New UK CA Web Portal

(Using a browser agnostic JS library to create Certificate Signing Requests and Key-Pairs)

david.meredith@stfc.ac.uk

john.kewley@stfc.ac.uk

sam.worley@stfc.ac.uk

suleman.tariq@stfc.ac.uk

jens.Jensen@stfc.ac.uk

Abstract (to read at own leisure)

The UK CA is trialling a new Web portal that creates Certificate Signing Requests (CSRs) and .p12 files using a client-side Javascript crypto library that is compatible across all modern Web browsers. Certificates can be requested, renewed and downloaded without ever sending the private key or password over the wire. In addition, since the Javascript library is served by the portal, this approach greatly simplifies dealing with certificates because users no longer need to install any client-side crypto software. The Javascript library is not used as a TLS replacement and all communications are served over HTTPS.

The following sequence of events is executed when requesting and/or renewing a certificate: a) the user provides the requested certificate attributes so that a new public/private key pair and CSR can be locally generated using Javascript, b) the private key file is encrypted with the user's password and is saved locally in a plain text file (encrypted PKCS#8), c) the CSR is POSTed to the CA for approval (via SSL), d) after signing, the user is emailed their certificate serial number, e) user accesses a second interface to download their certificate, f) after the certificate is downloaded, the user selects their local private key file and provides their password in order to create a local .p12 file using Javascript.

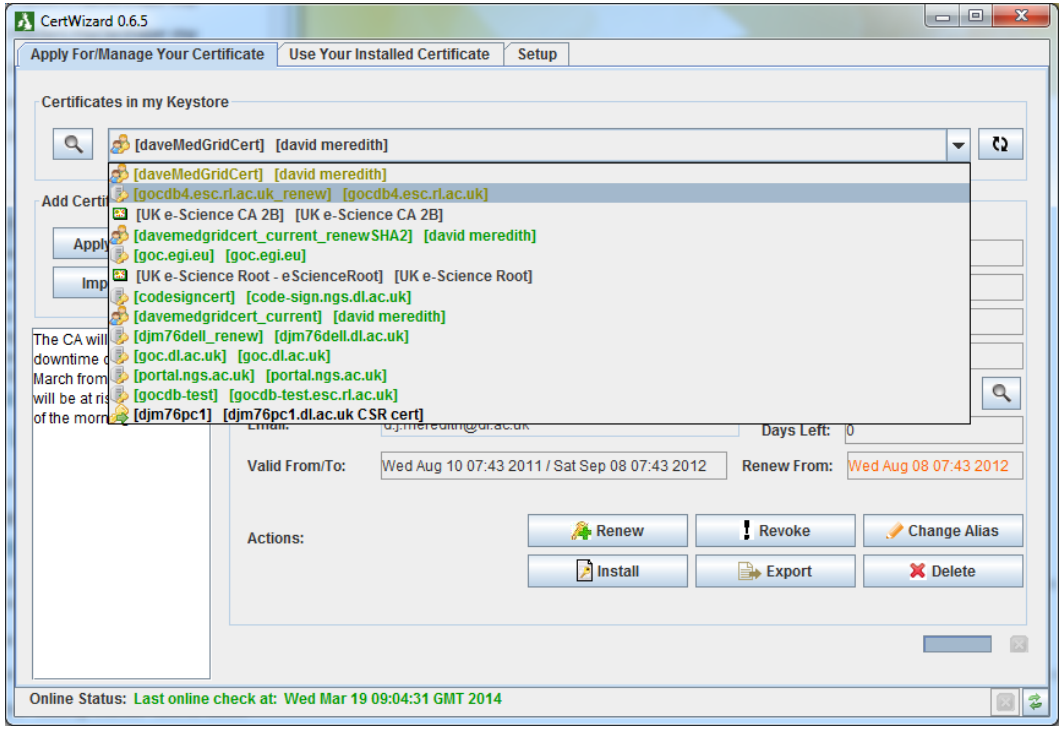
Certificate Signing Requests

- CSR involves creation of a Public/Private key pair
- CSR (pubkey + cert attributes) sent to CA for signing.
- Private Key stays on the client, not sent over wire

- So why are creating CSRs tricky?
  - Specialist crypto software is needed to create public/private keys on the client
  - Try to avoid key creation on the server and sending private key down the wire

UK Client Software 1

- **CertWizard** (connects to UK CA REST server)
  - Bouncy Castle API (create keys + CSR)
  - No external dependencies (no OpenSSL)
  - Uses PPPK protocol to auth client
  - PPPK **allows renewal of expired certs**



Store multiple certs/CSRs in standard .p12 keyStore file

UK Client Software 2

- **<u>CLI Scripts: Perl+Python</u>** (connect to UK CA REST server)
  - Need OpenSSL installed on client
  - Bulk processing of many certs
- **<u>OpenCA Portal</u>**
  - Served us well for yrs, but are now using v.old version
  - Therefore limited browser interoperability
  - Writes into browser key-store so updates have broken interoperability in the past
- **<u>New CA Portal (focus of this talk)</u>**
  - Uses recent(ish) JS Crypto lib which is browser independent (ForgeJS and SJCL)

Forge JS: http://digitalbazaar.com/forge/

- "A native implementation of TLS in Javascript and tools to write crypto-based and network-heavy webapps"
- Uses SJCL "Stanford JS Crypto Library"
- Note: We only use the JS Forge tools to create client-side keys, CSRs and .p12 files
- **We don't use Forge JS as an alternative to TLS** (we still use HTTPS - see later discussion)

https://ca-dev2.ca.ngs.ac.uk/caportal/

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options

UK CA | RA List | Certificates | Login/MyCert

In development (more coming soon)

UK Certification Authority Portal

Login / View My Certificate

- You **DO** need to have a UKCA issued user certificate in your browser to login.
- RA's please login to view RA Actions (new RA menu items will be added).
- To re-authenticate using a different certificate, clear your ssl cache/restart browser.

