

“Trustworthy” Electronically Signed XML & Data Confidentiality in Electronic Processes

An Approach for Compliance with Federal Guidelines
Presentation to CIO XML Working Group, March 19, 2003

Matthew McKennirey, 703 734 3000 ext 120
mmckennirey@conclusive.com

Today's presentation

Part 1. Meet GPEA requirements

Use XML signature to comply with NARA and DOJ guidelines for creating and preserving “trustworthy” digitally signed documents:

- **Records Management Guidance for Agencies Implementing Electronic Signature Technologies, National Archives and Records Administration, October 18, 2000**
- **Legal Considerations in Designing and Implementing Electronic Processes A Guide for Federal Agencies, Dept. of Justice, November 2000**

Part 2. Comply with Federal legislation on data confidentiality

Use XML encryption for data confidentiality and to enforce role-based access in electronic workflow.

- **Computer Matching and Privacy Protection Amendments of 1990, Privacy Act of 1974**
- **Health Insurance Portability and Accountability Act of 1996 ("HIPAA")**
- **Gramm-Leach-Bliley Act of 1999, Privacy of Consumer Financial Information Regulation SEC 17 Cfr Part 248**
- **Guidance on Inter-Agency Sharing of Personal Data - Protecting Personal Privacy, OMB December 20, 2000**

For those joining by tele-conference, or for further research, resources for this presentation are available as follows:

■ HTML version of Presentation

- ◆ <http://www.conclusive.com/download/XMLWG19Mar03.htm>

■ Presentation (PDF format)

- ◆ <http://www.conclusive.com/download/XMLWG19Mar03.pdf>

OR at

- ◆ <http://xml.gov/agenda/20030319.htm>

■ XML documents

- ◆ XML document for slide extracts (generic XML structure)

- ◆ <http://www.conclusive.com/download/XMLWG19Mar03XML.xml>

- ◆ XML from demo application, retrieved from Audit Vault

- ◆ http://www.conclusive.com/download/XMLWG19Mar03_epay.xml

■ PDF output from demo application

- ◆ http://www.conclusive.com/download/XMLWG19Mar03_epay.pdf

Part 1. Meet GPEA requirements

- **Paperwork elimination requires legally effective electronic signature.**
- **Guidance has been provided by**
 - ◆ **National Archives (NARA) in respect to what is “trustworthy” and what they believe is necessary to preserve electronically signed documents**
 - ◆ **Department of Justice (DOJ) in respect to the legal considerations involved in electronic processes**
- **This part of the presentation will lay out how NARA and DOJ guidance can be applied to XML documents using W3C XML signature standards and Public Key Infrastructure (PKI) technology.**

The presentation today will create an XML document that is electronically signed and preserve it applying NARA guidelines for both processes.

- Document will be XML that conforms to W3C XML Schema
- Document is created and edited in the context of a web application (example is filling in an XML electronic form)
- Document is electronically signed (W3C XML digital signature with PKI certificate) and includes confidential data which is encrypted (W3C XML encryption)

NARA definition of the characteristics of electronically signed trustworthy records (also ISO 15489):

■ Reliability:

- ◆ Content can be trusted as a full and accurate representation of the transaction(s)

■ Authenticity:

- ◆ Can be proven to be what it purports to be and to have been created or sent by the person who purports to have created and sent it.

■ Integrity:

- ◆ The integrity of a record refers to it being complete and unaltered.

■ Usability:

- ◆ Can be located, retrieved, presented, and interpreted.

To preserve “trustworthy” documents NARA guidelines are: “It is necessary to preserve its [the document’s] content, context, and sometimes its structure.”

Trustworthy Record

Content (Signed)

```
<payment>
  <status>New</status>
  <serial>7</serial>
  <request_fname>Barb</request_fname>
  <request_lname>Lance</request_lname>
  <date>2003-02-20 13:29:50</date>
  <total_amt</total_amt>
  <comment>"Comment"</comment>
  <myfile>
    <name>Payments_CT.xls</name>
    <size>27648</size>
    <date>1/21/03 11:19:06 AM EST</date>
  </myfile>
  <ownerdn>Barb Lance</ownerdn>
  <managerdn>E-PAY-PLI/Treasury</managerdn>
  <hrdn/>
  <test/>
  <tlauditrecord/>
</payment>
```

Context

- What application?
- Who were the parties?
- What role did user have?
- What credentials were presented?
- What authority was cited?
- When did this occur?
- What jurisdiction had authority?

Structure “Presentation”

Payment Issue			
Serial No.	9	Date 2003-02-26 14:04:21 Status Approved	
Request			
First Name	Barbie	Last Name L Account	
Item		Amount	
		Total	
Comment			
Attachment	Expenses.xls		
Attachment2			
Approval			
First Name	B	Last Name Merlin	
Title	Magician	Date 2003-02-26 14:21:38	
Issue Approval			
First Name		Last Name	Date
Center		ID	Auth Code
HR			
Comment			

To conform to guidelines, the high level structure of our XML document will consist of these 3 elements: **Context**, **Content** (which is signed), and **Structure**.

[From here on, to avoid confusion with the structure of the XML, we will refer to NARA's definition of Structure as the "Presentation" of the XML– that which defines the user experience of the XML.]

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Structure of NARA compliant signed XML document-->
<Transaction>
Context  [ <Context/> <!-- identifiers to establish validity -->
          [ <Content> <!-- self-explanatory -->
            [ <Signature/> <!-- Signature(s)-->
              [ <EncryptedData/> <!-- Encrypted data -->
                [ </Content>
                  [ <Presentation/> <!-- the user experience of the XML -->
                    [ </Transaction>
```


Let's start with Context. This is where we record session, authorization, and credential information that describes the context in which the XML was created or modified. This <Context> will be "extensible" to add any "identifiers" required.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Example of type of data to be captured in Context -->
<Transaction>
  <Context>
    <userUniqueID/> <!-- User Identity -->
    <timestamp/> <!-- Time Transaction started -->
    <sessionID/> <!-- HTTP session ID -->
    <userRole/> <!-- Role assigned to user -->
    <certIssuer/> <!-- Certificate Authority -->
    <clientTokenType/> <!-- Type of token used -->
    <application/> <!-- Name of application -->
    <serverID/> <!-- Identity of server -->
    <authName/> <!-- Name of the authority -->
    <organization/> <!-- Name of the organization -->
  </Context>
</Transaction>
```

Context

Because Context is critical, we will sign it, both by the application and by the user. To keep things straight we will structure our XML with a Header and Body and put Context into the Header.

```

<!-- Example of type of data to be captured in Context -->
<Transaction>
  <Header>
    <Context>
      <Header>
        <Signature/> <!-- Signature of application -->
      </Header>
      <Body/> <!-- Various identifiers -->
    </Context>
    <Signature/> <!-- Switches between user and app -->
  </Header>
  <Body>
    <Content/>
    <Presentation/>
  </Body>
</Transaction>

```

Context

Content

Presentation

The next high level element in our XML is Content. A design objective is to make any XML document “trustworthy”, regardless of its own internal structure and content. In today’s example we use a <SecureForm> with signature and encrypted data elements. Any XML document could be included here.

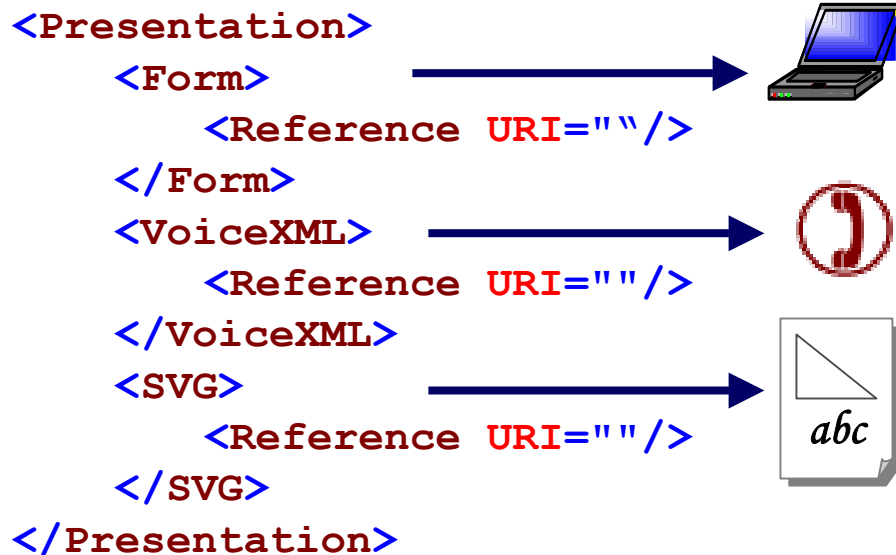
```

    <Transaction>
      <Header>
Context  [ <Context/> <!-- Context information -->
          </Header>
          <Body>
Content  [ <BodyContent>
          [ <SecureForm>
          [ <Data/> <!-- The Content -->
          [ <Signature/> <!-- Signature info -->
          [ <EncryptedData/> <!-- Encrypted data -->
Presentation [ </SecureForm>
              <Presentation/> <!-- User experience -->
            </BodyContent>
          </Body>
        </Transaction>
  
```

NARA requires that signature be part of the preserved Content. Considerations in respect to the application of signature include:

- Ensure the appropriate content is actually signed**
- Ensure the means of signature conforms to a recognized standard**
- Ensure the signature is valid (in PKI:verifies, not revoked)**
- Ensure the signatory knows what they are signing. Be able to demonstrate the signatory was made specifically aware of what they were signing**
- Capture the reason (why) for the signature. Does the signatory mean to indicate they (a) saw the document? (b) attest to some part of the document ?(c) are bound by the document?, etc.**
- Only sign what the signatory intends. If the signatory does not mean to attest to the validity of other users' statements, the latter should not be signed by them.**
- Ensure a signatory knows what others have signed. Be able to demonstrate the signatory specifically knew what others signed.**

The final high level element of our XML is the Presentation of the document (“Structure” in NARA terms). Preserving the Presentation is critical to the interpretation of the signed XML. The same XML Content may be presented to, and experienced by the user, in many different ways in the same application.



