Crypto – for the masses

Guido Aben, AARnet
Xander Jansen, SURFnet
Gijs Molenaar, Pythonic

eResearch Australasia 2012
Power Users are here – big, but few. Here be metadata, ontologies, grids, HPC, GPU coding…

"The Chasm"

"Masses" beyond this point

“Regular” researchers are here; more modest requirements, but lots more of them. What keeps them up at night?

"the masses"? You can’t say that in public!
Total users

- Users: Total user count: 4182 users

Total uploaded files

- Files: Total uploaded files: 51062 files

Total uploaded bytes

- bytes: Total uploaded bytes: 20283769474915 bytes
So yes, we seem to reach a reasonable proportion of a previously hard-to-address audience.

That’s an opportunity to add (well, re-introduce) two “cultural” improvements:

1) Confidentiality (implemented with crypto)
2) Provenance (implemented with timestamping)
crypto

Stanford Javascript Crypto Library

http://crypto.stanford.edu/sjcl/

- Only one of its kind (apparently...)
- Very complete. Only has 2 limitations:
  - Doesn’t support streaming data
  - Only supports plain text.
-- so we need to Base64 the blobs...

(note: key distribution is out of scope)
More ways to crypt a cat?

A different approach would be to generate a zip file with AES encryption locally. The resulting crypto file could then be distributed without the use of the download javascript code...

...but there is no javascript ZIP library that supports this... and you certainly don’t want to kludge in AES into JSZip yourself!
HTML5 requirements

“File” object
(to open the file in the first place)
FileSystem API / Blob slice
(to chunk the file and write it back to disk)
FileReader
(to access a chunk’s contents)
WebWorkers
(to ensure crypting is non-blocking)
Upload:
Open file, slice file, ask user for key, encrypt blob, upload encrypted blob to server, meanwhile encrypt next blob
Download (actually harder!)

Get file in chunks by adding a ‘Range’ header to the XMLHttpRequest (no magic; standard html4)

Decrypt using SJCL

Reconstruct file on disk

This part is tricky; standard file reconstruct would use BlobBuilder, but this reconstructs in memory – won’t scale.

HTML5’s FileSystem API should be used for reconstructing the original file on disk, but this API is only supported by Chrome (Firefox almost there)
HTML5 is a “living standard”.
Oh the fun we had.

Debugging seems to be impossible.
Chrome 17 has some support for debugging Javascript in a webworker, but that doesn’t at all work well. Forces you to code with maximal error checking and logging if you want a webworker to actually work.

W3C (the html5 standards body) changed the semantics for FileSlice on the fly, prompting Mozilla to implement a proprietary MozSlice, and then the standards body reverted to the previous definition for Slice, just when we had recoded to use MozSlice...

WebWorkers stopped working with Firefox 8 when used from localhost. Fixed again in 9, but imagine the confusion...
Crypto benchmarks

Speed
- Encryption and upload (local): 1.39 MB/s
- Decryption and download (local): 1.25 MB/s

On an intel i5 2500K, 16 GB HD, OCX Vertex 3 SSD 120 GB, connected to 10Gbit/s network

Crypted filesize
- crypted/plain size ratio: 1.78x
Data is first Base64 encoded; inflates by ~1.3x
AES encryption in turn again inflates by ~1.3x
Crypto trials:

- 3 academic hospitals in .nl signed up for trial
- Crypted exchange of medically sensitive data
- Waiting for Firefox HTML5 FilesystemAPI to become stable enough...

...soon!
Trusted timestamping

1. **Within a company**
   - Calculate hash
   - Data
   - Send hash to TSA
   - 1011...10101
   - 1011...10101 + Timestamp
   - Calculate hash
   - 0010...01011
   - Apply private key of the TSA
   - Signed timestamp and hash are returned to requester
   - 0010...01011 + Timestamp
   - Store together

2. **Timestamping Authority (TSA)**
   - This is a digital signature of the hash concatenated to the timestamp
   - 0010...01011 + Timestamp

VANguard is a Whole-of-Government program delivered by the Department of Industry, Innovation, Science Research and Tertiary Education (DIISRTE). VANguard, together with AUSkey, is the Australian Government’s preferred authentication solution for secure B2G and G2G transactions.

VANguard’s Timestamping Service records the date and time of an electronic transaction using certified time from the National Measurement Institute. An agency can use the timestamp as evidence of the time and date that a transaction was completed online.
Timestamping trials:

- spent a bit long implementing timestamping in php... guilty as charged
- In our defence, RFC3161 mandates ASN.1
- ASN.1 and php are not friends

Have resolved to eat our hat and revert to calling JAVA API from php.
That was 3 weeks ago. It’s all working now though!

...soon!
sneak preview to compensate:
Personal, cloud-based storage?

• High demand from users
  ...who now resort to doing their own thing!

• Commercial services pricey
  ...even when bought in bulk

• Most are domiciled in the U.S.
  ...make up your own mind about what that means...

• Can be slow, off-net
• Based on geographically diverse anycasted ownCloud instances with Hadoop as underlying replication substrate
• Integration with LDAP or SAML2
• Supports sharing between users
• Complements cloudstor’s large-file transfer facility

• Some nodes hosted on , some on cloud infrastructure made available by the fine people at

• Quite open to other asset owners contributing nodes, storage...
• In Australia, on-net – high speed, high reliability
• 100GB free per user from day one
• More capacity to come online in 2013
Murchison Radio-astronomy Observatory

AARNet POP

< 155 Mbps

< 622 Mbps

< 1 Gbps

< 2.5 Gbps

< 10 Gbps

WDM Transmission

Coherent detection WDM Transmission
Interested? Let us know!

eresearch@aarnet.edu.eu