Request New User Certificate

You **do NOT** need to have a UKCA issued certificate in your browser.

Request New Host Certificate

- You **DO** need to have a UKCA issued user certificate in your browser to request a host certificate.
- To re-authenticate using a different certificate, clear your ssl cache/restart browser.

Download Certificate

- You **do NOT** need to have a UKCA issued user certificate in your browser to download a certificate.

Courtesy UK CA and STFC.

Demo: View Browser Certificate

Forge test PKCS#12 in | Forge test PKCS#12 in | Forge test PKCS#12 in | SSL Error | Cert Owner Home

https://ca-dev2.ca.ngs.ac.uk/caportal/cert_owner

UK CA | RA List | Certificates | Login/MyCert | RAOP Home | RAOP Actions | CAOP Home

In development (more coming soon)

Your Certificate Details

Certificate Attribute	Value
Serial Number (cert_key)	3589, (hex: e05)
Common Name (CN)	david ra meredith
Distinguished Name (DN)	CN=david ra meredith, L=DL, OU=CLRC, O=eScienceDev, C=UK
Issuer DN	CN=DevelopmentCA, OU=NGS, O=eScienceDev, C=UK
Email	david.meredith@stfc.ac.uk
Status	VALID
Not Before	Tue Nov 19 11:15:13 GMT 2013
Not After	Sun May 18 12:15:13 BST 2014
Signature Algorithm	SHA1withRSA
Type	X.509
Version	3

Your Roles with the UK CA

ROLE_CAOP	You have CA Operator privileges.
ROLE_CERTOWNER	You have a certificate issued by the UK CA.
ROLE_RAOP	You have RA Operator privileges.

