A Missing Piece of RSS Technology

Ladda PREECHAVEERAKUL[1], Wichuta KAEWNOPPARAT[2], Dararat SAELEE[1]
Department of Computer Science, Faculty of Science, Prince of Songkla University
Hat Yai, Songkhla, Thailand
{ladda.p, dararat.s}@psu.ac.th[1], kwichuta@bunga.pn.psu.ac.th[2]

In the Information Age, people use RSS (Really Simple Syndication) Technology to help them to easily get the latest contents from websites by accessing only a single website as well as by using mobile devices. Several companies have started to use RSS for distributing their information to customers. However, most contents published via RSS technology are public and not confidential such as credit card information, financial business information etc. Since the RSS technology does not have a mechanism to ensure that the incoming information is really secure, in this paper we have proposed a Secure Information Notifying System with RSS Technology (SInfoNS). We have applied the RSS technology together with the cryptography to make any RSS document become secure before disseminating it to relevant users. The SInfoNS also uses XSL to apply to private information retrieval and XML schema and SchemaPath definitions have been created for validation. The results displayed on a user's mobile device provide users with the latest information. The results of this study confirm that our system will aggregate RSS documents and disseminate information to each user. The SInfoNS enables RSS technology for the use of private information that can be securely distributed.

Keywords: *Cryptography, Really Simple Syndication, RSS, XML Schema, XSL*

1 Introduction

RSS [1] is an XML-based information technology widely used in Web log and news web sites for disseminating the latest news. Several companies have started to use RSS for spreading their information to customers. However, most contents published via RSS technology are public, not confidential. Therefore, a secure RSS mechanism is needed to send private information. Although, XML [2] has an XML encryption [3, 4] for sensitive data, a limitation of syntax in the RSS document does not allow for confidential information.

In this paper, we propose a system that aggregates both general and sensitive information in organizations by using RSS documents and transmits these to relevant users. The system will securely provide incoming information to a user by encrypting some parts of an RSS document, especially the private information. Decryption occurs when a user needs to see his or her private information. The system also uses XSL to apply for private information retrieval and creates a Secure RSS Schema Validator for validation. The results displayed on a user's mobile device, gives the users the latest information.

This paper provides the background and previous research work in Section 2 and Section 3, respectively. In Section 4, a supplement is presented that allows for use of RSS technology for private information. The implementation and study results are shown in Section 5 and conclusions are given in Section 6.

2 Background

2.1 Extensible Markup Language (XML)

Extensible Markup Language or XML [2, 5] is a markup language, developed by the W3C (World Wide Web Consortium), and is used to describe documents and data in a standardized, text-based format. It provides an easy way to manage and share information via standard Internet protocols. XML consists of three core components which are the XML document structure, an XML parser, and the Style Sheet Language.

- The XML document contains elements, attributes, and character data. An element always has a start and end tag: <book>…</book>. Each element may have additional information called attributes that contain values. For example, the book element can have an ISBN attribute:

```
<book ISBN = “974-94136-7-8”>
    Beginning XML
</book>
```

The XML documents can be either well-formed or valid. Well-formed documents have respect to syntactic rules whereas valid documents not only have respect to the syntactic rules, but also conform to validation language or schema language.

- The XML Parser identifies and converts XML elements contained in an XML document. Typically, there are two types of parser: a standard or non-validating parser and a validating parser. The standard parser reads and analyzes a document to ensure that it is well-formed, while the validating parser checks the document to ensure that it is valid. Additionally, programmers can access an XML document from their applications to expose the XML elements as a data structure via Document Object Model (DOM) or Simple API for XML (SAX).
- The Style Sheet Language is used to turn the XML document into its required form. The W3C has published two recommendations for style sheets: Cascading Style Sheets (CSS) and Extensible Stylesheet Language (XSL). XSL [6] is used for advanced styling such as creating table or other documented formats.

Figure 1 shows a workflow of an XML document.