- An interactive form in a browser (XForm or similar). The visual presentation (Prompts, Drop downs etc) is important to know what the user intended.
- A voice response system (VoiceXML). Voice prompts and choices are critical to knowing what the user intended.
- A graphic or other printed document (SVG / XSL / XFO)
- Others means yet to be invented (3D Hologram?)

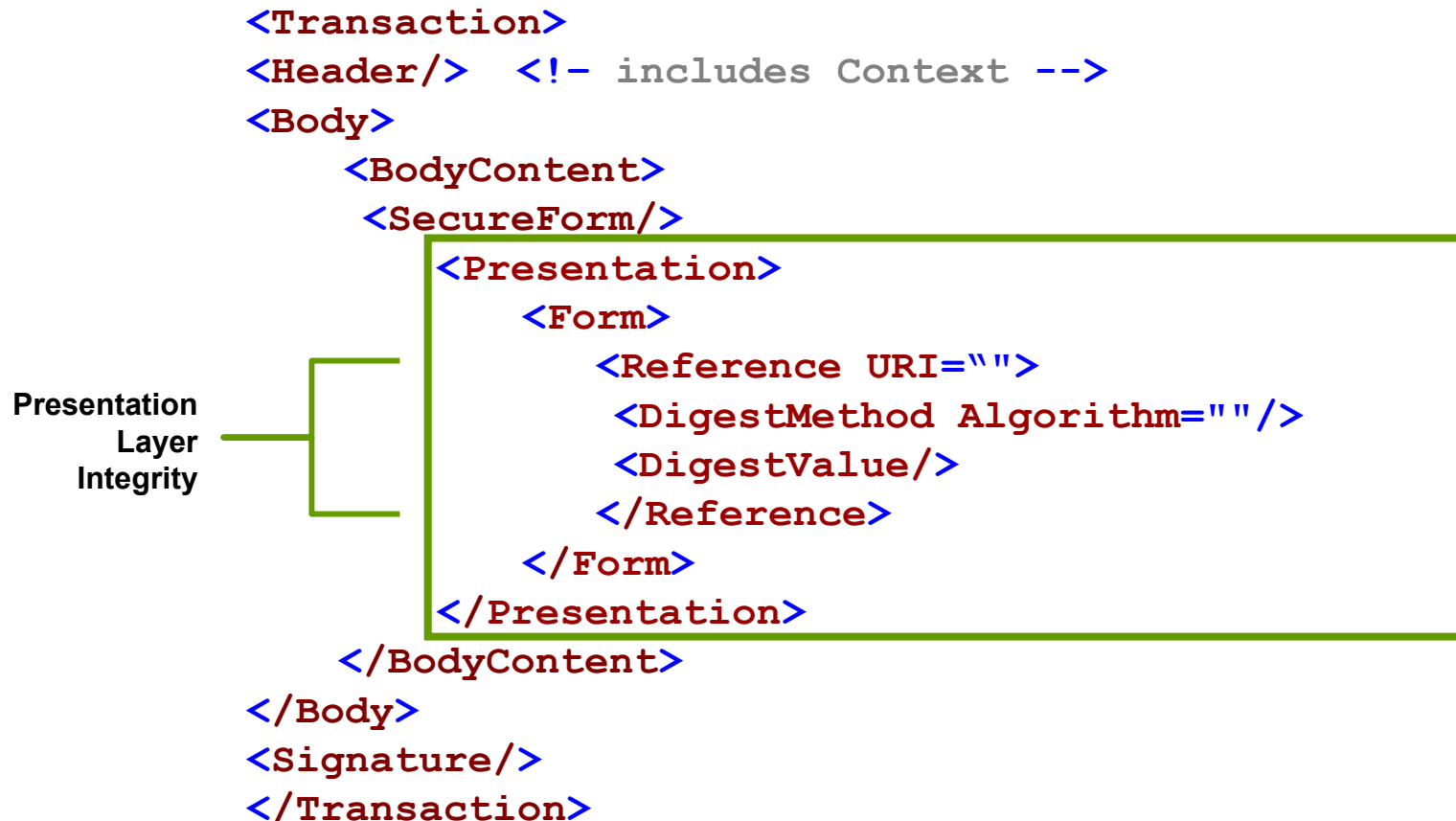
To facilitate integration of the signed XML with the rest of the infrastructure and with other XML applications the <Presentation> is structurally separate from the <Content> in the XML.

- Supports simultaneous multiple forms of presentation from the same XML Content
- Supports integration with other XML applications
- Minimizes storage / bandwidth requirements – store / download 1 copy of the “presentation”, not “n” copies
 - ◆ Forms: the simple form in today’s demo has 29 Kb of content (after the 1st signature) and 146 Kb of presentation. Difference in storage / download requirements for this simple form is a 6:1 ratio
 - ◆ Forms: don’t force users to download formats they already have in order to complete another instance of the same form
 - ◆ Forms: supports automatic, instant, deployment of revisions to forms: change version number of form in XML document template on the server and all users will use the new version the next time they connect to the application
 - ◆ Forms: separation of content and presentation may be critical for the practical preservation of forms created annually in tens of thousands of copies (storage and processing requirement)

The Presentation is treated as a resource and identified by a “URI”. To ensure its integrity the hash of the presentation resource is signed within the trustworthy XML document such that the integrity and authenticity of the presentation layer can be verified.

```
<Transaction>
<Header/>  <!-- includes Context -->
<Body>
  <BodyContent>
    <SecureForm/>
    <Presentation>
      <Form>
        <Reference URI="">
          <DigestMethod Algorithm="" />
          <DigestValue/>
        </Reference>
      </Form>
    </Presentation>
  </BodyContent>
</Body>
<Signature/>
</Transaction>
```

Presentation Layer Integrity



We now have all the elements NARA identifies for preservation to create a trustworthy document: Context, Content (Signed), and the Presentation layer.

The next step is to ensure the preservation mechanism meets guidelines as well.

NARA recognizes two approaches to preservation:

- **(1) Maintain documentation of the proof of the validity of the document at the time the record was signed – retain contextual information and documentation of process**

OR

- **(2) Maintain the ability to re-validate the digital signature –retain ability to perform digital signature verification**

Our approach is “best of both”, conform to the latter (re-validate signature) and include the contextual elements of the former.

- **Ensure the document’s integrity over time (internal integrity)**
- **Ensure the integrity of the sequence of events that created the document (sequential integrity)**
- **Preserve the integrity of the user experience at the time (presentation integrity)**
- **Ensure the usability of the stored document: the mechanism of preservation must be able to reproduce the document as the user experienced it**
- **Avoid proprietary or non-standard technologies (hardware, software, data formats) to mitigate issues of obsolescence**

To preserve the document, we store the XML in a SQL compliant database (TrustLogic Audit Vault) that is cryptographically “enhanced” to preserve records.

Metadata facilitates document search and exploitation of the document set . Other metadata (not shown) is held in a separate table within the database.

XML document, including signature and encrypted data

Internal and sequential integrity: each record is hashed and each record holds its hash and a cumulative hash with the previous record

Metadata

XML Document

Record Integrity

REC	PROCESS	EVENT_TIME	EVENT_DETAILS	EVENT_DATA	DIGEST	CUMULATIVE_DIGEST
1	11 Approve	26 Feb 2003	Payment Approved	<TLTransaction><Header>	Ó™Ç&4f€ÇÉ<.oymZ	—Ö"□□□□%4Ä□éÖ['-□i□i*
2	10 Logon	26 Feb 2003	User authorized	<TLContext><Header><Si	βÉ@€-<b□□Cè^a#u>]□~W□□'□i,x-□'~□'YÖ3
3	9 Logon	26 Feb 2003	Certificate Validated	<TLTransaction><Header>	τtèuY„N†™àX4□□y□	éMrg=,"x•b...¶□Eä□×□c Y
4	8 Logon	26 Feb 2003	CA PASSED trust	<TLTransaction><Header>	<âp□□iF*□™W@oc>	4ñ½Tu□□É□Ä□□(€□[*
5	7 View	26 Feb 2003	Payment being	<TLTransaction><Header>	YV¶i□TÖbj□δδA□□	úâ□LKÄ†C□□□i7^+¥/E%]
6	6 Request	26 Feb 2003	New payment	<TLTransaction><Header>	a¥f¼U□□□ckÉÜ=□	4□□□□□?k□□C_)SNYê□+□
7	5 Logon	26 Feb 2003	User authorized	<TLContext><Header><Si	ý3_öý%4sAlxÜ□□CÄy8	H**^2_Ü¥§iP)g•L`yÄDÜß
8	4 Logon	26 Feb 2003	Certificate Validated	<TLTransaction><Header>	žür@9w@%□fo>GÜöA	IA8□s"w•YŸÆ×□S`b 9†

This is a partial view of the EventDetails table from TLAudit, the “protected” audit database of TrustLogic.

[Interface to table shown here is MySQL Control Center]

Today's demo shows how this all works in practice.

- **Workflow: (1) payment request, (2) approval, (3) issuance**
- **Form is XML, each user signs their data, confidential data is encrypted (more on that later)**
- **“Presentation” alternatives include:**
 - ◆ (1) Interactive form: XSL + JavaScript using a commercial browser as interface
 - ◆ (2) PDF record (at user's option) : using XSL + XFO
- **“Preservation” mechanism is TrustLogic Audit Vault**
 - ◆ Decision on what is preserved is determined by application
- **PKI based cross domain authentication**
 - ◆ Users have a variety of tokens and certificates from different trust domains
 - ◆ Different Certificate Authorities each with their own certificate profiles, policies, and revocation technologies

The demo will step through two processes in the workflow: create the XML then modify the XML by another user, each user signing data they create and encrypting specific elements for specific roles.

Process 1: Create XML Document

■ Client side operations

- ◆ Logon & download form
- ◆ Complete form (payment request)
- ◆ Sign & Save for off-line editing
- ◆ Logoff & close browser
- ◆ Using local copy retrieve form, edit, save, print PDF
- ◆ Logon
- ◆ Upload saved form

■ Server side views

- ◆ View sequence of events in Audit

Process 2: Update XML Document

■ Client side operations

- ◆ Logon (2nd role in payment sequence)
- ◆ Retrieve payment request
 - ◆ Verify existing signatures
- ◆ Complete approval and submit
- ◆ Logoff & close browser

■ Server side views

- ◆ View updated sequence of events in Audit
- ◆ Retrieve preserved form from Audit
 - ◆ verify signature
 - ◆ view as XML
 - ◆ view as Form

The <Presentation> in today's demo uses XSL and JavaScript to interact with the XML through a commercial browser. This approach does not impose any constraints on the web application developer in terms of user interface and application functionality.

