Self-Protecting Documents for Cloud Storage Security

Manuel Munier[1] Vincent Lalanne[1] Magali Ricarde[2]

[1] **LIUPPA**
Univ Pau & Pays Adour
Mont de Marsan, France
manuel.munier@univ-pau.fr
vincent.lalanne@univ-pau.fr

[2] **BackPlan**
Project Communication Control
Pau, France
magali.ricarde@backplan.fr

TSIS 2012 (TrustCom)

June 25-27, 2012

Liverpool, UK

This paper

- Information system security is currently one of the most important goals for enterprises

- The problem becomes even more difficult when documents go "outside" the organization
  - ⤳ *storage services are outsourced (eg cloud)*
  - ⤳ *a user wants to "checkout" a document from the information system to work offline*

⇒ Problem: how to ensure security and privacy for the document once it has left the information system ?

This paper

- We use an object oriented approach to encapsulate within the document itself some security components (access control, usage control, traceability,. . .)

⇒ The "intelligent" document self-manages its own security
→ *data centric solution*

⇒ In previous work we defined a secure autonomous document architecture for Enterprise Digital Right Management

Table of contents

1. Context of Information Sharing
2. Autonomic Documents
3. Platform Implementation
4. Conclusion & Perspectives

Context of Information Sharing

- Information sharing ?
  - collaborative work for **enterprises**: reports, medical records, design documents (with related reviews & certifications), whole project as bulk document,. . .
  - documents can go outside the company where they have been designed (export from IS). . . and return (import updated documents)
  - we have to control how partners use the documents
    - access control (of course. . .)
    - usage control (cf. obligations)

      eg, user has to read a section before writing his review
    - traceability, trust (cf. metadata, auditing,. . .)

⇒ **D**igital **R**ight **M**anagement approach with user licenses

→ **E**nterprise-**DRM**

Context of Information Sharing

Document security enforced on server side

- "Classic" DRM architectures
  - server ciphers the digital document & build user license
  - client side viewer deciphers the document according to rights found in the license

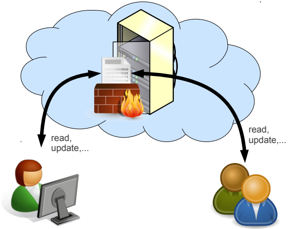
⇒ well suited for "classic" multimedia documents

- content providers & read-only viewer clients
- the document is created once and never changes
- security policy remains the same

Context of Information Sharing

Document security enforced on server side

- E-DRM architectures
  - documents are not "static" ⇒ updates, item deletion, new data,. . .
  - security policy may change during the document lifecycle



⇒ client application has to contact the server to check access & usage rights for user actions

- server can also provide audit facilities
  - → *traceability allows to control how information is used & to demonstrate that it has been used as defined in the security policy*
- off-line use by leasing the document for a finite period of time

eg Adobe LiveCycle Policy Server

Context of Information Sharing
Specific needs

- Our specific needs
  - users can update shared documents (≠ "classic" DRM)
  - usability with legacy applications: share resource on cloud, email attachment, USB flash drive,. . .

    → *users could exchange docs without having to work on a server*
  - multi-site enterprises, virtual enterprises, nomadic users

    → *using a centralized site for working (actions) is seen as a constraint*

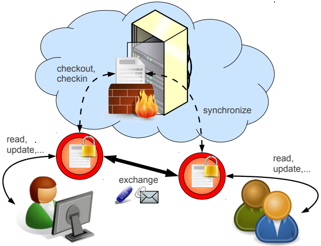
    → *information system ≡ data warehouse to manage & synchronize exchanges between users*

⇒ "Classic" centralized architectures do not suit these needs

Context of Information Sharing

Object oriented approach

- OO approach to encapsulate
  - **data**: content of the document itself
  - security control **components**: access control, usage control, traceability & metadata, collaborative work management,. . .