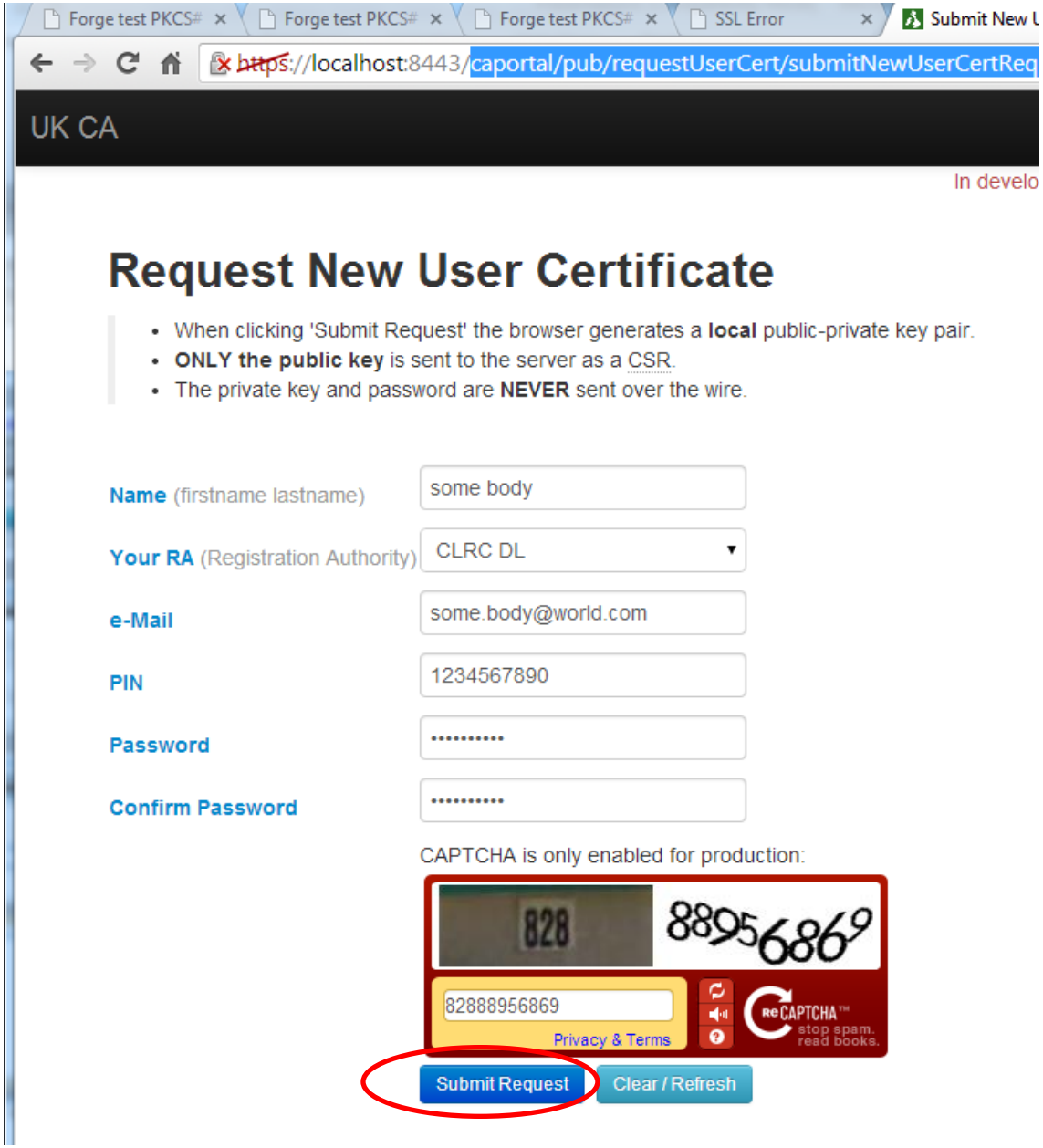
Certificate Actions

Revoke Certificate | Reason to revoke (value is required)

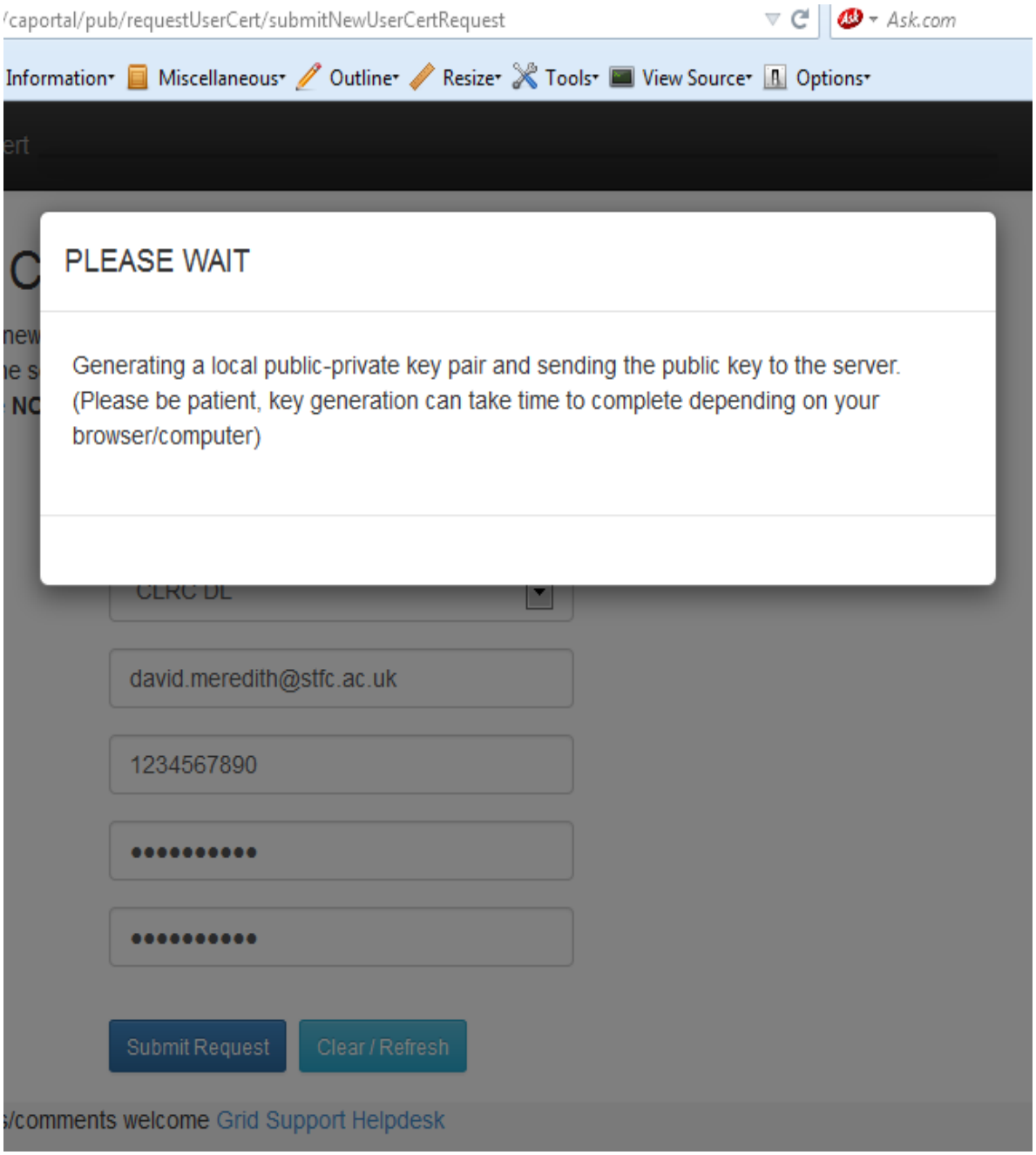
Some pages need a Certificate to gain access