User Name and Authorized Role (Demo application design element)

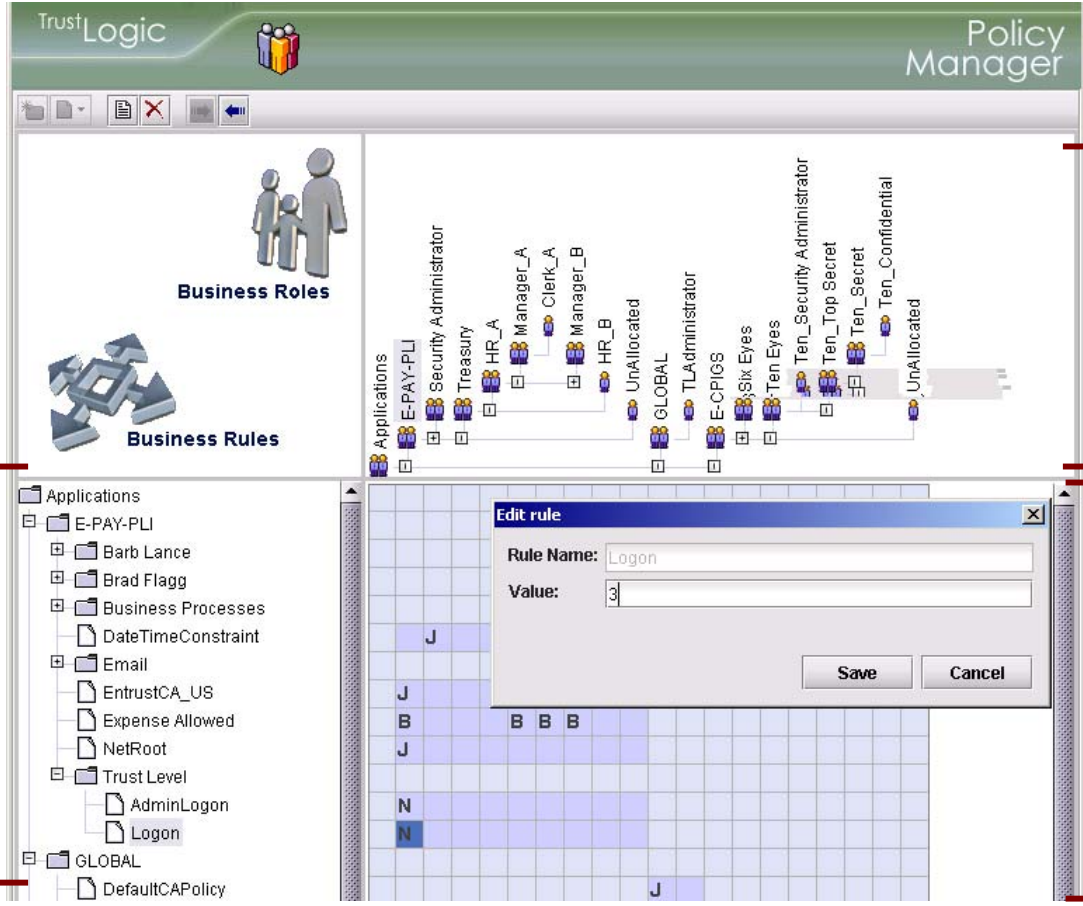
The TrustLogic infrastructure preserves the presentation layer but plays no role in the design of the user interface – that is an application responsibility.

XSL / JavaScript Presentation of XML

Authentication & Authorization Info (Demo application design element)

When the user logs on, or as required by the application, policy rules are applied based on the user's role. These are defined and managed through the Policy Manager interface.

Policy rules may include topics such as what assurance levels are needed to permit a user to logon to the application or to execute a process, etc.



Hierarchical Role Structure for application of rules – matches Role structure for cryptographic access control to data

Policy rule names

Role name / Rule intersects where rules are defined: can be (J) Java, (B) Boolean, (N) Number or (S) String

When the user submits the form, a signature confirmation dialog asks the user to confirm precisely (a) what they are signing, (b) what identity they are using, and (c) for what reason they are signing. They may save a copy locally and encrypt that copy.

Signature confirmation is a function of the TrustLogic Agent software – not the web application and is a consistent interface for the user across all web applications. It will always present the data to be signed and get user confirmation. The web application cannot get anything signed without the user being consciously aware of using their “signature”

The screenshot shows a web browser window displaying the 'E-Pay(PLI) Demonstration Site' with a 'Secure Electronic Payment System' header. The main content area is titled 'Payment Request' and contains a form with the following fields:

- Serial No.: 12
- Request: First Name: Barbara, Item: Capital Items, Expenses Items: n/a, Comment: Justification provide
- Attachment1: C:\Conclusive_Dem
- Attachment2: (empty)
- Approval: First Name: (empty), Title: (empty)
- Issue Approval: First Name: (empty)

Overlaid on the right side of the browser window is a 'TrustLogic Signature Confirmation' dialog box. It contains the following information:

- TrustLogic Agent logo
- Message: 'Please confirm you wish to sign the following data.'
- Data to be signed table:

Data name	Value
Comment	Justification provided in attached spreadsheet
Record 1	
Record 2	
Record 3	
Data na...	Value
Item	Capital Items
Amount	678
Record 1	
Record 2	
Data na...	Value
Attachm...	0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAAPPqA...
- You are signing as:
 - Your Name: cn=Barb Lance,ou=Purchasing,o=Lance Cc
 - Issuer name: o=Entrust,c=us
- Reason: An Authorized User
- Buttons: Save, Encrypt, Sign, Cancel

Data to be signed

Identity being used

Reason for Signature

When off line the user can verify signatures on local copies, continue to edit the XML in a browser, or generate a PDF file from the XML. (Create PDF shown)

The PDF is created using XSL / XFO. The PDF format is text searchable.

When the XML is edited and saved off-line all the cryptographic security (signature and encryption) continues to be applied.

The screenshot shows the TrustLogic Agent interface with an Adobe Acrobat window open. The Acrobat window displays a 'Payment Request' form with the following data:

Serial No.	Date	Status
8	2009-02-26 00:33:24	New

Request	
First Name	Barbara
Last Name	Lance
Account	56789
Item	Capital Items
Amount	56
	Expenses
	10
	na
	0
Total	66

Comment: Attached spreadsheet for justification

Attachment1: Expenses.xls

Attachment2: Tysons_Area.bmp

Approval	
First Name	Last Name
Title	Date

Issue Approval	
First Name	Last Name
Center	ID
HR Comment	Date
	Auth Code

To keep users informed of the authenticity of data, any accessible signatures on XML presented to them are displayed. The verification, for as many signatures as are found, includes (a) what was signed (b) who signed (c) what reason did they give.

Signature verification, like confirmation, is a function of the TrustLogic Agent software – not the web application and is a consistent interface for the user across all web applications.

Within the verification dialog the user can extract file attachments and save them locally in their original format.

The screenshot shows the TrustLogic Signature Verification interface. On the left is a navigation menu with options like Home, Register, Payments, Approve, Issue, View, Upload, Role Logon, Admin Logon, and Log off. The main area shows a 'Payment App' form with fields for Serial No., Request (First Name, Item), Comment, Attachment1, Attachment2, and Approval (First Name, Title). The 'TrustLogic Signature Verification' dialog is open, displaying the following information:

- TrustLogic found 1 signature(s).
- Signature 1 (indicated by a red arrow pointing to the 'Number of Signatures' label).
- Signed Data: 1 attachment(s) (indicated by a red arrow pointing to the 'Attachment' label).
- Signed Data table:

Data na...	Value
Comment	Justification provided in attached spreadsheet
- Record 1 table:

Data na...	Value
Item	Capital Items
Amount	678
- Record 2 table:

Data na...	Value
Attachm...	[-Base64Encoded-]
FileName	Expenses.xls
- Signature Information:

Signed by:	cn=Barb Lance,ou=Purchasing,o=Lance
Signed at:	Fri Feb 28 14:19:48 EST 2003
Reason:	An Authorized User

Red arrows on the right side of the dialog point to the following labels:

- Number of Signatures
- Attachment
- Signed Data
- Signatory
- Reason for Signature

The secure XML can be retrieved from the Audit Vault, and presented in the same manner as the user experienced the XML. Role based cryptographic data security always applies.

The saved form is presented here to the System Administrator whose role does not allow data access to encrypted elements on the form. A user with role access would "see" all the data.

The screenshot displays the TrustLogic Audit Viewer interface. At the top, there is a search icon and the text 'TrustLogic' and 'Audit Viewer'. Below this is a toolbar with various navigation icons. The main area is divided into two sections:

- Event Log Table:** A table with columns: Server, No., Event time, Application, Process ID, Event details, Event ID, User name, and Role. It contains several rows of event data, including logon events for 'E-PAY-PLI'.
- Payment Issue Form:** A web form titled 'Payment Issue' with fields for Serial No. (9), Date (2003-02-26 14:04:21), and Status (Approved). Below this is a 'Request' section with fields for First Name (Barbie), Last Name (L), and Account. There is also a table for items with columns 'Item' and 'Amount', and a 'Total' field. A 'Comment' field and an 'Attachment1' field (Expenses.xls) are also visible.

On the right side of the form, there is a context menu with the following options:

- View as audit record...
- View as XML...
- View as Form...
- View Signatures...

Red lines on the left side of the form indicate that certain data elements are saved in clear, while others are saved encrypted for a specific role(s).

Data elements saved in clear

Data elements saved encrypted for a specific role(s)

That's the end of Part 1 of our presentation: creating signed XML documents and preserving them in accordance with NARA and DOJ guidelines.