⇒ autonomic document self-manages its security

→ *such a document is a kind of information system on its own*

→ *data centric solution*

Context of Information Sharing

Object oriented approach

- How to "use" such a document ?
  - when "opening" the document, the user should provide her/his license
  - security control components are configured according to security rules contained in the user license

    → *permissions, obligations, metadata required,. . .*
  - they check all the accesses to information (embedded IS)

    → *access control, usage control,. . .*

    → *metadata recording*

    → *traceability, trustworthiness management,. . .*

⋮

- user can then:
  - forward the document to another user (who handles the document according to her/his own license)
  - publish the amended document on the data warehouse (sync)

Context of Information Sharing

Example: Oil & Gas project

- Project: construction of a pipeline or an oil installation
- Many documents: specifications, drawings, records of expertise, procedures, certifications,...
  - → relationships between documents (eg reviews and certifications binded to design documents)
- Many partners: civil engineering, pipefitters, instrumentation engineering, land surveyor, utilities,...
  - → metadata
    - *traceability, validation (certify checkpoints)*
    - *confidence & trustworthiness indicators, impact risk of a change, performance indicators*
    - *in case of litigation: proof of conformity, digital forensics,...*
  - → security policy
    - *(contextual) access control*
    - *usage control: required actions, collective obligations,...*

Context of Information Sharing

Example: Oil & Gas project

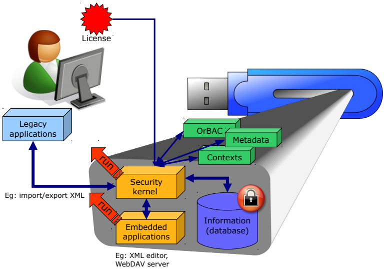
- Information management
  - → now: papers, folders/files on "simple" file server
  - → emerging: document registry
    - *document management service (versions, configuration,. . .)*
    - *collaboration workflow applications*
    - eg *BackPlan[1]: Project Communication Control*
  - → future: cloud storage (& security)
    - *use documents from laptops, smartphones, tablets*
    - *access anytime/anywhere*
    - *structured & complex documents, advanced security policies*
    - *traceability, digital forensics, indicators*
    - eg *self-protecting documents*

[1] http://www.backplan.fr/

Autonomic Documents

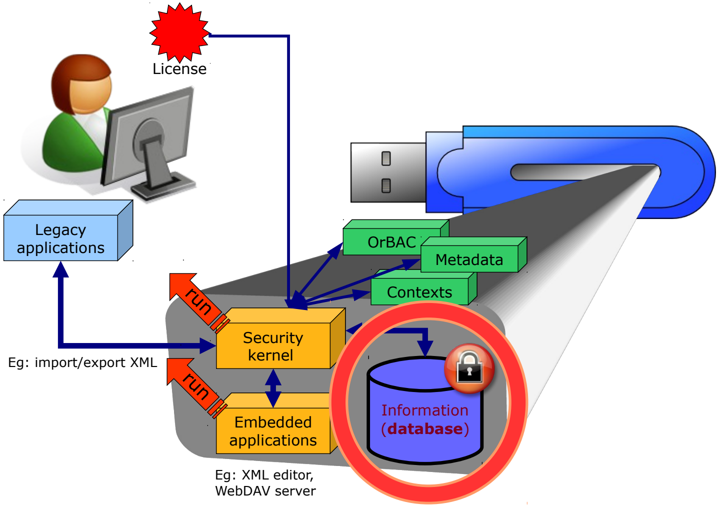
Overall architecture

- Main components
  - embedded database
    - → *contents of the document, metadata*
  - security kernel & security modules
    - → *enforce the security policy*
    - → *monitor all actions on the doc*
  - embedded applications & services
    - → *dedicated tools*
    - → *export/import mechanisms*
  - user license
    - → *permissions, prohibitions, obligations*
    - → *metadata to be collected*



Autonomic Documents

Embedded database



Autonomic Documents
Embedded database

- In previous work[2] we defined a new data model for embedded information system
  - multi-view approach to ensure both confidentialty & integrity
  - formal model to store data & calculate views
  - mapping of user actions to "low level" actions
- Dilemma privacy vs. integrity
  - → **Confidentiality**: How to prevent the disclosure of information to unauthorized individuals (or systems)
    - *breach of access control: someone can perform actions without the proper permissions*
    - *system behavior allows one to deduce the existence of hidden information*
  - → **Integrity**: How to avoid data to be modified without authorization
    - *someone accidentally (or with malicious intent) modifies/deletes data by side effects of a legitimate action*