Menu bar and available pages are contextual based on roles

Demo: Get a Certificate Step 1: Submit Request

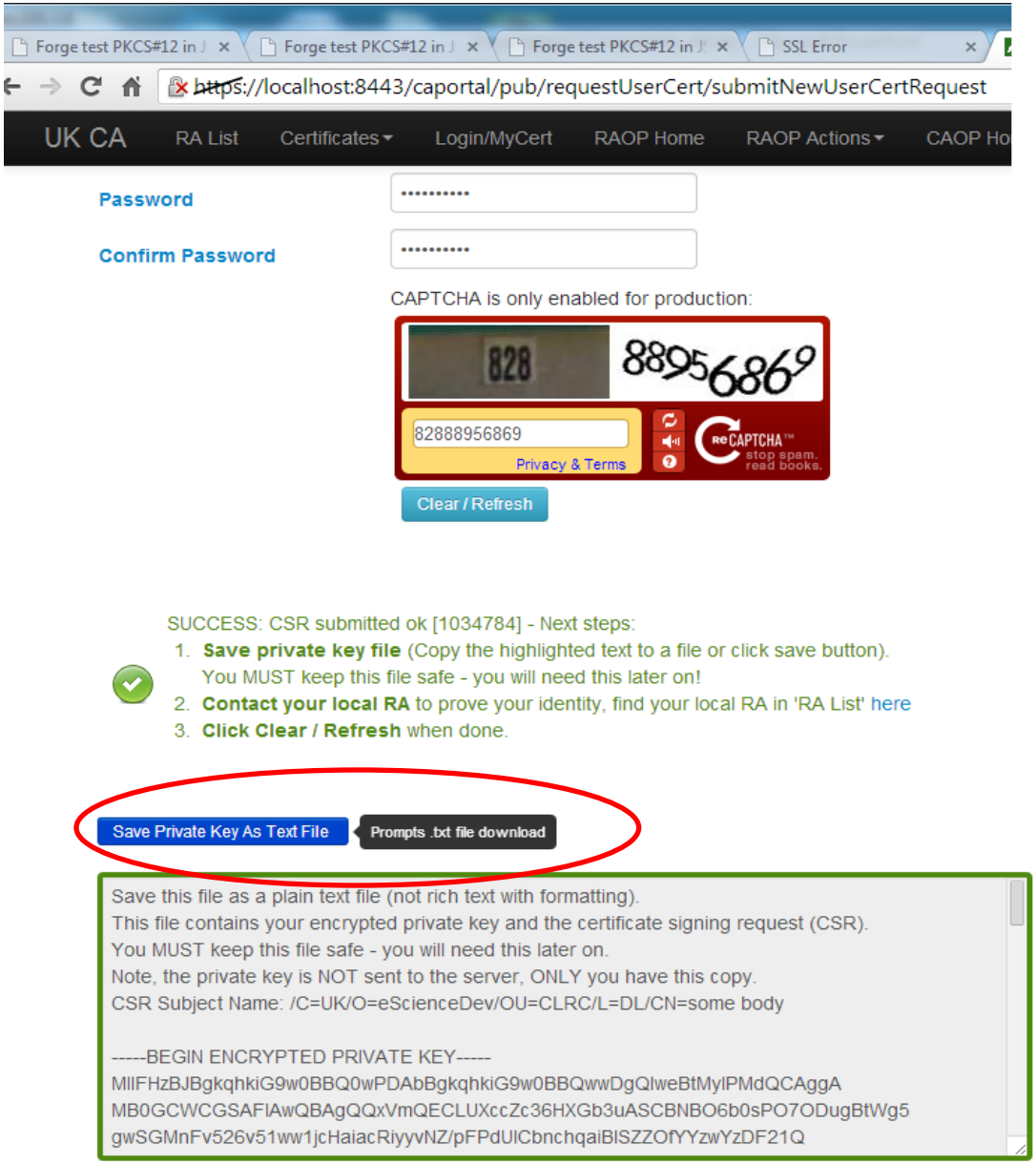


- Access = permitAll (client certificate not required in browser)
- Fill in form
- Click ‘Submit’



- CSR + Private key are created via Forge JS in browser
- CSR is sent to server via ajax POST (https)
- Private key + password are **NOT** sent to server