To sum up Part 1:

- **Our XML document captures:**
 - ◆ **Context**
 - ◆ **Content**
 - ◆ **Presentation**

- **The mechanism of preservation maintains the internal integrity of the XML document and the sequential integrity of its creation and modification (including Context and Presentation)**

- **The interface to preserved XML documents provides for:**
 - ◆ **Re-validation of signature**
 - ◆ **Retrieval as XML**
 - ◆ **Retrieval as Presented to user**

Part 2. Comply with legislation on data confidentiality

- **Requirements for data confidentiality have been established in a range of Federal (and State) legislation and regulation**
 - ◆ **Computer Matching and Privacy Protection Amendments of 1990, Privacy Act of 1974**
 - ◆ **Health Insurance Portability and Accountability Act of 1996 ("HIPAA")**
 - ◆ **Gramm-Leach-Bliley Act of 1999, Privacy of Consumer Financial Information Regulation SEC 17 Cfr Part 248**
 - ◆ **Guidance on Inter-Agency Sharing of Personal Data - Protecting Personal Privacy, OMB December 20, 2000**

- **This part of the presentation will lay out how data confidentiality can be ensured in electronic processes where the data structure is XML by using W3C XML encryption and Public Key Infrastructure (PKI) technology.**

W3C XML encryption defines a “unique” encryption event. A single event in which something is encrypted and the result stored in an XML structure.

Algorithm	{	<?xml version="1.0" encoding="UTF-8"?">
		<EncryptedData Id="1045766063567" xmlns="http://www.w3.org/2001/04/xmlenc#">
		<EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc">
		<Parameter Name="AlgorithmMode" Value="http://www.conclusive.com/2000/02/xml-crypto#mode-none"/>
		</EncryptionMethod>
		<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
		<KeyName>119509261</KeyName>
		<KeyValue>
		<RSAKeyValue>
		<Modulus>AKjxh9XnlDDT3FTXkON5qsFJk379e2vVAmq2Hgp6q9irDG/M8uD/rj</Modulus>
		<Exponent>AQAB</Exponent>
		</RSAKeyValue>
		</KeyValue>
		<EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
		<EncryptionMethod Algorithm="http://www.conclusive.com/2000/02/xml-crypto#rsa">
		<Parameter Name="AlgorithmMode" Value="http://www.conclusive.com/2000/02/xml-crypto#mode-none"/>
		</EncryptionMethod>
		<CipherData>fot8cr1bPTVaWlv9m/gD3IRfglyQW</CipherData>
		</EncryptedKey>
		</KeyInfo>
		<CipherData>
		<CipherReference URI="">
		<Transforms>
		<Transform Algorithm="http://www.conclusive.com/2000/02/xml-ref">
		<DATAREF>// ENCRYPTED["Payee_Account" and @Index="1"]</DATAREF>
		</Transform>
		</Transforms>
		</CipherReference>
		<CipherValue>GJm8G8ETkVJeMLk1wGsDo4E6ib9zIoyvhVEyaiQJPIE8A1YLrtKiVRobVS9emPaL</CipherValue>
		</CipherData>
		</EncryptedData>
Cipher Data	}	

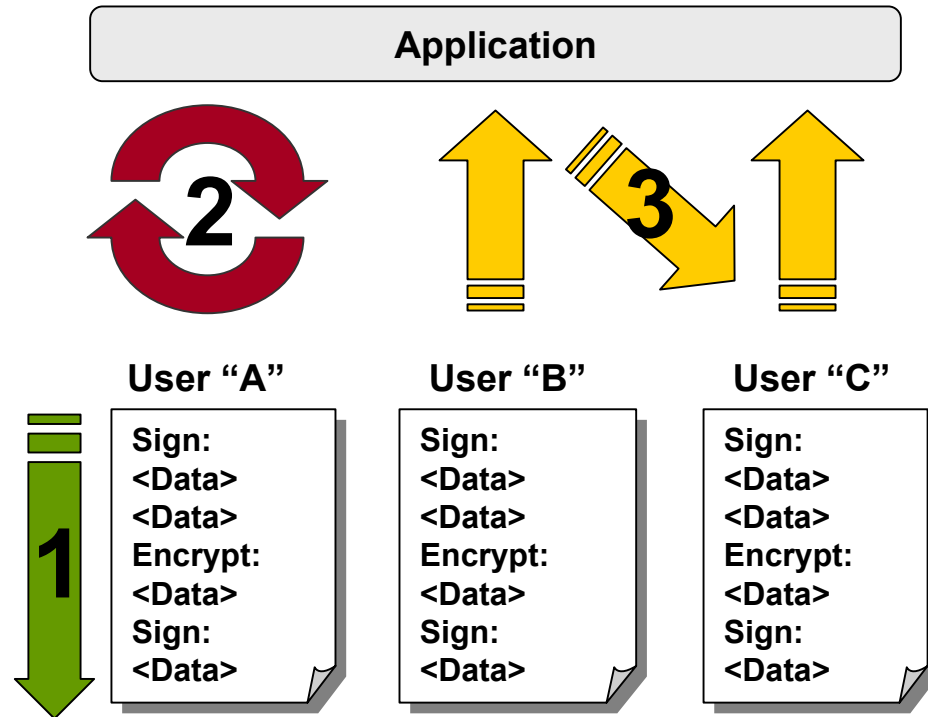
Important Note:

“In accordance with the requirements document the interaction of encryption and signing is an application issue and out of scope of the specification”. [W3C XML Encryption Syntax and Processing, December 10, 2002 para 6.1]

Encryption in an interactive web application and throughout a workflow process is, however, anything but a “unique” event. It is a ballet of cryptographic operations that are choreographed in interdependent sequences as data is exchanged between users within a web application. Three such “sequences” need to be defined, in a manner that ensures they are all coherent.

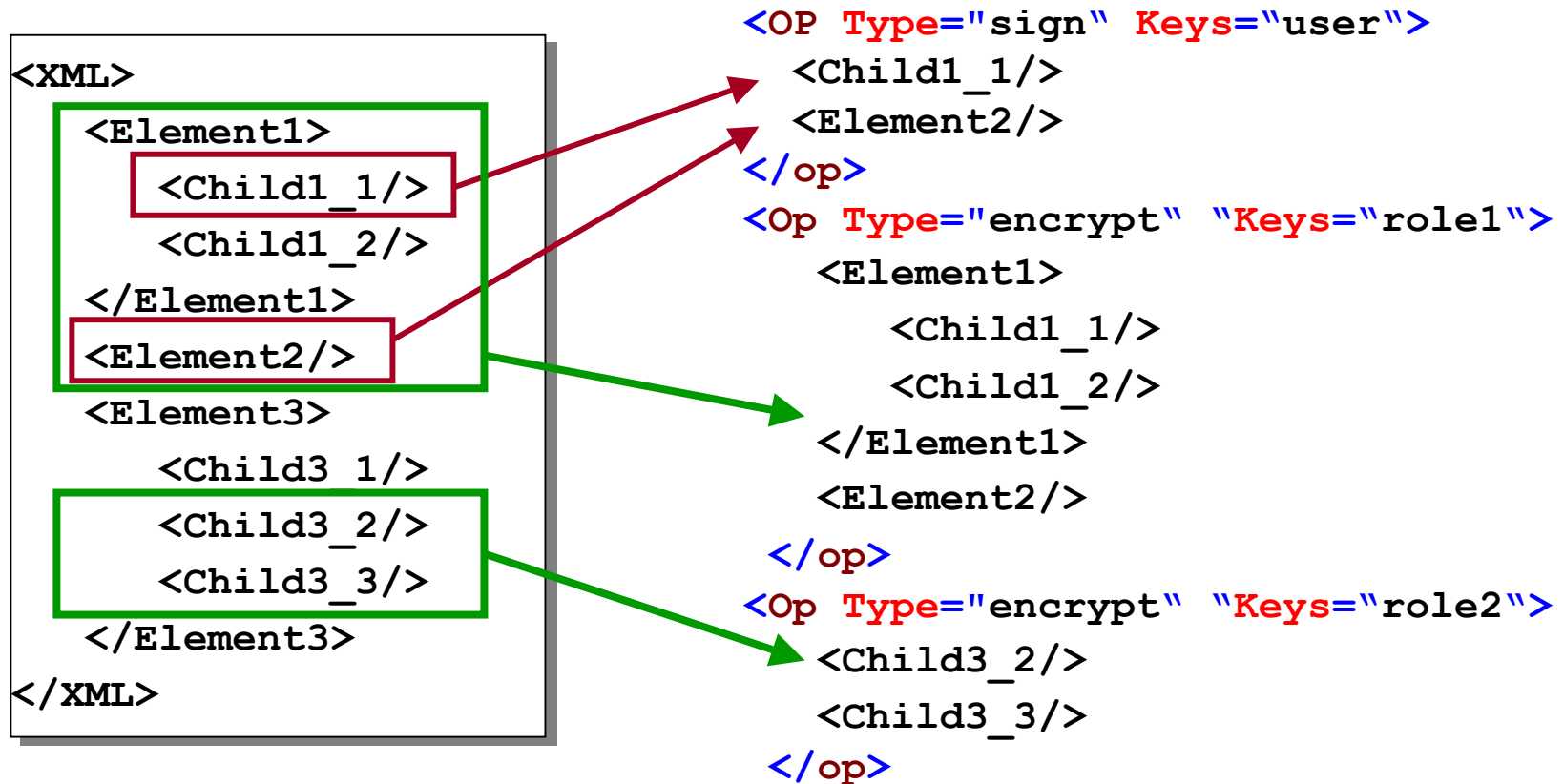
(2) Interactive
sequence defines operations at application and user locales as data moves back and forth.

(1) Internal
sequence describes which elements to be processed within the document and in what order crypto operations occur.

