Design and Implementation of a Secure Service-Oriented Workflow Platform

Tao-Ku Chang Jyun-Siao Huang

Department of Computer Science and Information Engineering, National Dong Hwa University

Hualien, Taiwan, ROC

{tkchang, m9921029}@mail.ndhu.edu.tw

Received 12 April 2013; Revised 31 August 2013; Accepted 28 October 2013

Abstract. The current focus of enterprises on automating processes has led to the extensive application of workflow management systems. To integrate traditional workflow management systems with others has previously been difficult; however, rapid development of Web services has solved the communication problems among heterogeneous platforms and allows for the efficient integration of legacy systems. This paper emphasizes the design and implementation of a secure service-oriented workflow platform effectively integrating heterogeneous platforms. Moreover, current system architectures could adopt the developed WS-Security application framework rapidly and inexpensively.

Keywords: Service-oriented architecture, Workflow, Security

1 Introduction

The application system in an organization or enterprise may have been developed over many years, and the resulting diversity in the types of database system and system platform can impede the interoperability between system services. Considerable manpower is needed to convert data accessed from different systems. For example, if one department of an organization wants to use the system service of another department, the personnel in charge of each system need to coordinate diverse message exchange formats, which can require a considerable amount time and money.

This situation is especially difficult when sharing service information and integrating systems among different enterprises, due to the likelihood of greater differences in the application systems used by different organizations. One of the current solutions is to employ Web-services [1] techniques that provide a standard means of interoperation between different software applications running on diverse platforms and/or frameworks.

The exploding popularity of the Internet has brought strong growth in e-commerce, and prompted more enterprises to employ web services and WfMSs (workflow management systems) [2][3][4][5] to support their processes. The first goal is to integrate WfMSs between organizations and consider how to support access control, since services may have to access data maintained by other sides. This paper describes the design of a secure service-oriented workflow platform that can design workflow and integrate heterogeneous application systems.

The remainder of this paper is organized as follows: Section 2 gives an overview of relevant technologies and standards, Section 3 presents the proposed secure service-oriented workflow platform, Section 4 describes the syntax of a secure service-oriented WfMS document, Section 5 presents our implementation and experimental results, and Section 6 draws conclusions about the work described in the paper.

2 Related Techniques and Standards

Fig. 1 shows the service-oriented architecture of Web services. First, the service requester generates request message *X* in SOAP [6] format according to the WSDL [7] document obtained from the service provider or service broker (UDDI [8]) (note that the WSDL is an XML-based [9] language for describing Web services and how to access them). The service provider replies by verifying if the service requester is authorized to obtain the service according to its authentication policy. It then performs the operations specified in *X* and returns the execution results in another SOAP document, *R*. Note that SOAP is not the only message format for implementing the SOA. In this paper, SOAP is used in the proposed system. A high-level language such as the Web Services Business Process Execution Language (WS-BPEL) [10] is the standard for assembling a set of discrete services into an end-to-end process flow, radically reducing the cost and complexity of process integration initiatives.

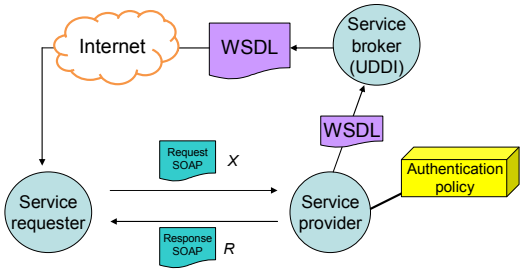

Fig. 1. Service-oriented architecture

However, these standards do not provide a security mechanism for protecting enterprise information when data are being transferred. Security is one of the main concerns when developing business applications, and this has prompted the development of security specifications for Web services. This involves applying techniques such as encryption or adding digital signatures to XML documents in the SOAP. The W3C proposed a specification for XML document encryption and the handling of digital signatures in 2002: XML Signature Syntax and Processing [11] and XML Encryption Syntax and Processing [12]. The international organization OASIS is starting to establish a standard for Web-services security, with the draft being called WS-Security [13]. The eXtensible Access Control Markup Language (XACML) is a standard defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate authorization requests according to the rules defined in policies [14].

A workflow is a computational model of computer-supported cooperative work [15]. Business tasks are modeled as workflow processes that are automated by the WfMS. The workflow model (also referred to as a workflow process definition) is the computerized representation of the business process. It defines the starting and stopping conditions of the process, activities in the process, and control and data flows among these activities.