Demo: Get a Certificate Step 2: Save Private Key

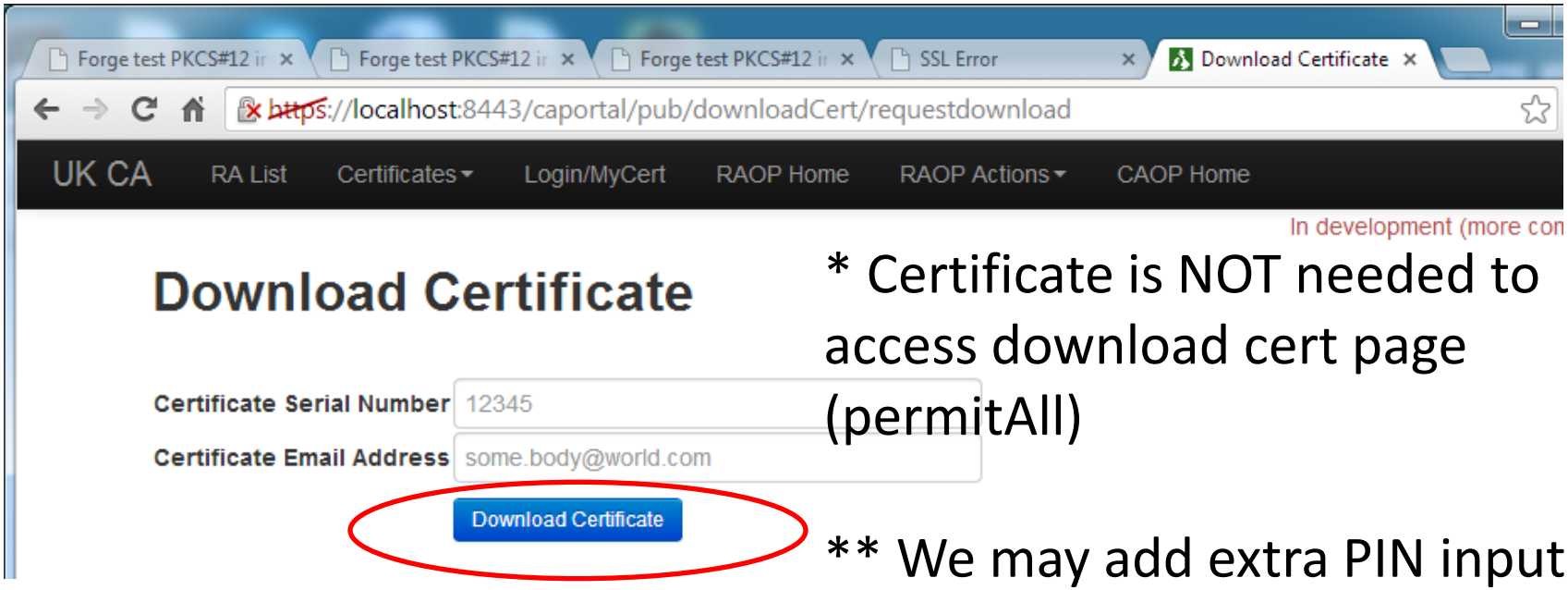


- Response received on same page via ajax
- Click ‘Save Private Key as Text File’ to save PKCS#8 (or manually copy from textarea).
- Click ‘Clear/Refresh’ to finish
- No browser history - can’t go ‘back’ !
- Only user has key

Demo: Get a Certificate Step 3: Download Cert

- After certificate is created/signed, you are emailed your Certificate Serial Number

- Go to 'Download Certificate' page, enter serial number, click 'Download Certificate'



* Certificate is NOT needed to access download cert page (permitAll)

** We may add extra PIN input field to authenticate user

Demo: Get a Certificate3 Step 4: Create .p12

Certificate [3650] Downloaded OK

Certificate Attribute	Value
Serial Number (cert_key)	3650, (hex: e42)
Common Name (CN)	dave.test2.dl.ac.uk
Distinguished Name (DN)	CN=dave.test2.dl.ac.uk,L=DL,OU=CLRC,O=eScienceDev,C=UK
Issuer DN	CN=DevelopmentCA, OU=NGS, O=eScienceDev, C=UK
Email	david.meredith@stfc.ac.uk
Status	VALID
Role	User
Not Before (Starts)	Wed Feb 26 11:31:10 GMT 2014
Not After (Expires)	Mon Aug 25 12:31:10 BST 2014
Signature Algorithm	SHA1withRSA
Type/Version	X.509 / 3

Clear/Refresh Page

Save Certificate Bundle (Combine Certificate + Private-Key)

Note, your private-key and password are **NEVER** sent over the wire to the server (how?)

-----BEGIN CERTIFICATE-----
MIIFWjCCBEKgAwIBAgICDkIwDQYJKoZIhvcNAQEFB
QAwSTELMAkGA1UEBhMCVUsx
FDASBgNVBAoTC2VTY2IlbmNIRGV2MQwwCgYDVQ
QLEwNOR1MxFjAUBgNVBAMTDURI
dmVsb3BtZW50Q0EwHhcNMTQwMjI2MTEzMTEwWh
cNMTQwODI1MTEzMTEwWjBdMQsw
CQYDVQQGEwJVSzEUMBIGA1UEChMLZVNjaWVuY
2VEZXYxDTALBgNVBAsTBENMUkMx
CzAJBgNVBAcTAkRMMRwwGgYDVQQDExNkYXZILn

ONLY you have this copy.
CSR Subject Name:
/C=UK/O=eScienceDev/OU=CLRC/L=DL/CN=dave.test2.dl.ac.uk
1234567890
(cert id 3650)

-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFHzBJBgkqhkiG9w0BBQ0wPDAbBgkqhkiG9w0BB
QwwDgQIWHtms/BsyOsCAggA