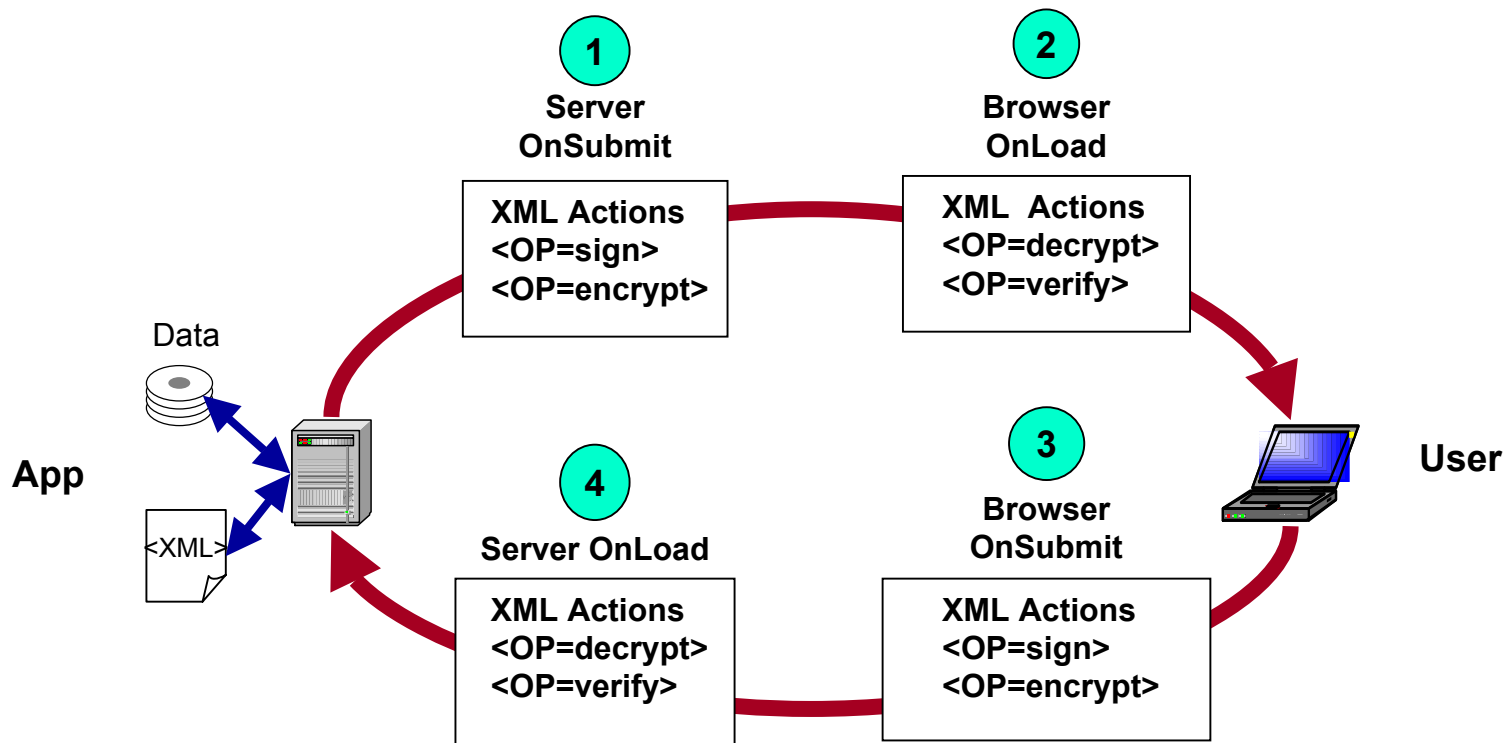


(3) Workflow
sequence defines operations as encrypted data moves between users

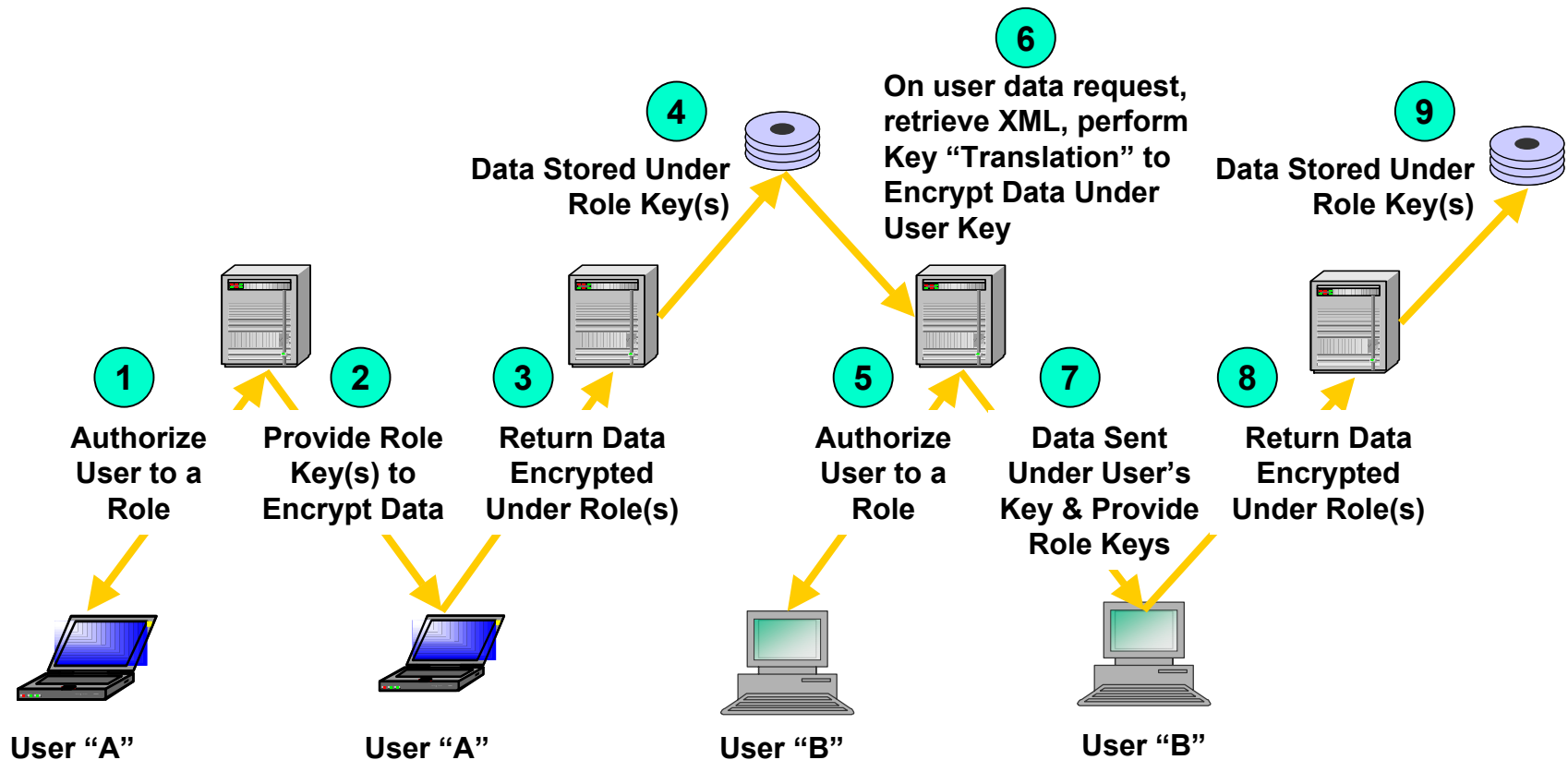
The internal sequence defines which XML elements in the document are to be cryptographically protected, in what order using what keys and algorithms.



The Interactive sequence describes what happens at the 4 points in time where XML cryptography takes place as data moves from the application to the user and back. The OnSubmit and OnLoad events are typically cryptographically mirror images of each other.



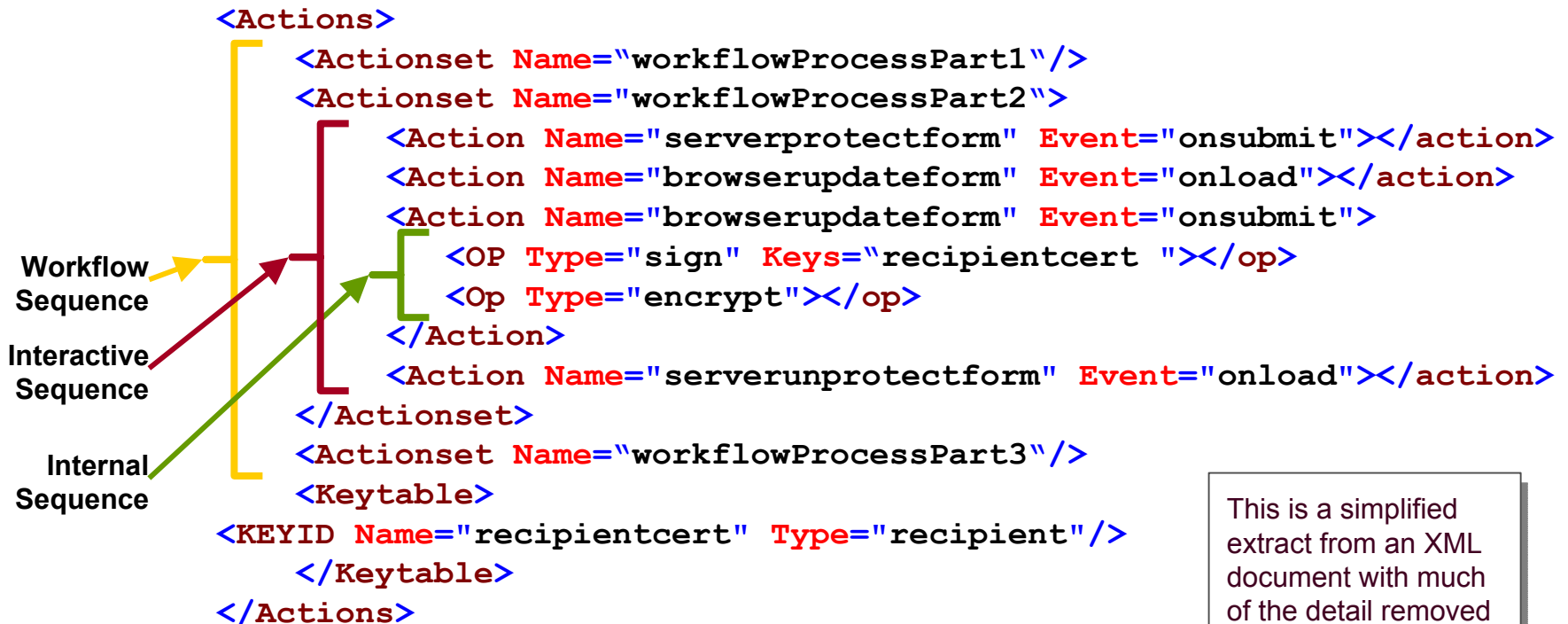
In the Workflow sequence, role based encryption is used to allow authorized persons, whose identity is not known when the data is encrypted, access to the encrypted data.