[2] M.Munier, *"A multi-view approach for embedded information system security"*, CRiSIS 2010

Autonomic Documents

Embedded database - Multi-view approach

- We decouple "*what the user sees*" from "*what is stored*"
  - versions & relationships
    - *at the data store layer, all versions of each object are kept with their own relationships*
    - *data are not independent of each other ⇒ semantic relationships can denote various kinds of associations:*

      tree (structural relation like "father/child" or "container/content")

      use (semantic relation like "a program uses a library", eg #include)
  - computation of views
    - *a user has only a partial view of data contained in the store*
  - mapping of user actions
    - *user actions (on user view) have to be translated into basic actions (on the data store): create new versions, update relationships,. . .*

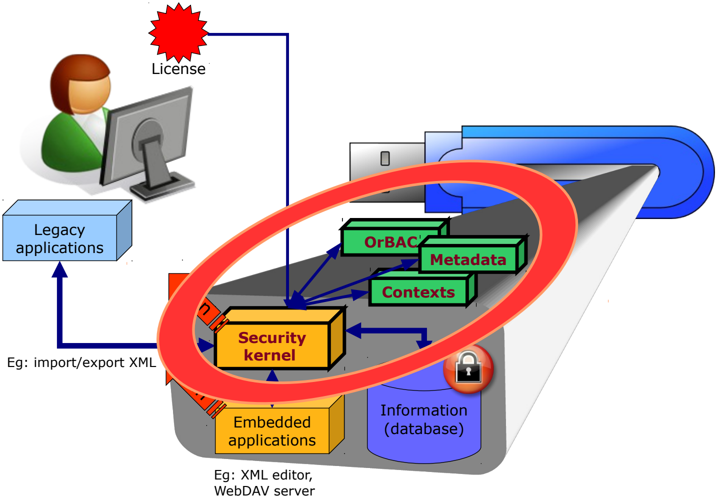
Autonomic Documents

Embedded database

- Benefits of this model
  - user actions have the intended effect on her/his view
  - system preserves the integrity of data (eg relationships between nodes)
- Embedding database within the intelligent document
  - nodes can be tagged with metadata
  - database is ciphered so that only the security kernel can access its content

Autonomic Documents

Security kernel & security modules



Autonomic Documents

Security kernel & security modules

- The **security kernel** is the core of our architecture
  - it is the document interface with the outside world
  - all the actions performed by the users to handle the document have to be done through the security kernel
- To enforce the security policy, the security kernel relies on various **security modules** dedicated to specific tasks
  - those responsible of **accepting or rejecting** user actions

    eg *access & usage control*
  - those collecting and attaching **metadata** to the actions

    eg *who performed this action, from which IP, at what time, with which application, in which context,...*
  - those **calculating new information** as actions go along

    eg *trustworthiness indicator, collaborative work management,...*

Autonomic Documents

Security kernel & security modules

- When the user requires the execution of an action, the security kernel performs control in two stages

  1. validate the action
     - the kernel requests each security module to validate the action
       → *some modules will add information to this action (eg metadata)*
       → *others will indeed accept/reject the action (eg access control)*
  2. process the action
     - basic operations implementing this action are then performed on the data warehouse
     - the security kernel broadcasts this action a second time to each security module so they can achieve the associated processing
       → *logging (eg access control, usage control)*
       → *adding metadata to nodes in the embedded database*
       → *computation of additional information (eg trustworthiness management, collaborative work management)*

