Electronic Document Technology Standards and Signatures

E-Courts 2008, Las Vegas

Erik Wilde, UC Berkeley School of Information
December 9, 2008

Abstract

PDF, PDF/A, OOXML, OpenDocument. What is the alphabet soup? In recent years technologists have been attempting to make electronic documents more transportable across systems and displays as well as improving their usability. This session will explain these various document formats and how your court can use the technology to improve data capture, display, and information security.

About this Presentation

Outline

About Me

Outline

About Me

- Computer Science at Technical University of Berlin (TUB) (88-91)
- Ph.D. at ETH Zürich (92-97)
- Post-Doc at ICSI, Berkeley (97/98)
- Various activities back in Switzerland (98-06)
  - teaching at ETH Zürich and FHNW
  - working as independent consultant (training, courses, consulting)
  - research in various XML-related areas
- Professor at the School of Information (since Fall 2006)
  - Technical Director of the Information and Service Design (ISD) program

---

About ISD

Outline

Information and Service Design (ISD)

- Part of UC Berkeley's School of Information
- Connecting our students with real-world scenarios and projects
  - "Building Stuff That Actually Works"
  - getting involved in project management and associated challenges
  - understanding the real-world challenges of information modeling
- Focus on open information systems and open information access
  - "usability" and "accessibility" should become terms beyond the UI realm
- Example areas of ISD interest
  - e-Books beyond "iTunes for books": open formats, flexible reuse
  - open data for field researchers: sharing information as simply as possible
  - location on the Web: how to turn the Web into a location-aware system

---

Information-Intensive Applications

- Traditional enterprise IT solutions have limits
  - built for long life-cycles of deployed system architectures
  - built for integration of existing systems into a unified landscape
- Many enterprise IT solutions cannot keep up very well
  - by definition, they never completely fail
  - they dictate the shape and direction of information flows
- The Web is by far the biggest information system that ever existed
  - built around an astonishingly primitive data model
  - the simplicity is not a deficiency, it is a feature
  - everybody can cooperate as long as there is minimal agreement
  - the Web's architectural principle can be reused for enterprise IT

Project: Environmental Data

- Government agencies collect and manage a lot of environmental data
  - some of it is accessible in historical or current archives
  - some of it is permanently produced by sensors
- Large-scale data aggregation presents various challenges
  - implementation issues of sensor deployment and management
  - organization issues of classifying and grouping sensors
  - access issues of being able to access subsets of the available data
  - policy issues of sensible data and possible access restrictions
- Web architecture presents a proven path for large-scale systems
  - built on loose coupling and cooperation rather than integration
  - built on a different architecture than traditional enterprise IT

Project: Justice and the Criminal Record

- Criminal records are important for background checks
  - companies collect information and are re-sellers
  - there is no expiration date for this information
- Criminal record information changes in important ways
  - new entry: important for background check (false negative)
  - expunged entry: not critical for background check (false positive)
  - little business incentives for companies to properly delete entries
- Information accessibility can introduce new challenges
  - how to hold people accountable for providing outdated data
  - how to create incentives for properly updating data

Dream Project: Services, not Sites

- Government agencies should provide services, not sites
- Sites are hard to build and hard to maintain
  - often built with specific use cases in mind
  - technology evolves and sites must be maintained to keep up
- Sites get in the way of services
  - often service access is possible only through a site
- Services provide all the necessary information
  - exposing what the public has paid for
  - not spending public money for building interfaces
- Policy issues around service design and information usage
  - information licenses must be developed to avoid data rot
  - "eat your own dogfood" is a good start, but not sufficient (tastes differ)

Document Standards

Outline

REST

- The Web is built on *Representational State Transfer (REST)*
  - *resources* are the "units of interest" in any REST design
  - peers interact by exchanging *representations of resources*
  - interactions can only use a *small number of predefined verbs* (4 in HTTP)
  - state transitions are using *hypertext as the engine of application state*
- Documents often are the core part of a RESTful system architecture
  - the only absolute core part of REST is identification (URIs)
  - communications are often based on HTTP(-S)
  - representations often use HTML or some XML vocabulary
  - representations have primacy over functions or interactions

---

Document Exchange as Business Interactions

- Traditional enterprise IT is based on integration
  - model the complete system as one big distributed program
  - implement the system using some distributed programming environment
  - programming is based on the abstraction of building one big system
- Web architecture is based on cooperation
  - there is no overarching model, there are only local models
  - peers can interact by exchanging information about resources
  - cooperation is achieved by agreeing on representations of resources
  - there should be no assumptions about availability, links can always break
- Names for the debate: "REST vs. SOAP" or "REST vs. WS-*"
- This is an ongoing debate and will not go away anytime soon