Choose File privateKeyAndCsrHost.txt

Password ••••••••••

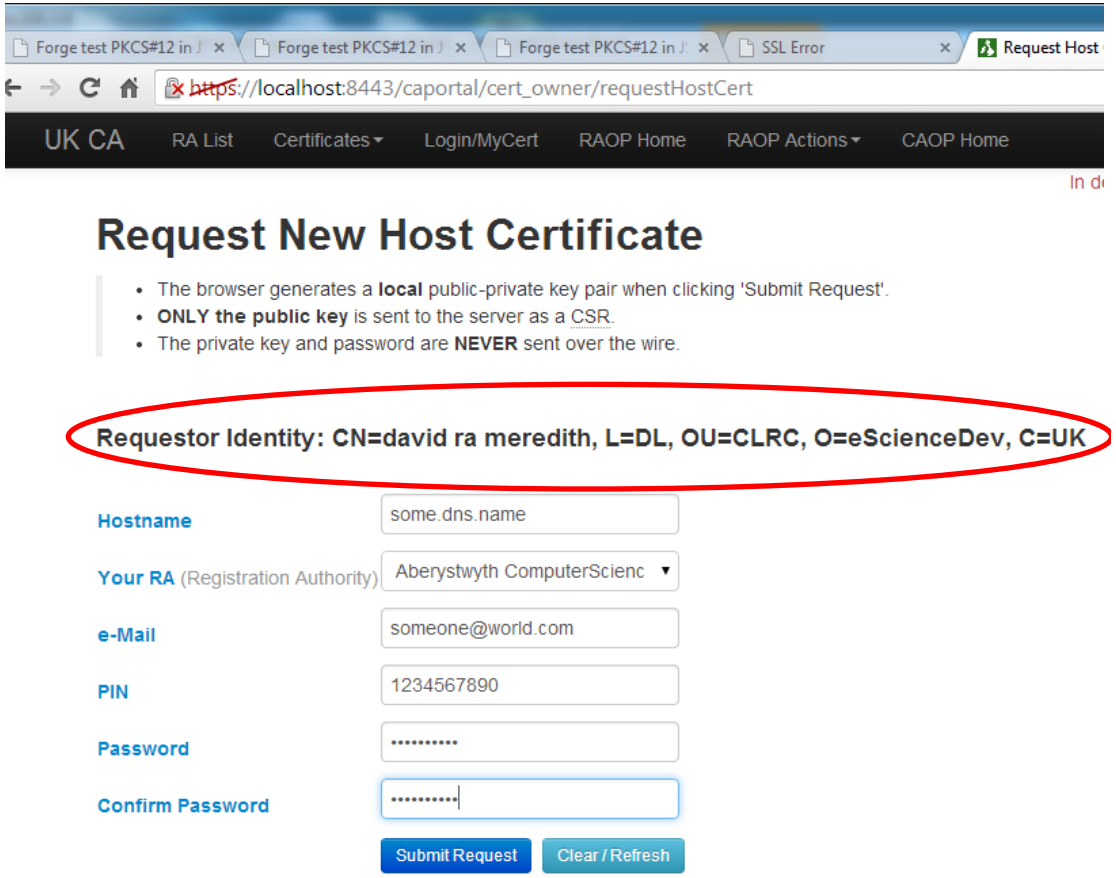
Save Certificate (.p12)

Will prompt .p12 download if valid private key and password entered

certBundle.p12

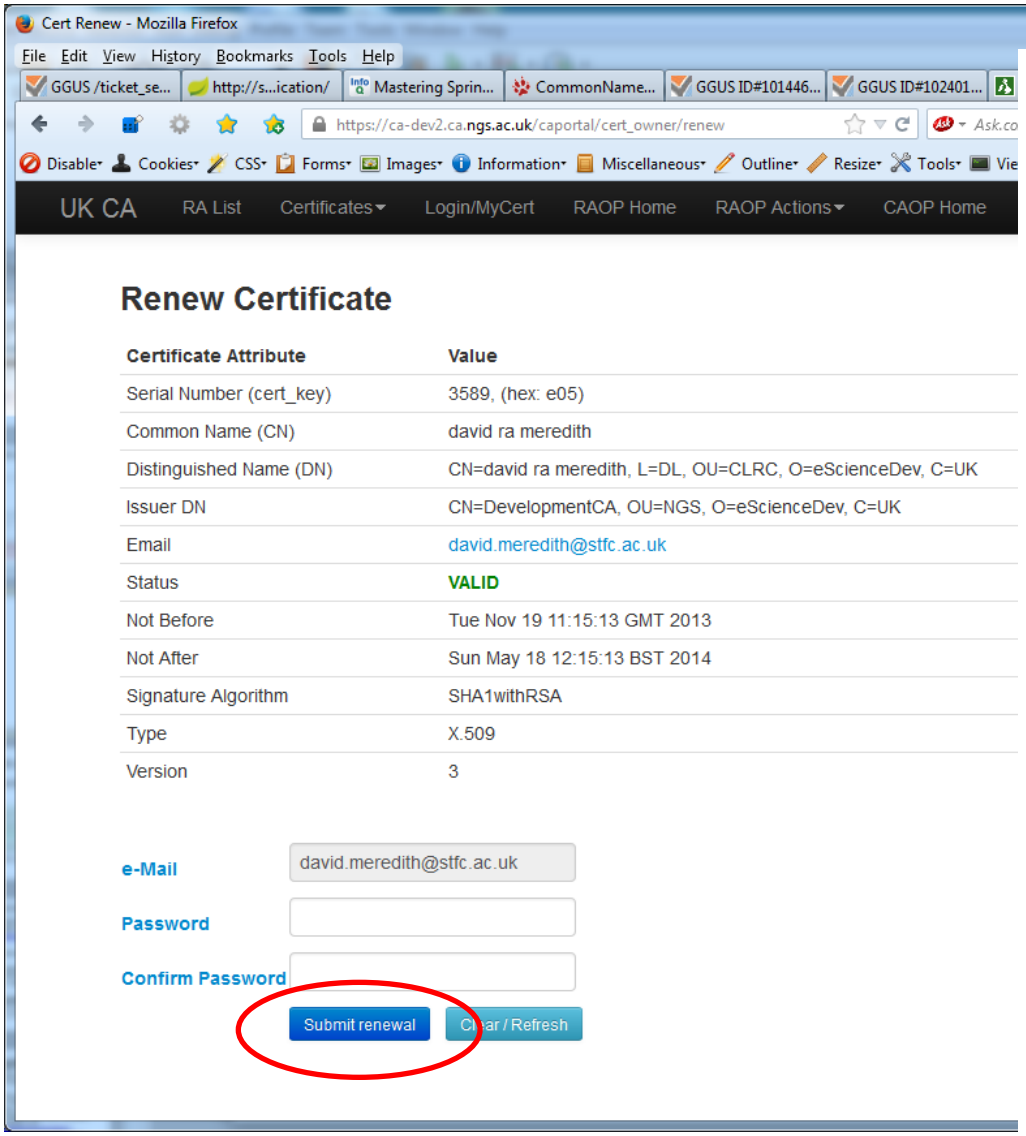
- Cert is downloaded
- Choose private key file
- Enter password
- Click ‘Save Certificate’
- ForgeJS creates a <u>local</u> .p12 file combining private-key + cert
- Password + private key are **<u>NOT</u>** sent to server
- .p12 downloaded