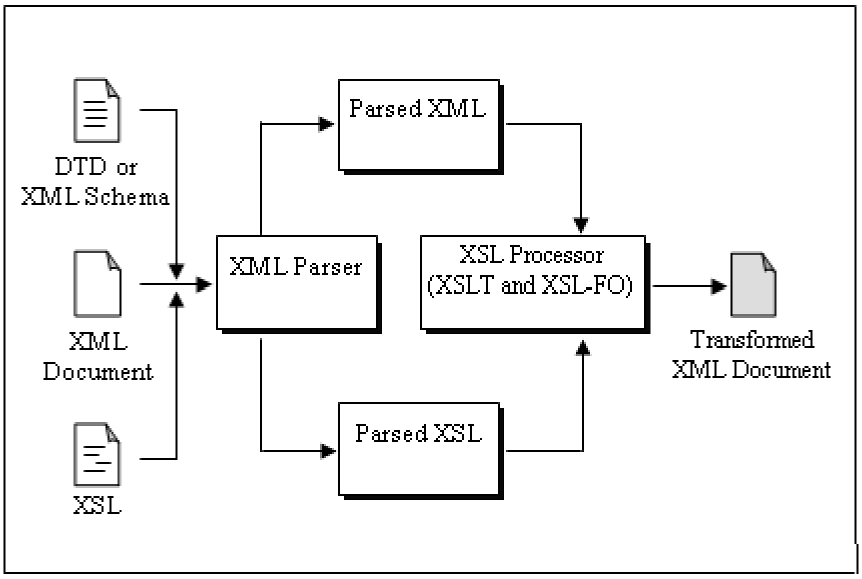


Fig. 1. A workflow of an XML document

2.2 RSS Technology

RSS or Really Simple Syndication is an XML application used to publish frequently updated contents such as blog entries, news headlines, etc. in a standardized format. An RSS architecture consists of three parts: publisher, aggregator, and subscriber. A publisher is a website that provides updated contents in RSS formats. An aggregator is responsible for aggregating contents from multiple websites. A subscriber is a user who reads the updated contents via an RSS reader. Figure 2 illustrates an RSS architecture. The RSS current version is 2.0. Its general structure [7] is shown in figure 3.

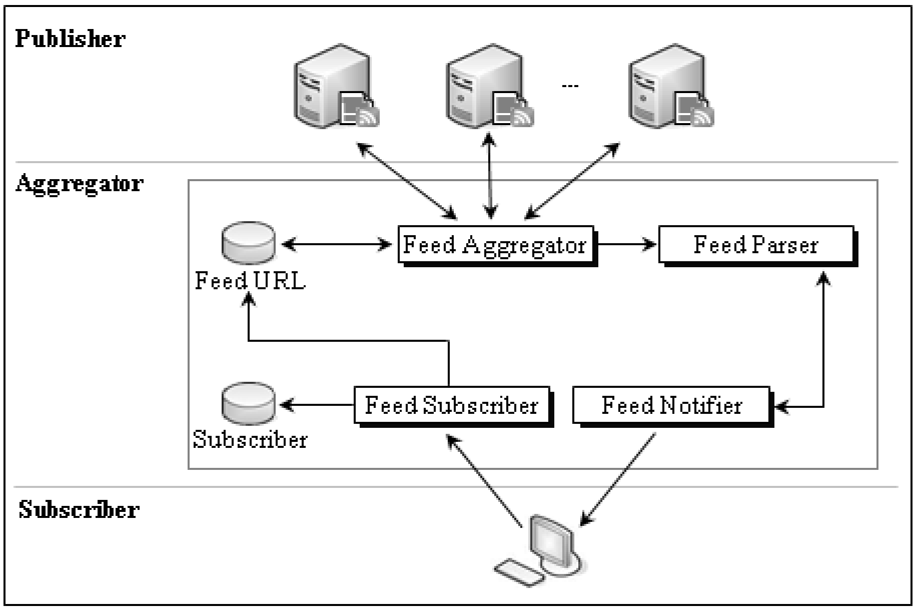


Fig. 2. The RSS Architecture

In figure 3, the first line in the document - the XML declaration - defines the XML version. The second line is the RSS declaration, which identifies an RSS document (in this case, the RSS version 2.0). The next line contains the <channel> used to describe the RSS document. The <channel> element has three required child elements:

1. <title> - Defines the title of the channel
2. <link> - Defines the hyperlink to the channel
3. <description> - Describes the channel

```
<?xml version = “1.0”  ?>
<rss version = “2.0” >
  <channel>
    <title>…</title>
    <link>…</link>
    <description>…</description>
      <item>
        <title>…</title>
        <link>…</link>
        <description>…</description>
        <pubDate>…</pubDate>
        …
      </item>
       …
   </channel>
</rss>
```

Fig. 3. General structure of RSS 2.0 document

Each <channel> can also have one or more <item>. Table 1. shows some RSS item tags.

Table 1. Examples of some RSS item tags

RSS item tag	Description
<title>	Defines the title of the item
<link>	Defines the hyperlink to the item
<description>	Describes the item
<pubDate>	Defines the last publication date for the item
<category>	Defines one or more categories the item belongs to

2.3 Cryptography

The rapid growth and widespread use of electronic data processing through the Internet fuels the need for protecting the information they store, process and transmit. There have been many examples of insufficient security in applications developed for the Internet. Information exchange through the Internet should be focused on the four core principles of security: confidentiality, integrity, authentication and non-repudiation. Thus, the main technique for securing information is to apply cryptography.