---

Application-Independent Formats

Outline

History of Document Interchange

- Plain text and structured text
  - plain text only needs agreement on a common character set (e.g., ASCII or Unicode)
  - first data formats were *comma-delimited* or *tab-delimited* structures
  - *SGML (Standard Generalized Markup Language)* was the first open document format
  - XML (Extensible Markup Language) streamlined SGML to become usable on the Web
- *Document formats* vs. *data formats*
  - *data formats* represent database-like structures (e.g., UML or ER)
  - *document formats* represent narrative documents structures
  - many existing document collections use something in the middle
  - many applications need something in the middle

Structured Documents

- Most real-world data is *semi-structured* or *unstructured*
  - documents use titles, paragraphs, lists, and tables
  - documents do not mark up person names, place names, …
  - *Natural Language Processing (NLP)* tries to extract structures
- IT people want structured data, users often don't like forms
  - building good UIs is one of the core tasks for acceptance
  - badly designed data entry is sabotaged and produces garbage
  - provide feedback about the benefits of good data
- XML is a language for building languages, *but don't do it*
  - XML does not define any semantics (i.e., it only defines structures)
  - XML supports semi-structured data (supporting incremental refinement)
  - vocabularies define structure and semantics of *XML document types*
  - vocabularies may provide/use *modules*, thereby allowing flexible reuse

---

HTML

- HTML is the standard document format on the Web
- Microformats can be used to improve document semantics
  - earlier microformats were not based on a common syntax
  - *RDFa* (October 2008) provides a standardized syntax
- Why HTML often is not even considered as a document format
  - designed for logical structures, so rendering depends on clients
  - designed for continuous display, so paged content is not a natural fit
  - poor print support in regular browsers (problem of CSS and bad browser support)
- Why HTML should be considered as a document format
  - focus on content structures rather than rendering
  - easy to adapt to a wide variety of clients
  - printing problem can be solved with custom print processing

---

PDF

- *Portable Document Format (PDF)* evolved from a printer language
  - based on *PostScript*, a page description language for printers
  - removed some programming features, added a lot of file format features
- *Acrobat Reader* as a free product made PDF successful
  - the "give away the reader, charge for the writer" strategy
- PDF has become a complex and complicated specification
  - successful commercial products add features, which add data format complexity
  - backwards compatibility almost always means that no features will be removed
- Microsoft wants a piece of the pie with its *XML Paper Specification (XPS)*
- PDF 1.7 is the latest version (implemented by Acrobat 9.0)
  - published by ISO as *ISO 32000-1:2008* in November 2008

PDF Data

- PDF has evolved into a multimedia container format
  - support for various media types such as images, audio, and video
  - PDF forms allow interactive forms to be created and filled out
  - scripting can be used to further support interactive PDF
  - extensions allow 3D models to be embedded into PDF
- Text can also appear in a variety of ways
  - embedded images from scanning processes may only show text images
  - *Optical Character Recognition (OCR)* may result in poorly recognized characters
  - formatting software might include rendered characters (e.g., "ﬁ" vs. "fi")
  - formatted text might use non-embedded fonts
- rendering PDF is a challenging task
- searching PDF might be difficult or impossible

PDF Metadata

- *Metadata (data about data)* is essential for document management
  - it can be managed as an integral part of documents
  - it can be managed externally by having *metadata records*
- External metadata allows unified rules for metadata management
  - the same metadata can be captured for all resources
  - works for resource types with no metadata capabilities (e.g., books)
- Embedded metadata creates self-contained documents
  - packaging issues become easier
  - flexible embedded metadata formats support user-defined metadata models
- PDF supports various kinds of embedded metadata
  - earlier versions had a small set of hardcoded metadata fields
  - *Extensible Metadata Platform (XMP)* for extensible metadata (since PDF 1.4)

PDF/A

- ISO-standardized PDF profile for archiving PDF documents
  - focus on long-term archiving of PDF documents
  - color spaces must be specified (important for printing)
  - all fonts must be embedded
  - audio/video content and scripting are not allowed
- PDF/A is a good choice for archiving workflows
  - documents should be verified before accepting them as PDF/A
  - minimal amount of metadata must be embedded
- PDF/A-1b only focuses on the visual appearance of a document
  - scanned pages can be contained as images only
- PDF/A-1a also focuses on the content of a document
  - *tagged PDF* supports searching and repurposing of document contents