The Workflow Management Coalition (WfMC) developed a reference model for a WfMS. Fig. 2 depicts the workflow reference model that specifies the generic components and interfaces that form a WfMS, including workflow engines, process definition tools, workflow client applications, a worklist handler, and administration and monitoring tools. There are five interoperable interfaces for the generic components; these define five critical software component interfaces from the viewpoint of functionality to promote standardization of message exchange and implement interoperability between different processes [16].

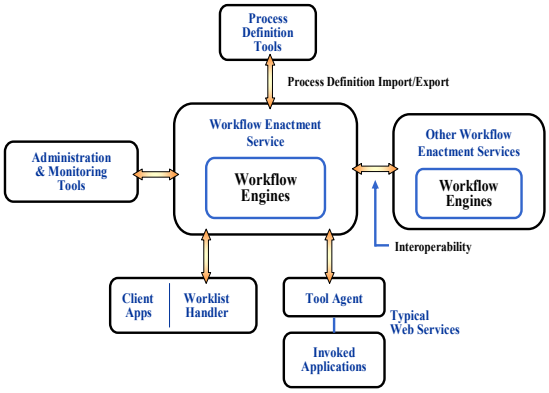

Fig. 2. Workflow reference model [15]

3 The Proposed Secure Service-Oriented Workflow Platform

Fig. 3 shows the system architecture for the proposed WfMS platform. The process definition designer first uses the process definition tool to design the process definition and then stores it in a database. The user then connects to the workflow server and sends a request, and generates a new process as described in the secure service-oriented WfMS document. When other participants in the process log into the system, they will acquire the worklist and select the work item to execute its task.

A complete workflow should include the following execution steps:

(1) Analyze the goal of the workflow. This involves determining the numbers of participants and arranging tasks for each of them. Workflow developers use process definition tool to design a workflow, then generate the secure service-oriented WfMS document and store it in a database.

(2) Start a new workflow. Users will connect to the workflow server and select the workflow that is going to be processed. This system will acquire the corresponding WfMS document and place an assigned serial number in the document's header. The workflow engine reads the initial processing arguments from the workflow control data and starts a new workflow. The workflow system records the result in the same WfMS document, and follows the security definition from this document to encrypt and embedded the digital signature therein; the activity has then finished.

(3) The system will generate a new version of the secure service-oriented WfMS document after the execution of an activity. Meanwhile the system will determine and notify the next participant from the definition of transitions and activities to start the next workflow process.

(4) Steps 1 to 3 are repeated until the workflow process is successfully completed. If the process is aborted by any abnormal condition, this will be recorded in the log in the database.

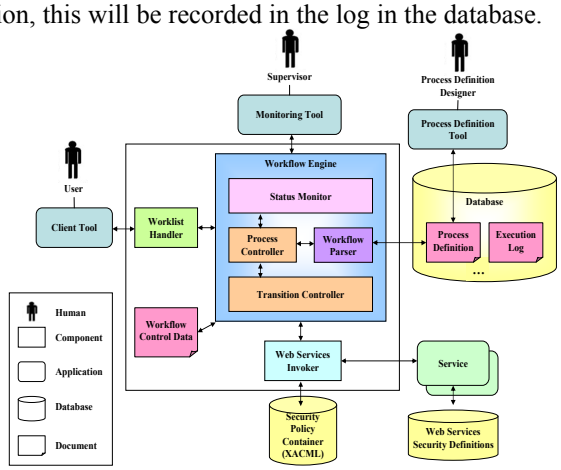

Fig. 3. The Secure service-oriented WfMS architecture

The many applications provided by Web services nowadays make it necessary to ensure the security of data transmission. However, adding a new security mechanism to the client or server requires modifications to the programs in the system. We implemented security authentication for the service-oriented WfMS and developed a WS-Security application framework that can be deployed to the service that implements WS-Security without changing the existing Web-services architecture. The operation process is shown in Fig. 4.