At each point in this series of sequences questions need to be answered, and the responses have to be coherent throughout the sequences:

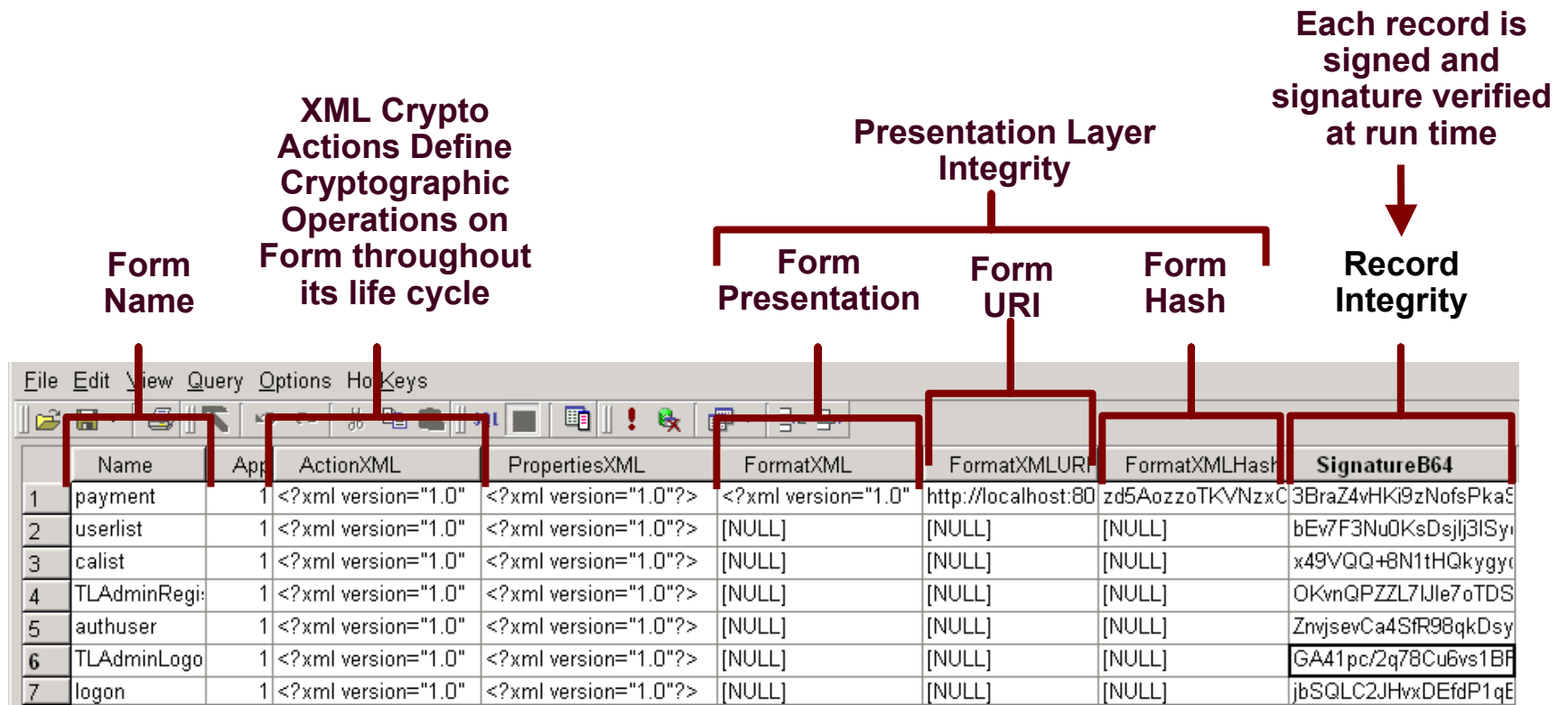
- **What is the event that initiates a cryptographic operation?**
- **What cryptographic operations should be performed for that event?**
- **In what order?**
- **Which XML data elements are to be processed?**
- **Where is the “protected” output to be stored?**
- **How should the original data type be recorded (if not XML)?**

Our approach is to define the cryptographic sequencing in XML as <Actions>. Each <Action> is a cryptographic operation associated to a specific HTTP event. These are collected into an <Actionset> that describes a complete interactive sequence.



This is a simplified extract from an XML document with much of the detail removed

The <Actions> XML is itself cryptographically protected to ensure its integrity. <Actions>, and all TrustLogic XML objects, are stored in a database where each record is signed and verified at run time. Objects that fail signature verification will not be processed



This is a partial view of the Forms table from TLData, the metadata database of TrustLogic.

[Interface to table shown here is MySQL Control Center]

“Object Manager” the TrustLogic interface to <Actions>, and all TL objects, uses certificate based authentication and role-based access to control the modification of <Actions>. (Approve Payment shown)

The screenshot displays the TrustLogic Object Manager interface. On the left, a tree view shows the hierarchy of objects under 'TrustLogic Root', including 'Applications', 'E-PAY-PLI', and 'GLOBAL'. Annotations on the left side identify these as 'Protected objects for E-PAY application', 'Other applications', and 'Global objects'. The main pane shows the 'payment' object expanded, with sub-objects like 'RequestPayment', 'ApprovePayment', and 'IssuePayment'. A yellow bracket highlights the 'ApprovePayment' object, labeled as 'Workflow Sequence'. A red arrow points to the 'sign' operation, labeled as 'Interactive Sequence'. A green arrow points to the 'encrypt' operation, labeled as 'Internal Sequence'. The bottom pane shows the configuration for the 'sign' operation, including 'Operation type: sign', 'Operation qualifiers' table, 'Select data' list, and 'Select keys' list. A blue arrow points to the 'Reason for Signing' field in the 'Operation qualifiers' table, labeled as 'Detail of a specific XML cryptographic operation'.

Protected objects for E-PAY application

The TrustLogic database protects all security related XML objects: Authority, User and Role profiles, Actions and Rules, at both the application and global level.

Other applications

Global objects

Workflow Sequence

Interactive Sequence

Internal Sequence

Detail of a specific XML cryptographic operation

Name	Value
Title	Reason for Signing
Reasons	(list)
Keys	default

Select data:

- _user_root
- Serial
- Request_Fname
- Request_Lname
- Date_Request

Select keys:

- TLCert
- OwnerDN

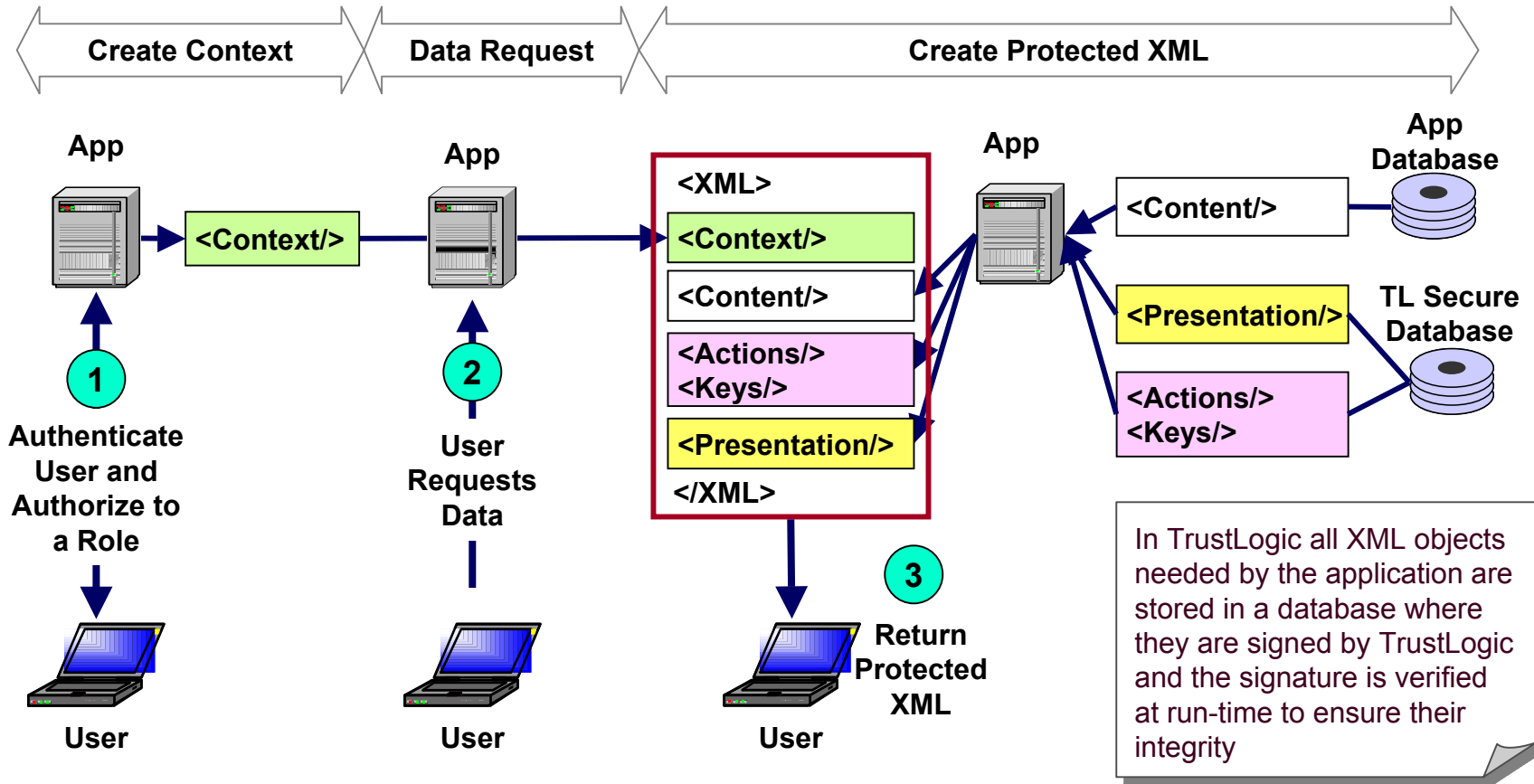
We intentionally define the cryptography on collections of data elements. Processing data elements one-by-one would:

- Be excessively onerous on the processor
- Make cryptographic sequencing very difficult
- Result in very large XML documents
 - ◆ Each of XML DSIG and XML ENC structures add up to 20 to 30 lines of XML for each element signed or encrypted

The <Actions>, and their associated <Keys> needed for role encryption, complete the structure of our protected XML document. They are an integral part of the document to ensure the protection of the XML at all times.

```
<!-- Example of structure for an XML payload -->
<Transaction>
Context [ <Header/> <!-- includes Context -->
          <Body>
            <BodyContent>
              <SecureForm>
                <Data/> <!-- Content -->
                <Actions/> <!-- Cryptographic instructions -->
                <Keys/> <!-- Keys for role encryption -->
              </SecureForm>
            </BodyContent>
          </Body>
Presentation [ <Presentation/> <!-- Presentation layer -->
              </Transaction>
```


At run time, the protected XML is put together from the `<Context/>` created during user authentication, the `<Content/>` provided by the application and `<Actions/>` and `<Presentation/>` stored by TrustLogic. The complete XML can be preserved in the Audit Vault.



