Design Exploration of Transparency Enhancing Technology for Government

Mathijs Houtenbos
Supervisor: Guido van ’t Noordende, Whitebox Systems

University of Amsterdam

2016-02-02
Introduction
- Motivation
- Research question
- Requirements

Existing Technology
- DigiD
- eID / Idensys
- MijnOverheid

Architecture
- Centralized
- Distributed
- Federated

Design

Conclusion

Questions
Motivation

Vision of Dutch digital government by 2017 and 2020

Citizens have right of transparency and control over their data
Research question

How could transparency enhancing technology be designed for use by the government without negatively impacting citizen privacy?
**Requirements**

**Citizens rights**
- View your data
- Correct your data
- Easily accessible
- Authorize others
- Granular access

**Government**
- Authentication
- Security
- Electronic ID
- Foreign eIDs
- Digital services
Existing Technology

- DigiD
- eID / Idensys
- MijnOverheid

These example systems are all part of the solution currently implemented by the Dutch government.
DigiD

- DigiD
- eID / Idensys
- MijnOverheid

Secure* identity provider for government sites.
Secure identity provider for government and business providing STORK 3 / 4 level authentication.
DigiD

eID / Idensys

MijnOverheid

Digital postbox for official government mail.
DigiD

eID / Idensys

MijnOverheid

Digital postbox for official government mail.
Distribution Architecture

- Centralized
- Distributed
- Federated
A centralized server architecture where all agencies (a1-aN) and users (u1-uN) use the same central service.
A distributed architecture where all agencies and users are peers.
A Federated server architecture where all agencies (a1-aN) and users (u1-uN) choose which service they use.
When we evaluate the requirements for the transparency enhancing system, with an additional focus on the requirement for *privacy by design*, it seems most promising to use a federated architecture for our design.
Design based on a federated architecture where all users can choose their **home service** by storing a signed pointer in a public directory.
Design - Home service

Public home service example:
User $u_1$ indicates his/her home service is $s_5$. 
Design - Home lookup

Public home service lookup example:
Agency \texttt{a1} needs to find \texttt{u1} home service.

Transparency Enhancing Technology  Mathijs Houtenbos  Supervisor: Guido van ’t Noordende, WHITEBOX SYSTEMS  19 of 30
Agency file push example:
Agency **a1** pushes a file to their user **u1** namespace at service **s5**.
Design - Key issuance

**Government key issuance**

- You Physically
- Identification
- Government Official
- \(\text{visit}\)
- \(\text{issue}\)
- eID Card
- Certificate

**Own keypair registration**

- You Physically
- Identification
- Smart Card
- \(\text{visit}\)
- \(\text{vouch}\)
- Certificate
Symmetric file encryption with asymmetric key encryption

- File
- Symmetric key
- Public key

= Encrypted file
= Encrypted key

Asymmetric key decryption with symmetric file decryption

- Encrypted file
- Encrypted key
- Private key

= File
= Symmetric key
Iteratively request a single user file:

1. Send user lookup
2. Return user home
3. Sign request
4. Send request
5. Authorise + Log
6. Return key + meta
7. Request file blob
8. Return file blob
9. Client decodes file
Users have namespaces with separate access control, for example:

**user12345678/belastingdienst**

- Signed ACL
- Symmetric key
- User12345678 key
- Belastingdienst key
- Accountant key

**user12345678/gemeente_amsterdam**

- Signed ACL
- Symmetric key
- User12345678 key
- E. van der Laan key
- My wife’s key
Design - Add file(s)

Add one or more files to a namespace:

1. Sign request
2. Send request
3. Return user home
4. Resend request
5. Authorise + Log
6. Return key + OK
7. Encrypt file(s)
8. Send file(s) + meta
9. Verify + Log
10. Forward file blob(s)
11. Report status
12. Forward status
Conclusion

Transparency enhancing technology that does not negatively impact user privacy is feasible.

Advantages
- Privacy by design
- Strong crypto
- Verifyable
- Scalable
- Future proof

Ideal scenario
- Only hardware tokens
- No data leakage
- Independent audits
- Large infrastructure
- Forward compatibility
Recursively request multiple user files through directory proxy:

1. Sign request
2. Send request
3. Forward request
4. Authorise + Log
5. Return key + meta
6. Request file blobs
7. Return file blobs
8. Forward file blobs
9. Client decodes files
Register your own additional smart card

Primary card + Certificate + New card → vouch → certificate
There are two scenario’s for compromised key revocation:

**User has/had multiple keys**
- Sign key revocation certificate with other key
- Re-sign important data with new primary key
- Effect is immediate
- No interruption
- No data loss

**User has lost last/only key**
- New key must be issued in person (STORK 4)
- Revocation takes duration of processing
- All namespaces need to be rebuilt
- May result in data loss