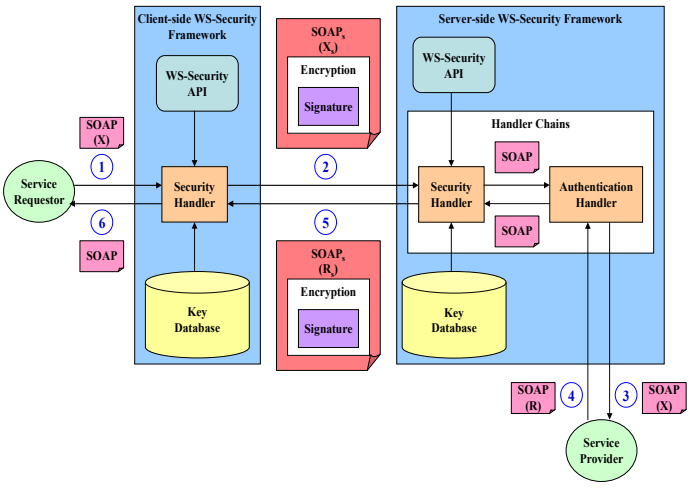

Fig. 4. Web services security

In this security model, Web services invoker will check the security policy described by XACML to determine whether user can access Web services or not. Moreover, the service requester and service provider have to write the client-side and server-side security handler programs, respectively. The security handler programs invoke subroutines in the WS-Security API to interpret the security policy stored in the database. The API provides a convenient way for the proxy programs to set up the required keys. After all the required keys are obtained, the proxy program can then instruct the WS-Security API to secure or desecure SOAP messages. A request message from a workflow participant to execute Web services will be captured by the security handler, which will start the encryption and add a digital signature to the SOAP message to ensure the confidentiality and integrity of data exchange. The authentication handler will then authenticate the identity of the participant to ensure that only authenticated people can access the targeted services.

There are six steps for securely invoking Web services in the proposed operational model:

(1) The service requester sends SOAP document X to the client-side proxy.
(2) The client-side security handler invokes routines in the WS-Security API to generate secured SOAP document X_s.
(3) The server-side security handler receives X_s and invokes the WS-Security API to desecure it. The desecuring process includes decrypting cipher data in X_s and verifying all the digital signatures embedded in X_s. The WS-Security API supports obtaining the identity of the service requester from the embedded digital signatures. It can then check the obtained identity against its authentication policy. The desecured document X is sent to the service provider.
(4) The response message is stored in SOAP document R, and this is sent to the server-side security handler.
(5) The server-side security handler invokes routines in the WS-Security API to generate secured SOAP document R_s.

The client-side proxy receives R_s and invokes routines in the WS-Security API to desecure it. In addition to decrypting the cipher data in R_s, the digital signatures embedded in R_s should be verified.

4 The Syntax of the Secure Service-Oriented WfMS Document

The syntax of a secure service-oriented WfMS document is designed according to the above-mentioned security requirements as well as the architecture shown in Fig. 3. The structure of a secure service-oriented WfMS document consists of five parts (see Fig. 5): the header, workflow definition, security definition, and digital signature. The root element of the example WfMS document has the start tag <WfMS:workflow xmlns: = "http://www.WfMS.ndhu.edu.tw">.

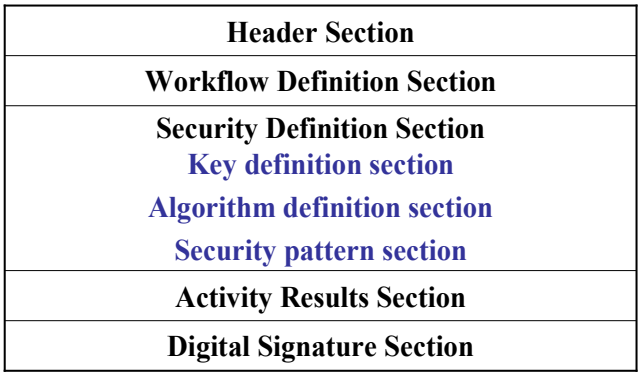

Fig. 5. The secure service-oriented WfMS document structure

- Header Section

Each secure WfMS document represents a workflow process. When a user creates a new workflow, the workflow engine will assign a new serial number to the secure WfMS document and place this in the header section.

- Workflow Definition Section

The workflow definition section provides a basic definition of the process, including its activities and transitions. Each activity describes the operations performed by related members in this workflow. Each operation is represented by certain information tools (e.g., a user interface for creating forms), and it reads, creates, or modifies the content recorded in an XML document. Transitions are defined by writing down the process of the workflow. When a user finishes executing its activity, the workflow engine will automatically follow the transition definition to determine which activities should be performed next. In this paper, WS-BEPL [10] is used in workflow definition.