To support existing document formats, TrustLogic provides for attached files, of any format, within the XML document, and extends to these attachments the same signature and encryption as on the XML. TrustLogic provides a standard, non-proprietary means to “sign” any file regardless of format.

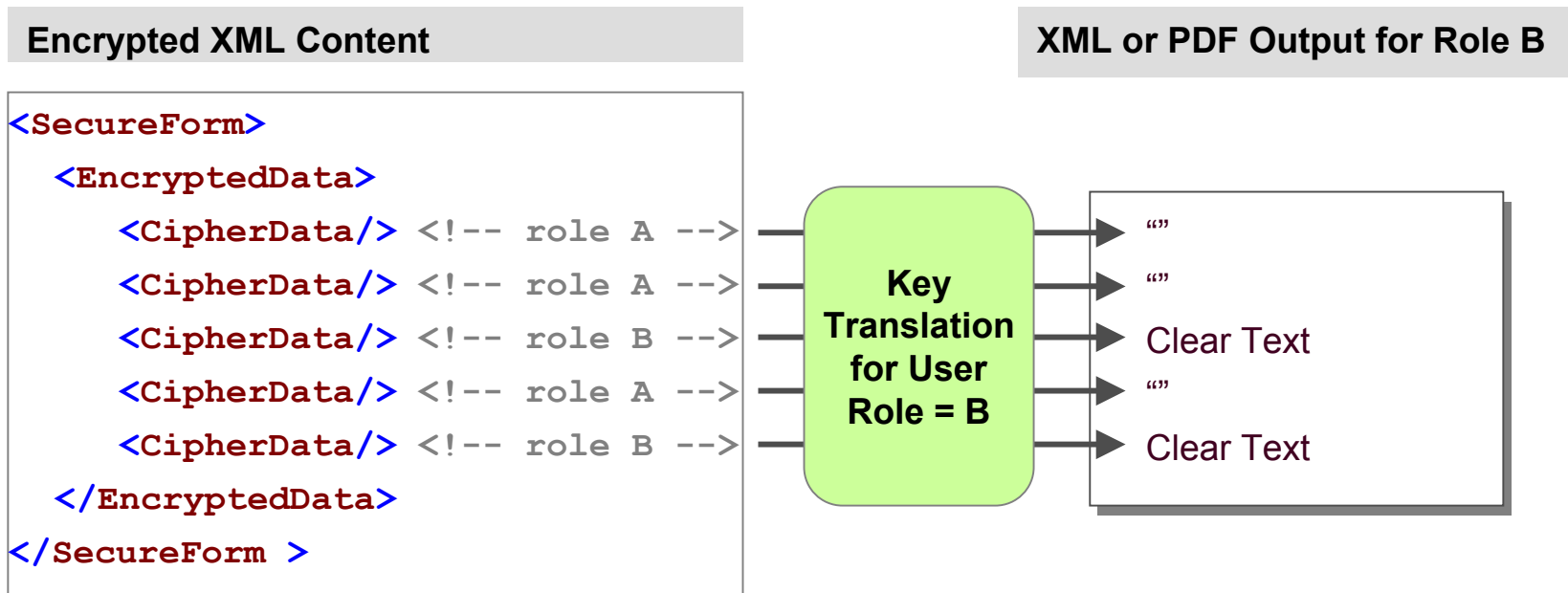


The embedded file becomes part of the protected XML document, the XML can be used as “metadata” to describe the attachment.

Role based encryption, and the ability to save the document as XML or searchable PDF, delivers “built-in” redaction capabilities.

To generate a “redacted” document that exposes data as defined by the data privileges of Role B is simply a case of retrieving the document as Role B. The document can be saved (as XML or PDF as appropriate) for distribution.

A community of users with different data access privileges can exploit an XML document repository, each retrieving only those specific data elements from each XML document to which they are authorized.



Now we can review the demo again to see how data confidentiality is maintained.

Retrieve XML document

■ Client side operations

- ◆ Log as HR-A and issue payment
- ◆ Log on as Clerk-A and review payment issuance
- ◆ Log on as Manager B and review payment issuance
- ◆ Log on as Treasury and review payment issuance

■ Server side views

- ◆ Sequential records
- ◆ Verify signature on payment issuance

Each user, depending on the location of their role in the role hierarchy has a different “view” of the XML data in the saved form, consistent with their role privileges.

Each will have access to data elements encrypted for their role or a subsidiary role. and will see the signatures on the data they have access to.

The <Actions> in this form first sign and then encrypt the data, including the signature (the sequence is defined by the application in the <Actions/>.

As the signature on encrypted data is also encrypted, signatures on data to which the user is not authorized will not be shown to that user.

TrustLogic, and all the XML it creates and uses, conforms to the following:

- **XML**

- ◆ **W3C Schema, W3C Digital Signature, W3C Encryption**

The cryptography supports the following:

- **Algorithms**

- ◆ **AES, 3DES, DES, RC2, RC4 / RSA / SHA-1, MD5**

- **PKI**

- ◆ **Tokens: PCKS#12, PKCS#11 (Smartcards & Biometrics), SC/PC (Smartcards), Entrust, MS Windows Registry**
- ◆ **Certificate Processing: US DOD (Certificate processing is based on issuer and will conform to issuer's certificate profile and policies)**

There are other issues that may have to be managed to achieve trustworthy digitally signed XML in a web application:

- Cross-domain trust domain management (PKI)
- Cross-domain identity and user key management (PKI)
- Support of multiple client side Token interfaces (PKI)
- Session security on XML transport between browser and server
- XML <> browser interaction
- XML <> application data interface

These have been addressed but are out of scope of today's presentation.

We demonstrated today a COTS product, using a COTS browser as the user interface, to interactively create, modify, store and retrieve XML documents that have been signed and encrypted.

The technology is:

- ◆ Consistent with NARA guidelines for creating “trustworthy” digitally signed documents in XML
- ◆ Consistent with NARA guidelines for the preservation and retrieval of such documents
- ◆ Consistent with Federal requirements for privacy and confidentiality of records within workflow processes through role based XML encryption
- ◆ Capable of inter-agency workflow through cross-domain authentication, identity management, and multiple token support
- ◆ Capable of on-line and off-line XML document editing while preserving data security (signature and encryption) at all times.
- ◆ Designed to allow multiple presentations of the same XML content according to the user interface (PC, Telephone, Print, Fax, etc.) while ensuring presentation integrity
- ◆ Capable of attaching files of any type, to the XML document. Attachments are signed and encrypted as any XML element

To support secure inter-agency workflow on XML documents TrustLogic provides comprehensive cross-domain authentication, authorization, and policy services.

■ Authentication

- ◆ **Cross-domain identity management and credential validation**
 - ◆ Key based, cross-domain identity management
 - ◆ Support for leading credential validation technologies
- ◆ **Application specific cross-domain trust management**
 - ◆ Define which CA's are recognized
 - ◆ Define what assurance level attached to each
 - ◆ Apply CA specific certificate policy and profile processing at run time
- ◆ **Support for multiple client side token technologies**
 - ◆ Participants in different domains can use different tokens
 - ◆ Same participant can simultaneously use different tokens

■ Authorization

- ◆ **Hierarchical role-based authorization, cryptographically enforced**
 - ◆ All users given a role in the application when authenticated
 - ◆ Role has cryptographic profile (which algorithm, key length)
 - ◆ Access to data elements and application processes can be associated to possession of a specific role
 - ◆ Data elements will be encrypted under role keys at point of origin and while at rest
- ◆ **Application centric (not network centric)**
 - ◆ Each application can have one or more role hierarchies
 - ◆ Users can have multiple roles in the same application and different roles in different applications

■ Policy

- ◆ **Hierarchical, role based rules**
 - ◆ Rules associated to a role, applied to all subsidiary roles unless subsidiary role has its own rule
- ◆ **Multiple rule types**
 - ◆ **Boolean:** to control execution of operations
 - ◆ **String:** for comparison type rules
 - ◆ **Number:** for comparison type rules
 - ◆ **Java:** to execute specific code, including integration with non-web based applications
- ◆ **GUI interface provides single point of control for managing rules across all applications**

Thank you.

Matthew McKennirey

mmckennirey@conclusive.com

703 734 3000 ext 120

Appendix

- **Demo technical details**
- **Current Activities : Encrypted XML database**
- **Key Translation Detail**
- **Run Time Data Request Flow of Events**
- **Low Lifetime Cost**
- **Trust Domain Policy**

For the technologists, today's demo used the following technologies:

- ◆ **Server side**

- ◆ **TrustLogic Server 2.0 (J2EE application)**
- ◆ **OS: Windows 2000 + ver. 1.3.1 Java Runtime Environment**
 - ◆ TrustLogic can run on MS NT/W2000 / Solaris / Linux servers
- ◆ **Web Server: Tomcat 4.0.6 (open source - stand alone mode) running JSP**
 - ◆ TrustLogic can support any web application language that provides for remote procedure calls
- ◆ **Database: MySQL ver. 4.0 (open source)**
 - ◆ TrustLogic can run on any SQL database with JDBC driver
- ◆ **Directory: Sun iPlanet ver. 5.0 (not required by TrustLogic, used to store CRLs and some CA certs for demo)**
 - ◆ TrustLogic can interact with any LDAP ver. 2.0 directory

- ◆ **User side**

- ◆ **TrustLogic Agent 2.0 (Java app, includes open source XSLT , XFO)**
- ◆ **OS: Windows 2000 + ver. 1.3.1 JRE (client & server on same machine today)**
- ◆ **Browser: IE 6.0 (Agent requires ver. 5+ browser, or ver. 6+ for XSL support)**
- ◆ **Tokens**
 - ◆ **User A: Entrust *.pdf (Entrust CA)**
 - ◆ **User B: PKCS#12 (*.pfx) (Sun iPlanet CA)**
 - ◆ TrustLogic supports all leading software and hardware tokens

The focus of our current activities, funded by NIST, is an encrypted XML database which supports searching encrypted XML.