Autonomic Documents

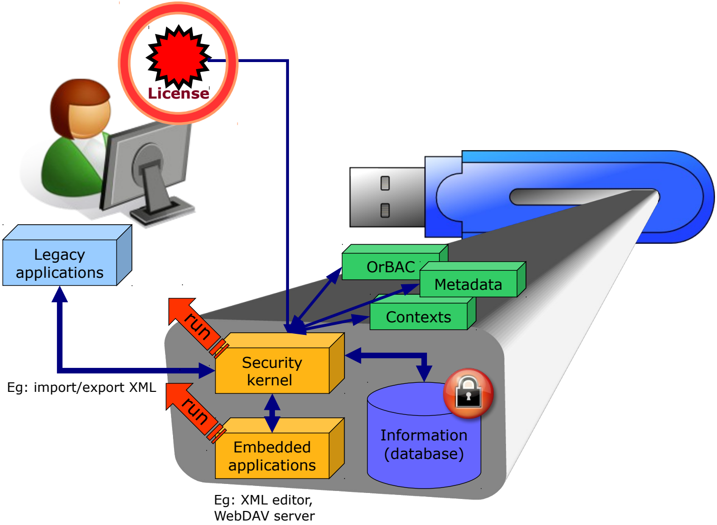
Security kernel & security modules

- Security modules we already developed

  1. access & usage control
     - we use the OrBAC model
       → *permissions, prohibitions, obligations*
       → *security rules can be dynamic, i.e. depending on the context*
  2. context management
     - we can control context activation in the OrBAC model
     - how to check conditions from the context definition ?
       → *direct access to the host system (eg a global clock)*
       → *metadata carried by the actions*
  3. metadata recording
     - put metadata on actions & nodes in the embedded database

Autonomic Documents

License contents



Autonomic Documents

License contents

- The license contains many information:
  - identity of the server that issued the license (the licensor)
  - data about the user to which the license is granted (the licensee)
  - all the information needed to configure the various security modules

    → for now, OrBAC security rules (with contexts)

    → which (and how) metadata should be collected ?

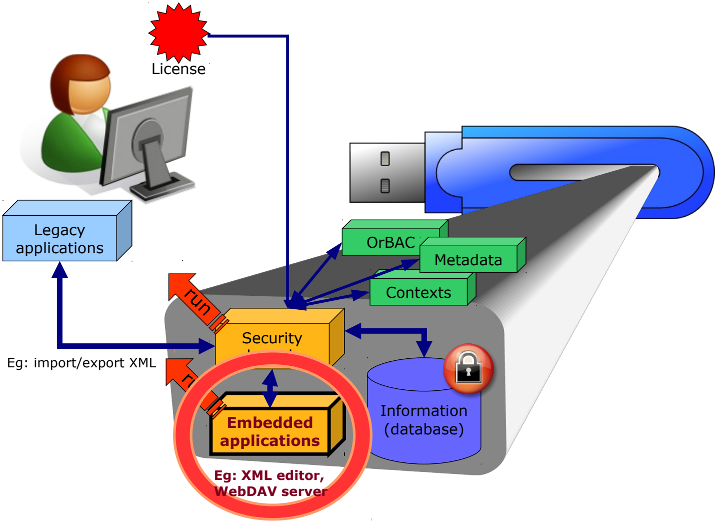
    → what triggers must be deployed to manage contexts ?

    → *(later) what information can be automatically computed ? (eg trustworthiness indicator)*

⇒ standards like XrML or ODRL do not suit our future needs

Autonomic Documents

Embedded applications & services



Autonomic Documents

Embedded applications & services

- How to interact with the document ?
  - **export & import** mechanisms (XML for example) to manipulate information through existing applications

    → *filters at the security kernel level to format information when exporting (checkout) and to interpret them when importing (checkin)*
  - **plugins** developed for existing applications

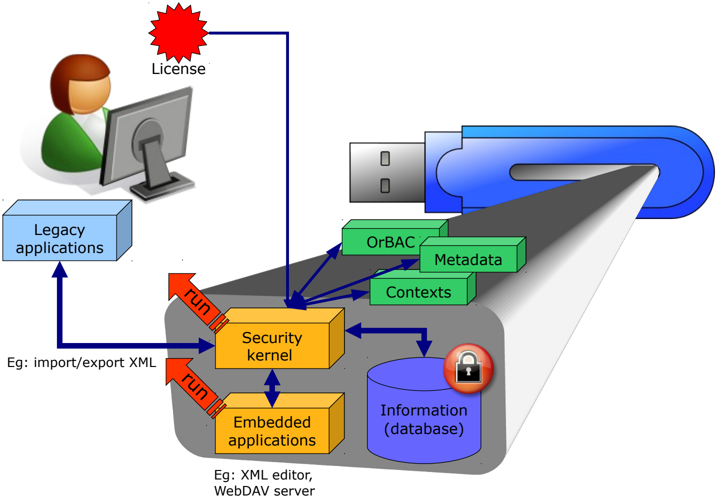
    → *the plugin can then talk directly with the security kernel to interact at the nodes and relationships level (finer granularity)*
  - use of **services and/or dedicated applications embedded** in the secure document