- Security Definition Section

The security definition section has three parts: the key definition, algorithm definition, and security pattern. The key definition describes the link, name, type, and location of the key. The algorithm definition is for encryption and signature. The security pattern describes the adopted combination of key and algorithm definitions.

- Digital Signature Section

The digital signature definition section defines when and how to construct digital signatures. The generated signature is stored in this section.

We specify syntax of security and digital signature definitions that are shown in the appendix A and B in the Backus-Naur Form in this paper. In a secure WfMS document, the workflow and security definition sections are

generated when a workflow is created and cannot be changed. The workflow must follow the definition to perform the required activities. Some information will be added in the secure WfMS document during the execution of the workflow process.

5 System Implementation and Experiment Results

Based on the infrastructure mentioned in Section 3, we use ASP.NET to develop a secure service-oriented WfMS platform to help a user design workflow process. Fig. 6 illustrates the operation of our workflow platform implementation.

To create a new workflow, users establish an organization structure using organization designer tool on this platform (see Fig. 7). According to goal of the workflow, flow designers start to design the flow of activities in a workflow. Our platform also provides a GUI-based tool for users to construct it (see Fig. 8). Before executing a workflow, users can perform flow simulation to check whether the flow could work successfully or not. Fig. 9 shows the snapshot of simulation processes.