---

PDF/X

- ISO-standardized PDF profile for pre-print document exchange
  - focus on high fidelity rendering of PDF documents
  - color spaces must be specified (important for printing)
  - all fonts must be embedded
  - various boxes must be defined for specifying the print area
- PDF/X is not a good choice for non-production workflows
  - often very specific for one publishing workflow
  - no constraints that focus on document management properties

---

OpenDocument (ODF)

- Developed as the native format for *OpenOffice*
- Standardized by ISO as ISO/IEC 26300:2006
- Main starting point was the need for an open office format
  - Microsoft's Office products used undocumented file formats
  - document management should be based on documents, not products
- ODF's success forced Microsoft to open the Office file formats
  - in 2005, Massachusetts stated that open formats should be used for public data
  - in 2007, Massachusetts added OOXML to the list of approved formats
- Disadvantages of ODF
  - not as widely supported (but getting there)
  - currently no standardized digital signature format (ODF 1.2)

OOXML

- Microsoft started OOXML as a response to ODF's challenge
- OOXML was blessed by ECMA (XPS uses the same strategy)
  - ECMA is often used as a simple first step in standardization
  - ECMA-approved specs can be fast-tracked in ISO
  - Microsoft's tactics caused a lot of controversy among experts
- OOXML is a compressed package of various resources
  - the *Open Packaging Conventions (OPC)* create an archive of all resources
  - OOXML is a structured archive with conventions for its contents
- Disadvantages of OOXML
  - 6'500 pages of file format specification
  - many redundancies for historical reasons (e.g., three different table models)
  - the document XML format is not easy to process

Application-Specific Formats

Outline

Why Use XML?

- Because you want to share data
  - share it in a format which is widely used and easy to use
  - enable others to use it on various platforms with existing tools
- Because you want to share data cheaply
  - it is easier to use XML than to invent something new
  - it is even easier to use an existing XML schema than to invent a new one
- Because you want to share data openly
  - if you invent new formats, people must process them
  - avoid applying the "security through obscurity" principle inadvertently
  - application-specific processing should be deferred to higher layers

---

Is XML Self-Describing?

- XML is often said to be "self-describing"
  - many people think this is the same as "self-explanatory"
  - the catch is what exactly it is you refer to by "describing"
- Database data cannot live without a database
  - database data is simply content, the structure is provided by a DBMS
  - XML documents have their structure encoded within them
  - compared to database data, XML in fact is "self-describing"
- What is the gap between "self-describing" and "self-explanatory"?
  - it is impossible to find out how the document could be modified
  - there are no semantics associated with neither structure nor content
  - so "self-describing" means, you can guess a lot, but you maybe wrong

---

XML is Syntax

XML documents can use a wide array of characters. They are defined by Unicode, which currently (Version 5.0) defines more than 100'000 characters (#100'000 added in 2005).

```
<?xml version="1.0" encoding="UTF-8"?>
<JAPANESE>
 <TITLE>専門家リスト </TITLE>
 <ITEM>アシム・アブドゥラー氏（コマースネット事務局長）</ITEM>
 <ITEM>アラン・A・メッコラー氏（メッコラーメディア会長兼CEO）</ITEM>
 <ITEM>アラン・サルディッチ氏（メトリコムディレクター）</ITEM>
 <ITEM>ウィスター・ウォルコット氏（パイロットネットワーク・サービシズ副社長）</ITEM>
 <ITEM>・エリック・リンゲワルド氏（ビー・インク副社長）</ITEM>
 <ITEM>ジェームス・L・バークスデール氏（ネットスケープ・コミュニケーションズ社長）</ITEM>
</JAPANESE>japanese1.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<文書 改訂日付="1999年3月1日">
 <題>サンプル</題>
 <段落>これはサンプル文書です。</段落>
 <!-- コメント -->
 <段落>会社名</段落>
 <図面 図面実体名="サンプル" />
</文書>japanese2.xml
```

XML is Character-Based

- XML is *not* a binary format, it is
  - "binary structures" cannot (or rather should not) be described using XML
- Multimedia formats often are binary
  - image formats such as GIF, JPEG, and PNG
  - audio formats such as MP3 and AAC
  - video formats such as MPEG4 and H.264
- But: multimedia also uses many XML formats
  - vector graphics formats such as *Scalable Vector Graphics (SVG)*
  - *Synchronized Multimedia Integration Language (SMIL)* for describing presentations