Demo: Request Host Cert



- Access = clientCert Required
- CAPTCHA not needed (user cert adequate)
- Response returns encrypted PKCS#8 private key as .txt file (see previous slide)

Demo: Renew Cert



- Certificate in browser is required to access page
- CAPTCHA not needed (user cert adequate)
- Save local private key .txt file (same as in previous slide)
- Use Download Cert page to download renew certificate

POSTing CSR via JQuery AJAX

- Private key/password are **NOT** sent to server:

```
// Only send public pem.csr to server (pem.privateKey) is NOT sent
$.ajax({type: "POST", url: link.attr("href"),
    data: $.param({ pin: pin })+
            "&"+$.param({ email: email })+
            "&"+$.param({ csr: pem.csr })+
            "&"+$.param({ recaptcha_challenge_field: my_recaptcha_challenge_field })+
            "&"+$.param({ recaptcha_response_field: my_recaptcha_response_field }),
    success: function(text) {
        if(text.substring(0, 7) === "SUCCESS") {
            MvcUtil.showSuccessResponse(text+' - Next steps:'+
```

Only PIN, email, CSR, CAPTCHA fields are sent to the server as shown above

Javascript Crypto?

- There are compelling arguments **against** using JS crypto to replace TLS, e.g. see:
  - www.matasano.com/articles/javascript-cryptography/

- JS Math.random() lacks entropy for true TLS purposes

- A hard problem to solve in JS; JS don't have as many sources of good entropy as there are for languages that have access to the disk
- An issue if using JS crypto as TLS replacement:
  - TLS requires some random bytes to be sent in the clear
  - If an attacker can steal those and figure out what *next sequence of* bytes will be => man-in-middle attack

However…

1. **We don't use JS crypto as a TLS replacement, we still use HTTPS**

2. We don't send some initial random bytes in the clear as required in TLS

3. Just require the random number to be unpredictable enough for key generation

4. Check no public key clashes in DB (none yet)

   http://blog.meadhbh.org/2013/08/in-defense-of-javascript-cryptography.html

Therefore…

- For our use case, JS crypto is likely ok
- Forge PRNG uses Fortuna algorithm to gain entropy from a pool of sources:
  - Page load times, navigator object details, two math random functions, an attempt also made at mouse movements + keyboard presses
    - http://digitalbazaar.com/2010/07/20/javascript-tls-2/

Fall-back Solutions

1. We use HTTPS so we could create the random number on the server and seed the JS script! (already requested on ForgeJS site)

2. With one portal config change and we can create the CSR/Key on server and send the client the PrivateKey in the response
   - Exactly the same user interface/GUI
   - But breaks private key policy (this is changing though)

```
createCsrOnClientOrServer=client
```

XSS Attacks

- **“Q. Why can't I use TLS/SSL to deliver the Javascript crypto code?”**
  - “A. You can. It's harder than it sounds, but you [can] safely transmit Javascript crypto to a browser using SSL. The problem is, having established a secure channel with SSL, you no longer need Javascript cryptography; you have "real" cryptography.”
    - www.matasano.com/articles/javascript-cryptography/

Nope, we still have a use-case for creating CSRs!

- **So, What's hard about deploying JS over SSL?**
  - You MUST guard against XSS attacks

XSS Attacks

- **MUST send <u>all page content over SSL/TLS</u>**. Otherwise, attackers could hijack the crypto code using the least-secure connection that builds the page.
- **Never use the "eval" function on data loaded from the server**
  'eval' is flexible – can "run" any string, thus any kind of code could be easily injected.
- **Never load html content sent from the server**
  Loading 'html' chunks from the server is another easy way to inject harmful code: e.g. server could push this little snippet:

```
<script src="/hijack.js"/>
```

- Always sanitise untrusted content provided by client (validate input + output, consider persistent XSS attack)
- https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet

Other Non-TLS Replacement JS Crypto Use-Cases

- End-to-end message encryption
  - Server never has access to your decrypted message, and unless you explicitly share your keys with the server, they never will.
- ‘Zero Knowledge’ WebApps
  - e.g. https://clipperz.is - An online Password Vault
  - Called ‘Zero Knowledge’ because server knows nothing about its users’ data
  - Everything you store online is locally encrypted by your browser using JS crypto (and sent via HTTPS)