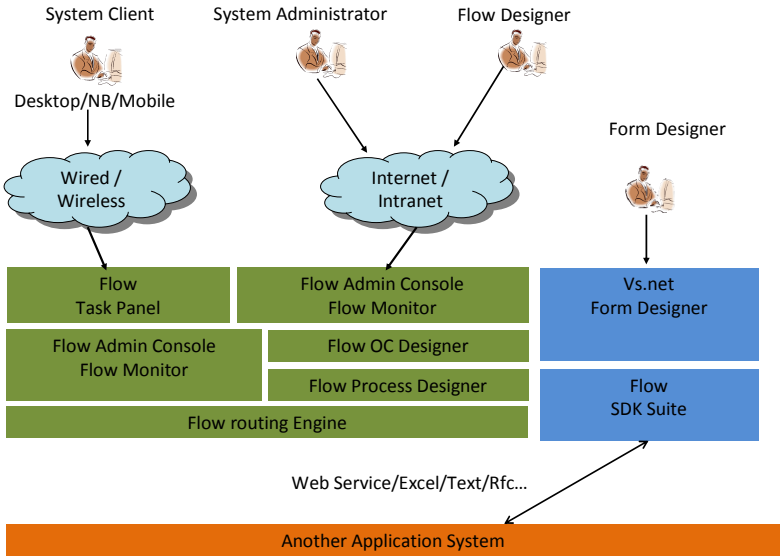

Fig. 6. The secure service-oriented WfMS platform implementation

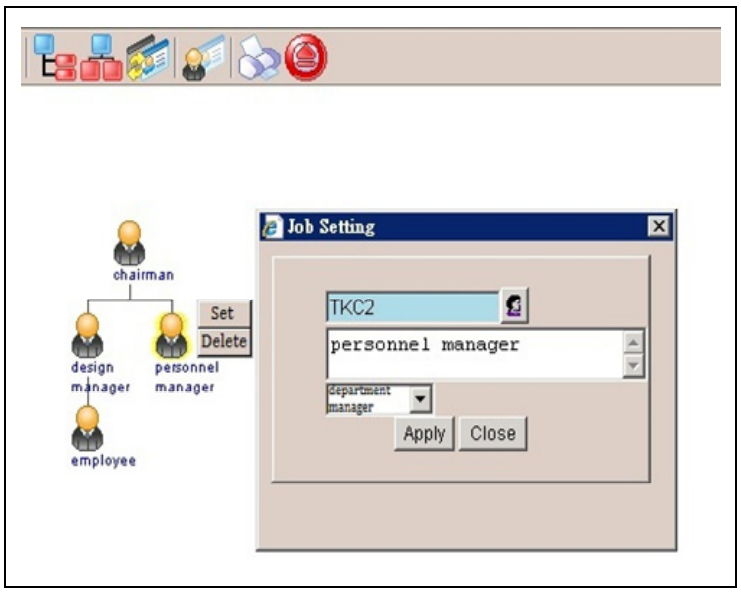

Fig. 7. The organization designer tool

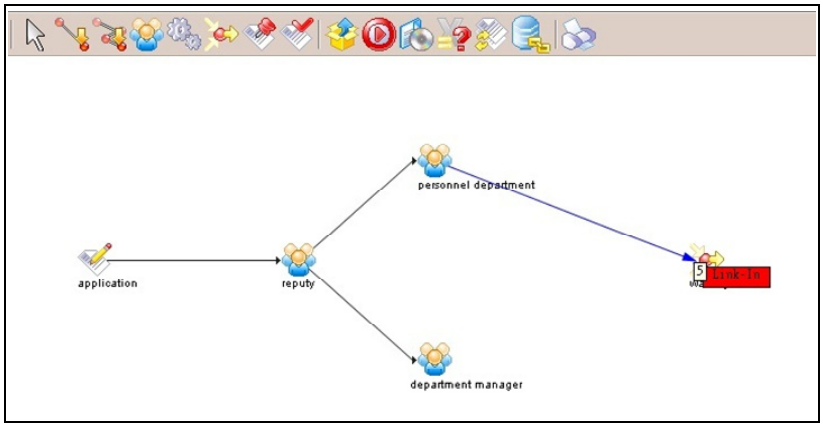

Fig. 8. The flow designer tool

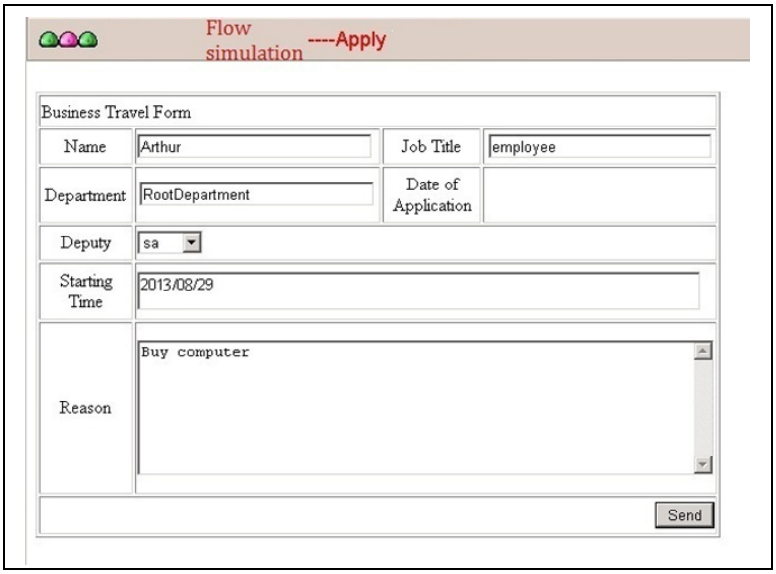

Fig. 9. The flow simulation

The workflow platform first reads an XML document to obtain the tree structure of the target XML document. The user sets up the security pattern for the elements and attributes in the editor's graphical interface. The user can generate a security pattern by selecting the available keys and algorithms, or input the related information of keys and algorithms manually.

Table 1. The times required to encrypt and decrypt the body of the SOAP message

Number of encrypted elements	Average time (in seconds)			
	100 bytes*		500 bytes*	
	Encryption	Decryption	Encryption	Decryption
10	0.6048	0.6750	0.6471	2.0538
20	0.6183	1.2096	0.7308	3.9798
30	0.6471	1.7865	0.8577	5.9913
40	0.6777	2.3346	0.9702	8.4942
50	0.7038	2.8683	1.1259	10.026
60	0.7317	3.4452	1.2933	12.1365
70	0.7731	3.9519	1.5885	17.0577
80	0.8298	4.5846	1.8846	16.6923
90	0.8856	5.1606	2.2482	18.9567
100	0.9567	5.7096	2.6154	21.6036

*Number of bytes to be encrypted in an element

We conducted experiments to evaluate the performance of the WS-Security API presented in Section 3. All the experiments were performed on a PC with a 3.4-GHz Intel Core i7-2600K processor, 4GB of RAM, the MS

Windows 7 operating system. All the encryptions and digital signatures used the RSA algorithm that implemented RSASSA-PKCS1 and RSASSA-PKCS1 with MD5 [17]. SOAP document had 101 elements: a tree plus its root node with 100 child element nodes, where each child node was associated with a text node. Each text node comprised either 100 or 500 bytes. Table 1 lists the execution times for encrypting and decrypting the body of the SOAP message, and Table 2 lists the times required for signing and verifying.