---

XML is a Syntax for Trees

- Not all data is easily represented by trees
  - overlapping markup (multiple "views" of the same content)
  - graph-like structures which are less constrained than trees
- What is it that you have in your tree?
  - XML encodes a structure purely on the syntactic level
  - what the structures mean is in no way described by XML
  - XML structures must be accompanied by semantic descriptions

---

XML Usages

- XML can be used in different ways
  - people should be able to use your XML directly using standard tools
  - if they *absolutely need* a set of special tools, something is wrong
- XML is hip, so everybody wants to use it
  - many things have been created ad-hoc and without much planning
  - if you start something which is XML-based, use XML responsibly
  - if you have to use some "bad XML", complain about it
- Finding the balance can be hard
  - XML is great for prototyping and experiments
  - once you decide to redesign your XML, it may be too late
  - *XML documents* may be short-lived, *XML schemas* are definitely not

---

Document Security

Outline

Identity

- Identity is a central hub of any IT security
    - *identity* is established by associating digital identities with real entities
    - identities can be *grouped* and they can have *assigned roles*
    - *authentication* is the process of verifying an identity claim
    - *access control* can be based on *identities*, *groups*, or *roles*
    - *authorization* is the process of providing access to a controlled resource
- *Authentication* is one of the tough problems of IT security
    - *usernames* and *passwords* are commonly used
    - additional cues (smartcards, images, biometrics) may be used
    - *security questions* often are a bad idea for establishing identity
- Almost all IT security revolves around some "digital identity"
    - users find many ways around inconvenient security implementations

---

One-Way Function

Outline

Essence of Data

- Hashes (or *message digests*) are a well-known principle in computer science
    - fast to compute (the goal is to make data handling more efficient)
    - few collisions (there are always collisions because of the smaller size)
    - *checksums* and *Cyclic Redundancy Check (CRC)* are popular hashes
- One-way functions are cryptographically safe hashes
    - not just for detecting errors, but also for preventing tampering
    - often referred to as *cryptographic hash* or *digital fingerprint*
- One-way functions must satisfy some additional criteria
    - it must be very hard to find an input producing a given output
    - it must be very hard to find two inputs producing the same output ("collision")

Reducing Data

Digital Signature

Outline

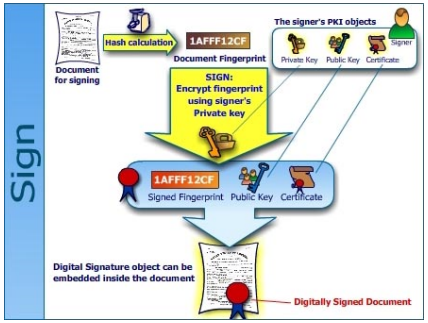
Encrypted Fingerprints

- Hashes are used to check data integrity
- One-Way Functions are used to check data integrity securely
    - it is not possible to reverse engineer data for a given hash
- Signed hashes can be used to ensure data authenticity
    - if the hash sum is signed, it cannot be changed
    - if the data is changed, its hash will not match the signed hash
- Digital signatures work as long as the hash can be securely signed
    - there must be a trusted Identity for verifying the hash signature

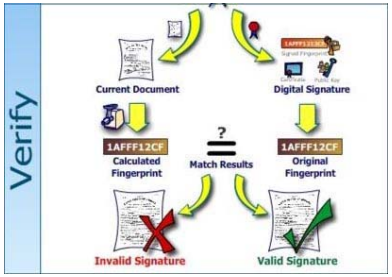
Certificate

- Certificates are digital signatures issued by a trusted party
    - most digital signatures are created with certified public keys
    - this means the digital signature is created based on a digitally signed key
- Who can you trust on the Web?
    - trust can only start to grow based on initial trust in something
    - many systems come with pre-installed trust (*root certificates*)
    - certificates from other issuers will cause browsers to complain
- Certificates (like domain names) are a very easy way to make money
    - in theory there are different levels of certificates with different levels of identity checking
    - in practice most sites choose the cheapest one that does not produce an error message

Creating a Digital Signature



Verifying a Digital Signature



Conclusions

- IT architecture has two major design phases
  1. modeling of information structures and business processes
  2. exposing required functionality through an interface/implementation
- Documents formats are essential for information models
  - build your own model and use existing formats as a guidance
  - provide implementations of the model by mapping it to (existing) formats
- Information models are the very core of many activities
  - "Getting the Job Done" requires good understanding of the job
  - short term hacks are sufficient for activities with a short term horizon
  - thorough analysis and understanding is required for longevity