Technical

Server Side:

- Tomcat + SSL (clientAuth="want" – same as 'SSLVerifyClient=optional')
  - Some portal URLs need a client cert to gain access
  - Some portal URLs are permitAll (no client cert needed)
- Spring: IoC, JDBC-pool, TX, DAOs, @Service, @Repository, @Inject…
- SpringMVC: @Controller, @Valid…
- Spring Security
  - Authentication and access-control framework
  - Provides certificate AuthenticationProvider
  - Provides AOP security rules to secure business layers
- Bouncy Castle Java API (most recent 1.50) for crypto, CSR validation, x509 processing

Client Side:

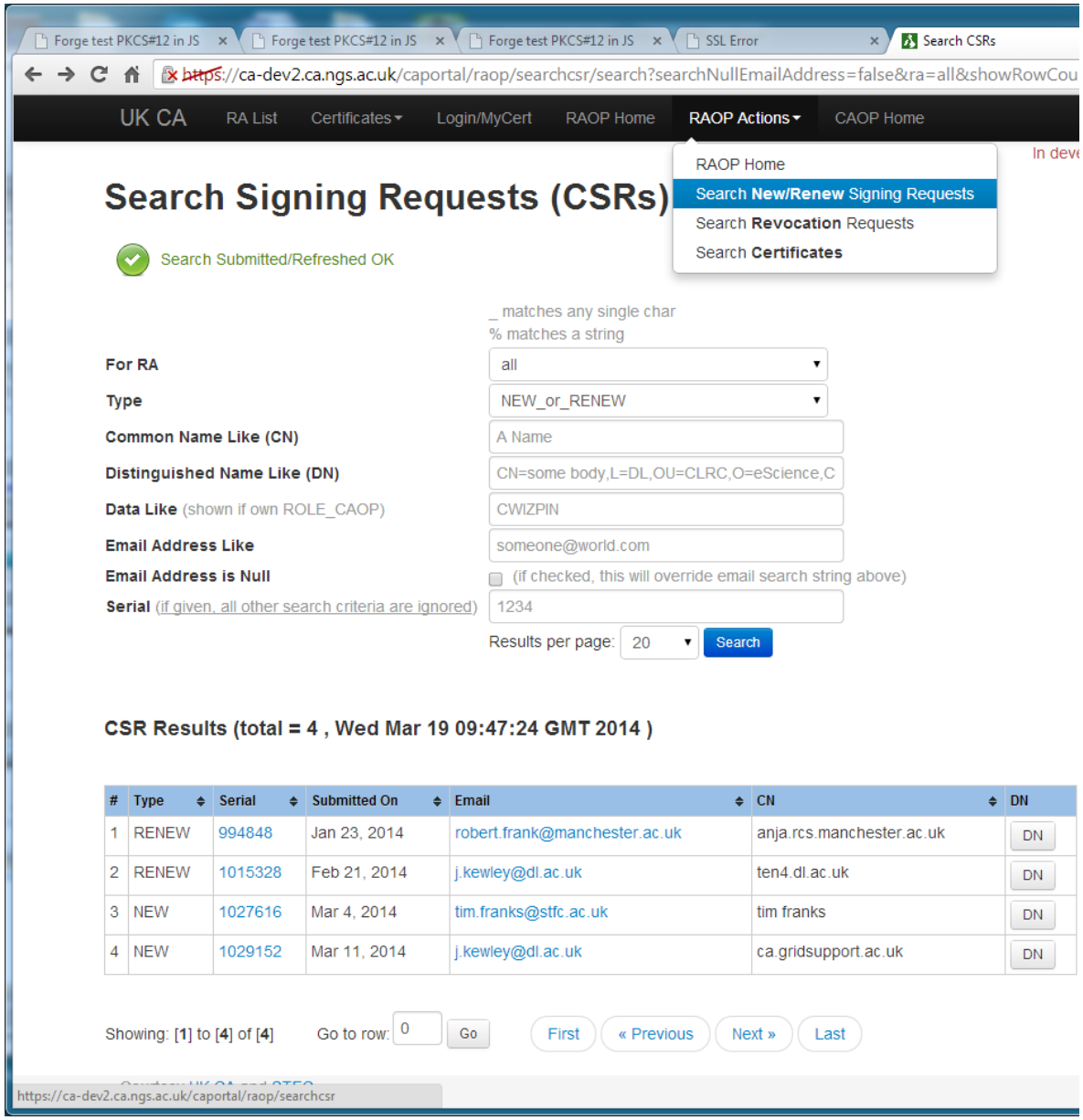
- JSP, HTML(5), JQuery / AJAX / Bootstrap, Forge JS Crypto, FileSaver.js (for saving files locally generated in browser), reCAPTCHA

Summary

- JS crypto appears to be suitable for our use case since we are not using it to replace TLS (akin to other 'zero-knowledge' WebApps).

- Browser agnostic
- Forge JS + Stanford JS Crypto Library seem to run fine in all modern browsers.
  - No interaction with browser or system keyStore so are protected from breakages caused by updates
  - Free to save encrypted PKCS#8 (.txt file) on e.g. USB and later on download cert on another pc.

Demo: /caportal/raop/* (RA Ops)



- Certificate with RAOP/CAOP role is needed to gain access
- Search CSRs, CRRs, Certs
- Approve / revoke etc…