    eg *after starting the different security components, the document can automatically start running a local WebDAV server to present the information as a tree of files/directories*

    → *access to information can then be made from traditional applications through a WebDAV client*

Autonomic Documents

Summary



Platform Implementation

- Intelligent document ≡ decentralized IS

  ⇒ it must bring together on the same "support"

    - a **database** (contents of the document, metadata,. . .)
    - several **executables** (security kernel, security modules, embedded services & applications)

- Embedded database
    - use of our prototype of secure versioned repository (**SeVeRe**)
    - model extension: support for operations on groups of objects

  ⇒ *users can store structured documents like XML (where every node is represented by an object) and manipulate them via routines in the checkout/checkin style at the level of a whole document or as part of the document (and not node by node)*

Platform Implementation

- Security concerns
  - **Java** ⇒ document can run on various OS (MS Windows, Linux, Android,. . .)
  - **ciphering** to protect embedded database, license contents,. . .
- Actual implementation
  - an easy solution: a USB flash drive that represents the document and can be exchanged (physically) between users

    ⇒ **standard USB flash drives** with an autorun configuration to launch **Java** programs
  - intelligent document as a **single file** (JAR archive)

    → *more user friendly: 1 file in the cloud/on a USB flash drive, 1 email attachment,. . .*

    ⇒ workaround to "update" a JAR file ☹

⇒ Next step: develop a **cloud storage service**

Platform Implementation

- Platform tested in the **FLUOR** project[3]
  - *convergence du contrôle de* **FL***ux et d'***U***sage dans les* **OR***ganisations*
  - → collaborative work based on intelligent documents embedding a small information system built from our model
  - http://fluor.no-ip.fr/index.php

- Future work
  - **policy management**
    - *security policy update ⇒ license management (revocation list,. . .) to propagate new security rules*
  - **risk analysis**
    - eg *ISO/IEC 27005:2011 information security risk management*
    - → *decentralized IS: benefits, but also new vulnerabilities. . .*

[3] work supported by the French ministry for research under the ANR-SESUR 2008-2011 project FLUOR

Contribution

- **Self-Protecting Documents for Cloud Storage Security**
  - E-DRM architecture using autonomic documents
    - → users only need a drop point (eg cloud storage service)
      only for checkout/checkin/synchronize operations
      ⇝ **documents ensure their own security** *(data centric solution)*
    - → users can exchange docs without going through the server
      eg email attachment, USB flash drive
    - → documents can carry dedicated applications & services
      eg service to present document contents as a filesystem, business applications,...
  - enterprise context
    - structured & complex documents
    - working documents ⇒ users can update the contents
    - relations between the partners are well defined ⇒ advanced security policy definition

Future Work

- Perspectives
  - **legal issues & privacy concerns**
    - **which (and how) metadata can be collected ?**
    - **what information can be automatically computed ?**

    ⇒ the contents of the license gives the terms of use of the document that the user must agree
  - **risk management**
    - autonomic documents ⇒ distributed information system

    → advantages & disadvantages, new vulnerabilities,. . .

    → ISO/IEC 2700x **risk analysis**
  - programming issues
    - implement new security modules

      eg trustworthiness management, collaborative work management
    - policy management

      update security rules, revoke licenses,. . .

Manuel Munier, Vincent Lalanne, Magali Ricarde

Self-Protecting Documents for Cloud Storage Security

Thank you for your attention.

manuel.munier@univ-pau.fr

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

http://www.univ-pau.fr/

BackPlan™ Project Communication Control

http://www.backplan.fr/