Table 2. The times required to sign and verify the body of the SOAP message

Number of encrypted elements	Average time (in seconds)			
	100 bytes*		500 bytes*	
	Sign	Verify	Sign	Verify
10	0.3240	0.0981	0.3231	0.1125
20	0.3654	0.1269	0.3798	0.1404
30	0.3942	0.1548	0.4644	0.2106
40	0.4563	0.1692	0.5481	0.2664
50	0.4779	0.1971	0.6606	0.3375
60	0.5211	0.2529	0.7596	0.4365
70	0.5769	0.2394	1.0413	0.6327
80	0.5904	0.3096	1.2510	0.8433
90	0.6615	0.3375	1.5336	1.0827
100	0.7317	0.4077	1.8279	1.3077

*Number of bytes to be encrypted in an element

6 Conclusion

We have designed a WfMS based on the reference model proposed by the WfMC that employs service-oriented architecture to effectively integrate heterogeneous platforms. The developed access control mechanism (XACML) and WS-Security application framework can be rapidly applied to an existing system architecture and increase the security of enterprise information exchange between stakeholders and customers. Moreover, the experimental results presented here demonstrate that applications exhibit good performance.

Acknowledgement

This work was supported in part by the National Science Council, Taiwan, under grant NSC-102-2221-E-259-015. The author wishes to acknowledge the comments offered by the anonymous reviewers.

References

[1] "Web Services Architecture," W3C Working Group Note 11 February 2004. http://www.w3.org/TR/ws-arch/.

[2] D. Georgakopoulos, M. Hornick, A. Shet, "Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure," *Distributed and Parallel Databases*, Vol. 3, No 2, pp. 119-153, 1995.

[3] M. Shi, G. Yang, Y. Xiang, S. Wu, "Workflow Management Systems: A Survey," in *Proceedings of International Conference Communication Technology Proceedings*, IEEE Press, 1998.

[4] W. Du and A. Elmagarmid, "Workflow Management: State of the Art vs. State of the Products," in *Proceeding NATO Advanced Study Institute on Workflow Management Systems*, 1997.

[5] "Workflow Reference Model Diagram 2010," Workflow Management Coalition Standard, http://www.e-workflow.org/standards/index.htm.

[6] "SOAP Version 1.2," W3C Recommendation 24 June 2003. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/.

[7] "Web Services Description Language (WSDL) Version 2.0," W3C Candidate Recommendation 27 March 2006. http://www.w3.org/2002/ws/desc/.

[8] Universal Description, Discovery and Integration (UDDI). http://www.oasisopen.org/committees/uddi-spec/.

[9] "Extensible Markup Language (XML) 1.0 (Fourth Edition)," W3C Recommendation 16 August 2006. http://www.w3.org/TR/xml/.

[10] "Web Services Business Process Execution Language Version 2.0," OASIS Standard, April, 2007.

[11] T. Imamura, B. Dillaway, and E. Simon, "XML Encryption Syntax and Processing Version 1.1," W3C Recommendation, 11 April 2013, http://www.w3.org/TR/xmlenc-core/.

[12] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon, "XML-Signature Syntax and Processing W3C Recommendation," 10 June 2008. http://www.w3.org/TR/xmldsig-core/.

[13] "Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)," OASIS Standard Specification, 1 February 2006. http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

[14] T. Moses, "eXtensible Access Control Markup Language (XACML) Version 3.0," OASIS Standard, 2013.

[15] *Computer Supported Cooperative Work (CSCW)*, Springer Netherlands, ISSN: 0925-9724.

[16] *Workflow Handbook*, Workflow Management Coalition, 2006.