[Yes, searching encrypted data appears to be an oxymoron. We specialize in “can’t be done”.]

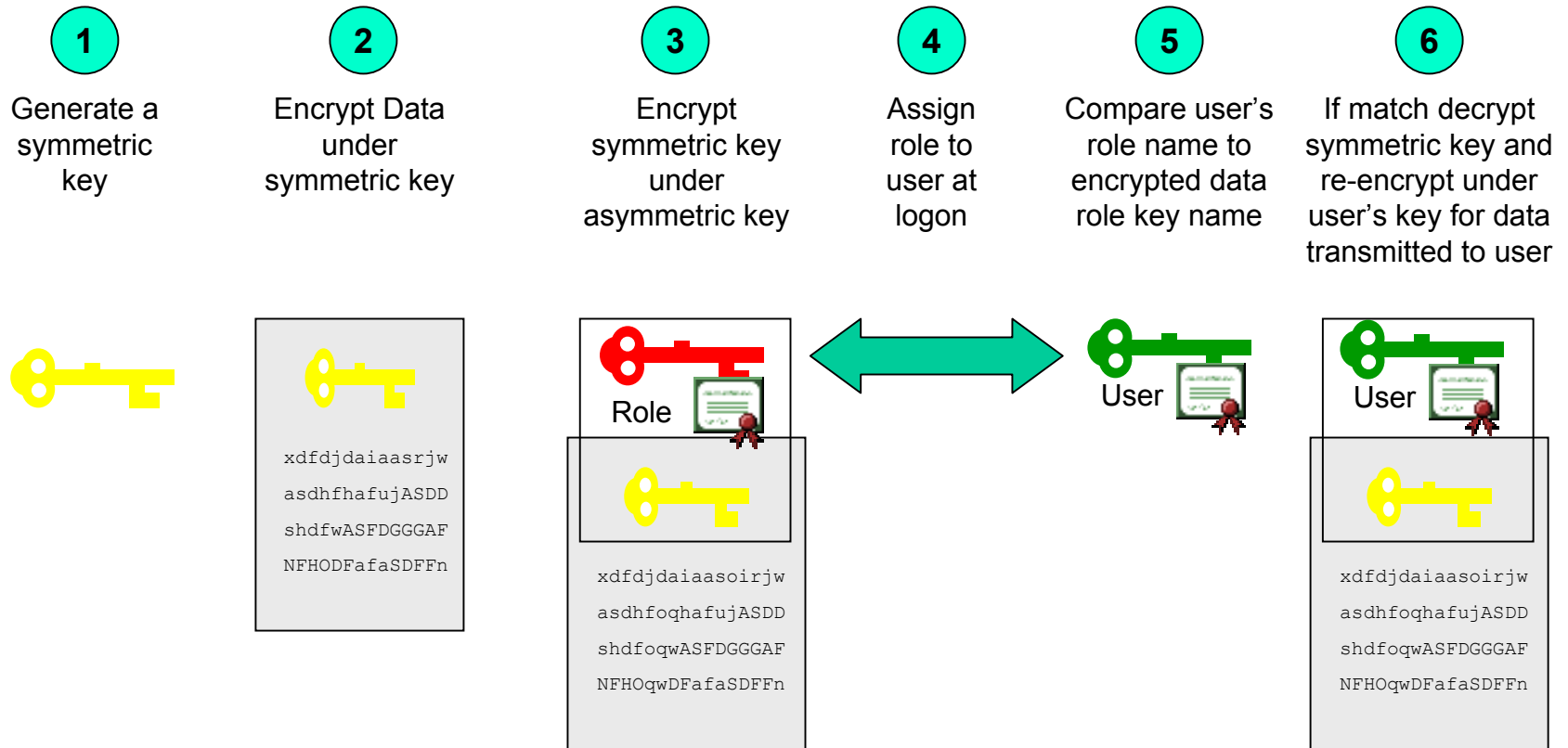
■ **Activity 1: (NIST) Future Secure XML Database**

- ◆ **Sensitive XML data is stored encrypted in the XML database under hierarchical role based crypto access**
- ◆ **Encrypted XML can be searched, and matches returned encrypted specifically for the authorized user**
 - ◆ **Constraint: Supports complete value match only (no partial element search)**

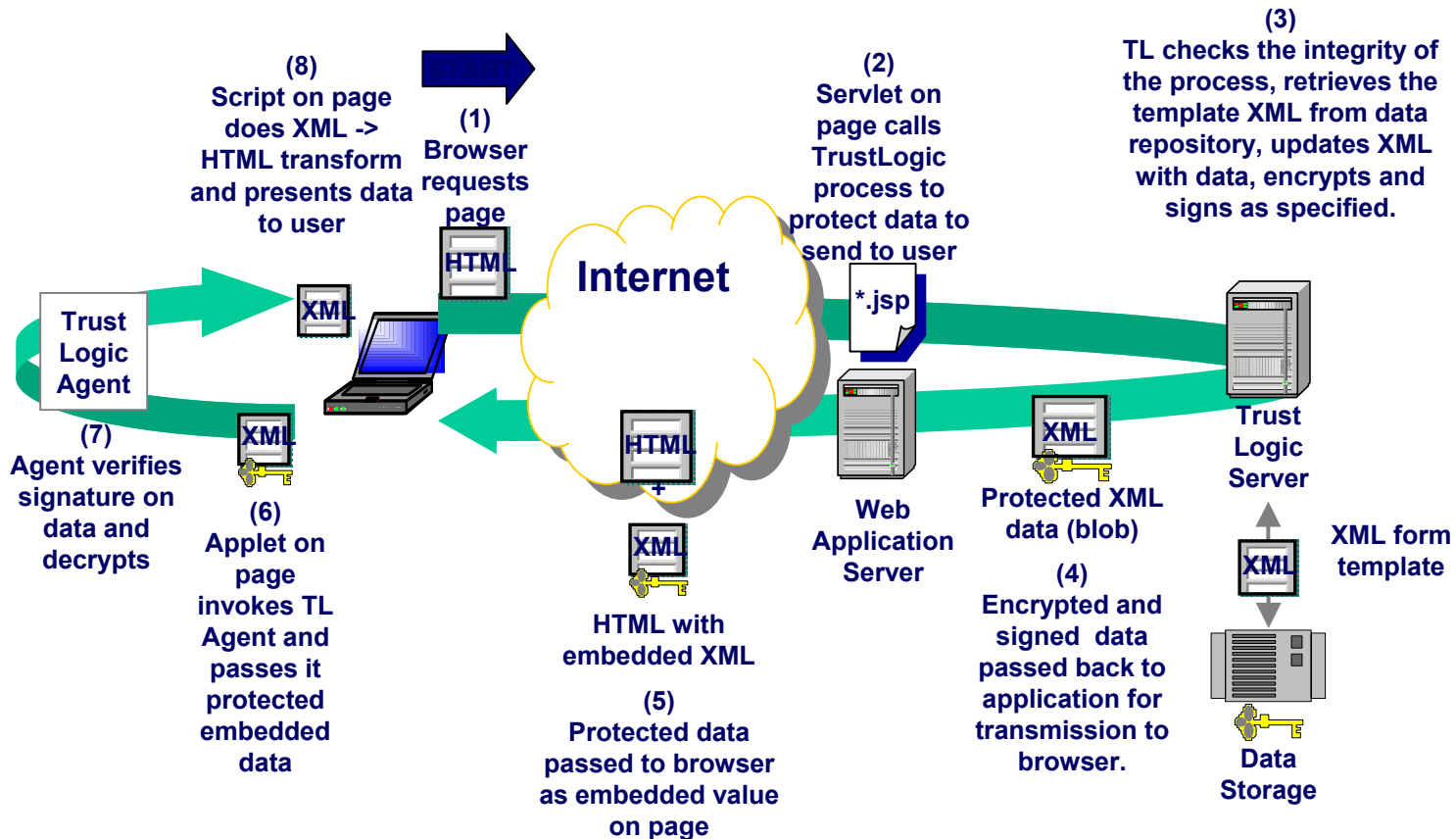
■ **Activity 2: (DOD) Secure XML Across Domain Boundaries**

- ◆ **Transfer Secure XML between physically distinct domains of different security levels through non-XML aware existing accredited guard devices**
 - ◆ **Constraint: Coherent role structure and access to role keys in applications in physically distinct domains**

The marriage of symmetric and asymmetric cryptography.



TrustLogic manages all aspects of moving protected XML back and forth using standard HTML as transport.



TrustLogic's architecture is designed to minimize deployment cost and for a low lifetime cost.

- **Highly modular, high degree of reusable modules with extensive templates for rapid integration; each successive application builds on previous applications and is less expensive to deploy**
- **Application is separated from PKI technologies and policy such that these can change / updated without touching the application**
- **Uses standard Java to define security tasks (“processes”) that define behavior (no proprietary scripting languages to master)**
- **Tools generate secure XML forms from existing HTML / JSP pages**

TrustLogic's ability to define a trust domain in 2 dimensions ("horizontally" in terms of which CAs are recognized, "vertically" in terms of assurance level associated to each CA) and support for leading tokens means organizations do not need to operate their own CA and can accept certificates from any CA it decides to "recognize".

- Organization does not have to operate its own CA and issue certificates.
- Organizations can extend "trust" to any CA it wishes to recognize
- Applications create their own sub-set of recognized CAs from those recognized at organization level
- Applications apply their own assurance levels to each CA which can be used within rules to determine privileges of users from each CA
- Can use token type as additional criteria for establishing privileges