb id c6ab8c24307f338f49e948b6e9b1ea03823e6ab5)


```
{
  "id": "7a754818bf50aef1d463ef6bee685411dc293cc8.c6ab8c24307f338f49e948b6e9b1ea03823e6ab5",
  "data": {
    "pvkEnveloped": {
      "cipheredKey": "31536f6fbab4d7efc63ee76b332cb2c6daccefd6182732ede4a6d71584c8592b03b6ec33c61ac...",
      "cipheredData": "U2FsdGVkX19TeoUi7SNMp5gJ8+rNK1xz/IdkP0uKjX42UurYGx82Wg3EhxOwPV2MceFWMBIPdqHs..."
    },
    "certificate": {
      "certId": "91ab4b2a606954b137a9e35c0cd1780e71fe6057",
      "issuerPublicKey": {
        "n": "772a7f0a8077b8039f652ded6d0b7beaca1d7824fa1d4c47d77062c570ea911714fa97ecde888a6c5eec...",
        "e": "10001"
      },
      "subject": "12/10/2013",
      "publicKey": {
        "n": "9554740ea58d37910c8902328d9d5e0f73fec05e50d308d2e41811cfc180a085557e5256cb921004ed0d4...",
        "e": "10001"
      },
      "idProof": "parent-signed",
      "signature": "2fa39479b089fc2f6009ef026e3a60bb0ac1638bf79b4aaf8ceaad3ca791c00b1fbaf7c0b1faa9ac..."
```

```
    }
  }
}
```

share (que ha emès el certificat "blue")


```
{
    "id": "blue",
    "data": {
        "publisherpbk": "91ab4b2a606954b137a9e35c0cd1780e71fe6057",
        "shares": [
            {
                "id": "ccb69a20aa480d329c3a2788939610c95ee338ef",
                "cipheredKey": "829ff03cf816bcebb3b834b9f759ae226fb5f9a9e5edeb2318af0c4d429f529daca...",
                "cipheredShare": "U2FsdGVkX1+4KnH13B538RyO941DqRAAI6hBPZJNyRQ="
            },
            {
                "id": "c7f8688b541843557763bbc54c3b9c015a224976",
                "cipheredKey": "18e75966e9af43a9999db14688d8aa616e58c209d8a541a99f38dfe5b2256004...",
                "cipheredShare": "U2FsdGVkX19EvuY89xKOKrtL3y3n2r7IzAcRf/gyPHI="
            },
            {
                "id": "769a6660e06161b936693cd8c581b44fc8048dff",
                "cipheredKey": "5afff351f4ccc1ff50fccc421f1554584d6dff8e06999dc2c936a129e379cb18...",
                "cipheredShare": "U2FsdGVkX1/u/DEawNmvwmkG+aiK7LNdH3a8IPx3JlI="
            },
            {
                "id": "0d408043fe07e6dfbb4928ccfe8febf2b157d1c5",
                "cipheredKey": "31ef64447072ebab3b216b2ba6e63265a65ae5444bfb81cc70b09dd21815a8b0...",
                "cipheredShare": "U2FsdGVkX1/tvBJEfqcFI9edkfVY7F4sU4VxE4ahQHU="
            },
            {
                "id": "91ab4b2a606954b137a9e35c0cd1780e71fe6057",
                "cipheredKey": "10b4d9fbb03fcef7a5b8e66e9237f19bf8d862160d050d3d8673e49b41ca8e39...",
                "cipheredShare": "U2FsdGVkX19WvF3drwZ+2Ws9Bz2ikUET+G1GoJ1d/8s="
            }
        ]
    }
}
```

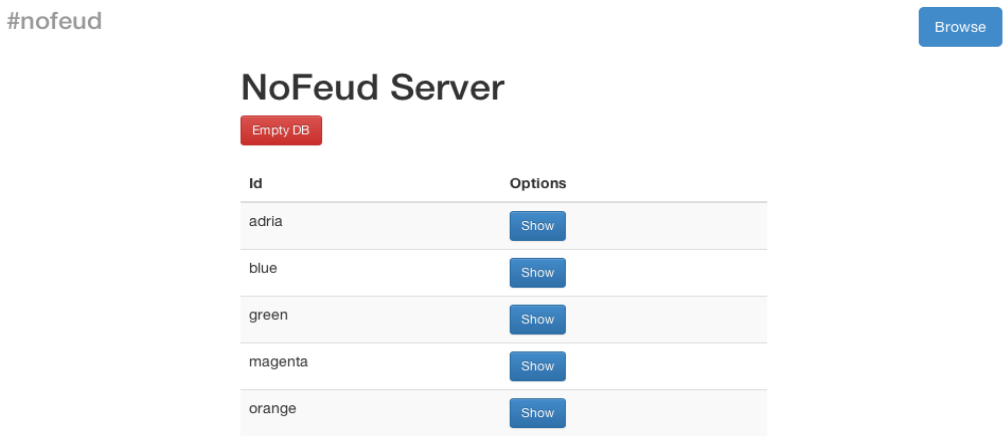
API de servei REST

El servei REST té la següent API:

Path	VERB	
`/api/master`	GET	Recupera tots els certificats M_u
`/api/master`	DELETE	Esborra tota la base de dades
`/api/master`	POST	Afegeix un certificat M_u
`/api/master/:subject`	GET	Recupera un certificat master a partir del seu subject
`/api/master/:id/share`	GET	Recupera els P_u a partir del id d'un certificat
`/api/master/:id/share`	POST	Estableix el P_u a partir del id d'un certificat
`/api/master/:idfrom/:idto`	GET	Estableix el K_u i la clau privada xifrada contra la clau publica de :idto de :idfrom
`/api/master/:idfrom/:idto`	POST	Recupera el K_u i la clau privada xifrada contra la clau publica de :idto de :idfrom

Pàgina de configuració

Finalment no s'ha fet una pàgina de configuració, sinó que només s'ha realitzat una senzilla pàgina de visualització de quants master certificates hi ha a la base de dades que permet esborrar-la (per a fer proves)



Conclusions

Aquest TFC ha sigut al excusa perfecte per a poder investigar i indagar sobre les capacitats dels navegadors web (i especialment Google Chrome) dintre del camp de la privacitat, i treure'n les següents conclusions:

- Google Chrome no està actualment prou preparat per a poder oferir mecanismes d'introducció i visualització segura de dades, en el sentit que es puguin manipular la pàgina web per a mostrar dades no accessibles des del javascript que està sota control del proveïdor web. I sembla que els altres navegadors tampoc.
- El navegador web encara no està madur per oferir la possibilitat d'oferir mecanismes de intercanvi d'informació entre iguals per a gestionar la seguretat de manera descentralitzada ; existeixen alguns mecanismes com WebRTC però encara són en beta i no estan enfocats en aquesta direcció.
- Si bé javascript es el llenguatge per excel·lència, en sí mateix es massa poc estructurat i tipat per desenvolupar eines de seguretat, cal estructurar-lo amb cura i amb eines addicionals.
- Actualment no existeix cap mecanisme consolidat per accedir a la criptografia del sistema operatiu des del navegador però almenys comencen a existir llibreries per el desenvolupament de la seguretat amb usant estàndards com X.509 i PGP.

Següents passos

- Els certificats que s'estan fent servir es un model massa "ad-hoc" i caldria canviar-los, per exemple per certificats X.509 o PGP, dels quals ja hi ha alguna implementació disponible amb javascript [27] [28]
- Actualment no hi ha verificació dels certificats Master, així que seria bo que anessin signats, per exemple per una clau PGP o amb el DNI electrònic. El camp idProof dels certificats està disponible per aquest ús però no s'ha implementat en aquest TFC.
- Actualment els plug-ins no tenen la potència suficient com per alterar les pàgines web perquè de la manera que nosaltres volem. Caldria que a la API de plugin de Chrome permetés alterar el comportament del DOM tenint accés directe d'una manera més profunda del que ho fa javascript. Això permetria, per exemple, que es visualitzi una sèrie de dades per pantalla, però que al accedir-les per DOM fossin unes altres.

[27] http://openpgpjs.org/
[28] http://kjur.github.io/jsrsasign/

- Actualment, un administrador de sistemes maliciós, amb accés al servidor de persistència, podria comprometre el sistema, per això el que seria interessant i saber que no s'ha modificat cap element del model de persistència, seria bo fer un mecanisme de blockchain amb mineria[29] com el que utilitza bitcoin. Això vol dir que el servidor només serviria per fer un discovery dels agents actius, i el resta de la comunicació es faria per P2P. A quest respecte el mecanisme WebRTC[30] que incorporen alguns navegadors ha propiciat que apareguin llibreries en javascript per convertir el navegador en una eina P2P [31] [32] [33] Així mateix, la persistència actualment no deixa tindre l'històric dels P_u ni dels K_u, i caldria mantindre l'històric per a poder recuperar missatges antics.
- Un usuari maliciós al facebook podria moure un comentari xifrat que s'ha fet a una conversació i copiar-lo a un altre banda sense consentiment de l'usuari, explotant aquesta vulnerabilitat. Seria interessant que, quan es xifra un missatge, crear una signatura basada en el hash de la unió del propi missatge amb el missatge anterior signada per M_u, si bé això faria que si es canvia un comentari anterior el missatge es donaria per dolent.
- Caldria ser curós amb el desenvolupament del codi en Javascript i fer servir un estil existent (com ara és la guia d'estil de Google [34], un ús més integrat del model de Promises[35] per a crides asíncrones, l'ús de la notació estricte[36] de javascript o fer servir alguna eina per el javascript amb tipus com ara bé es TypeStript[37]

[29] How Bitcoin Works Under the Hood. https://www.youtube.com/watch?v=Lx9zgZCMqXE
[30] http://www.webrtc.org/
[31] http://btappjs.com/
[32] https://github.com/Peer5/ShareFest
[33] https://github.com/muaz-khan/WebRTC-Experiment/tree/master/p2p-share
[34] http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml
[35] http://modernjavascript.blogspot.com.es/2013/09/promise-patterns.html
[36] http://ejohn.org/blog/ecmascript-5-strict-mode-json-and-more/
[37] http://www.typescriptlang.org/Playground/

Bibliografia

- Schneier, Bruce: Liars and Outliers: Enabling the Trust that Society Needs to Thrive 2012
- Schneier, Bruce: Applied Cryptography, 1996
- Rifà, Herrera, Domingo, Criptografía, Fundació per a la Universitat Oberta de Catalunya. 2006