[17] "PKCS #1 v2.2: RSA Cryptography Standard," RSA Laboratories, October 27, 2012. http://www.emc.com/emc-plus/rsa-labs/pkcs/files/h11300-wp-pkcs-1v2-2-rsa-cryptography-standard.pdf

Appendix:

A. Syntax of security definition section

```
Key Definition→
    <key_definition [key_link="Key Link"]>
        <key_name>Name of Key</key_name>
        [<key_type>String*</key_type>]
        Download Protocol {Download Protocol}
    </key_definition>
Download Protocol→
    <download_protocol
        location="URL adr"
        [security_pattern_name="Security Pattern Name"]
        [pki_certificate="PKI name"]
        [verify_certificate="Yes_or_No"]
        {proxy="URL adr {,URL adr}"}/>
Key Link→ String
Name of Key→ String
PKI_name→X.509|PGP|SDSI
Yes_or_No→Y|Yes|YES|N||No|NO

Algorithm Definition → Built-in Algorithm | Downloadable Algorithm
Built-in Algorithm→
    <algorithm_definition [algorithm_link="Algorithm Link"]
        use="(SECURITY|SIGNATURE|DIGEST)">
        <algorithm_id>
          Algorithm id
```

```
      </algorithm_id>
      [<cipher_format XML_text="(NO|YES)"/>]
   </algorithm_definition>
Downloadable Algorithm→
   <algorithm_definition [algorithm_link="Algorithm Link"]
      use="(SECURITY|SIGNATURE|DIGEST)">
      <algorithm_name> Name of Algorithm </algorithm_name>
      [<type> Type of Algorithm </type>]
      [<version> Version of Algorithm </version>]
      [<property Attribute Lists />]
      [<cipher_format XML_text="(NO|YES)"/>]
      Algorithm Download Protocol {Algorihtm Download Protocol}
   </algorithm_definition>
Algorithm Link → String
Algorithm id → String
Name of Algorithm → String
Type of Algorithm → String
Version of Algorithm → String
Algorithm Download Protocol →
   <download_protocol linking_method="DDL"
      [security_pattern_name="Security Pattern Name"]
      Jar_file_location="URL adr"
      Serialization_file_location="URL adr"
     {proxy="URL adr {,URL adr}"}/> |
   <download_protocol linking_method="plug-in"
      [security_pattern_name="Security Pattern Name"]
      location="URL adr" Attribute Lists
      {proxy="URL adr {,URL adr}"} ostype="OS type"/>
Attribute Lists → Attribute {Attribute}
Attribute → Attribute Name="String"
Attribute Name → String
OS type → Linux | windows 2000 | windows XP | Sun Solaris
Security Pattern Definition →
   <dsl:security_pattern name="Security Pattern Name"
      encrypted_format="W3C|DSL">
      <key_infomation>
        <encryption_key>
          Encryption Key Definition
        </encryption_key>
        <decryption_key>
          Decryption Key Definition
        </decryption_key>
      </key_infomation>
      <security_algorithm>
        Algorithm Definition* |
         <algorithm_definition algorithm_link="Algorithm Link"/>
      </security_algorithm>
   </dsl:security_pattern>
Security Pattern Name → String
Encryption Key Definition →
   Key Definition* | <key_definition key_link="Key Link"/>
Decryption Key Definition →
   Key Definition* | <key_definition key_link="Key Link"/>
```

B. Syntax of digital signature definition section

```
Digital Signature Definition →
   <dsl:digital-signature
      name="Signature Name"
      time="When to Put Signature">
      <key_infomation>
        <sign_key>
          Key_definition*|
           <key_definition key_link="Key_Link">
        </sign_key>
        <verify_key>
           Key_definition |
           <key_definition key_link="Key_Link">
        </verify_key>
      </key_infomation>
      Signature Algorithm Definition
        <digest-element>
          Digest Item Lists
        </digest-element>
   </dsl:digital-signature>

Signature Algorithm Definition→
   Integrated Algorithm | Separated Algorithm
Integrated Algorithm→
   <signature_algorithm>
     ( Algorithm Definition*|
        <algorithm_definition algorithm_link="Algorithm Link"/> )
   </signature_algorithm>
Separated Algorithm→
   <security_algorithm>
     ( Algorithm Definition |
       <algorithm_definition algorithm_link="Algorithm Link"/> )
   </security_algorithm>
   <digest_function>
     ( Algorithm Definition |
       <algorithm_definition algorithm_link="Algorithm Link"/> )
   </digest_function>
  <digest-element>
When to Put Signature→BEFORE|AFTER
Digest Item Lists → Digest Item {,Digest Item}
Digest Item → <digest-item Select Expression scope=(element|content)/>
```