Cryptography is a tool to encrypt and decrypt data. Every encryption and decryption process has two aspects: the algorithm and the key. In essence, a sender encrypts his message with the algorithm and the key, obtains the result as a cipher text, and sends it to a recipient. The recipient decrypts an incoming cipher text with the algorithm and the key and restores the original message. The popular techniques in cryptography are symmetric cryptography and asymmetric cryptography. Blowfish and RSA [8] are examples of symmetric and asymmetric cryptography, respectively.

3 Previous Research Work

RSS technology is widely used for news web sites and web log. However, there has been applied research using RSS. For example, Glotzbach et al. [9] applied RSS to provide students with a method of receiving course announcements. Cold [10] used RSS to enhance research methods for students by gleaning current information from online journals, other publications, web logs and other sources without visiting the sites daily. Maruyama et al. [11] proposed a subtree Element-Wise encryption that encrypts some part of the XML document for security. Barrett and Cook [12] presented XML Security using XSLT regarded encryption and decryption as another XML document transformation operation. Chang and Wang [13] devised an application program interface for securing XML documents called Document Security Language (DSL). In addition, Gregorio [14] applied Greasemonkey (a Mozilla Firefox extension) to decrypt RSS documents by writing a symmetric key algorithm. However, this approach also works well with browsers other than Mozilla Firefox.

As mentioned above, we found that RSS technology has been used in several applications. However, most research has not proposed any method to secure RSS documents. A Secure Information Notifying System with RSS Technology for Mobile Users (SInfoNS) has now been proposed [15].

4 A Supplement of RSS Technology for Private Information

The SInfoNS designed in [15] enables each organization to notify both public and private information as well as to receive updated information on mobile devices. This SInfoNS model consists of two main components: SInfoNS Aggregator (SInfoNSA) and SInfoNS Generator (SInfoNSG). However, to make the system work properly, another two components have been proposed. Therefore, the SInfoNS model consists of four main components: SInfoNSA, SInfoNSG, SInfoNS Publisher (SInfoNSP), and SInfoNS Validator (SInfoNSV). Figure 4 illustrates the SInfoNS model.

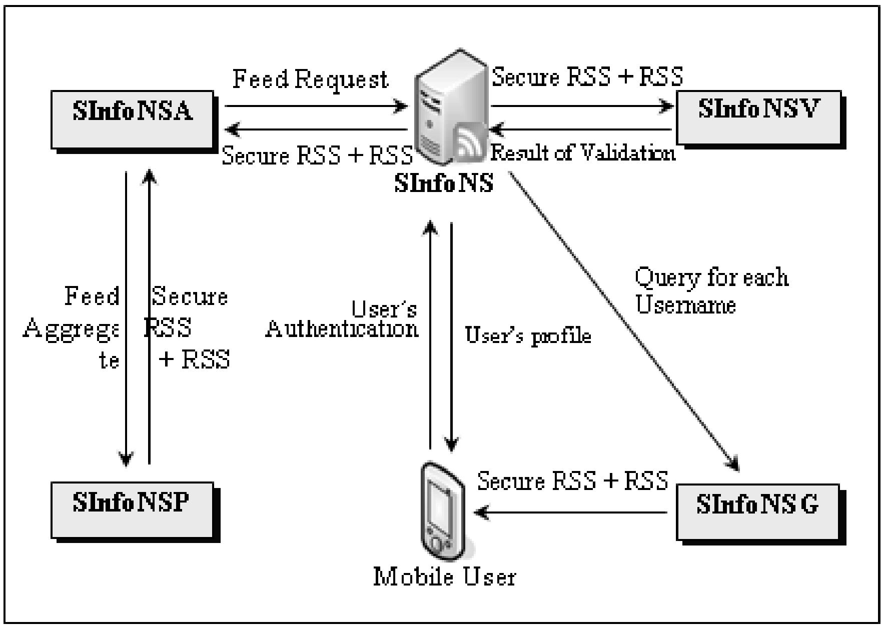


Fig. 4. The SInfoNS Model

4.1 SInfoNS Publisher (SInfoNSP)
The SInfoNSP acts as a publisher that consists of the following steps:
1. Assign a public key to each organization, the organizations will then encrypt their random secret key.
2. Encrypt relevant private information to each subscriber in the RSS format as shown in figure 5 and describe as follows:
The message or private information is encrypted in RSS documents using a random secret key called cipher message.

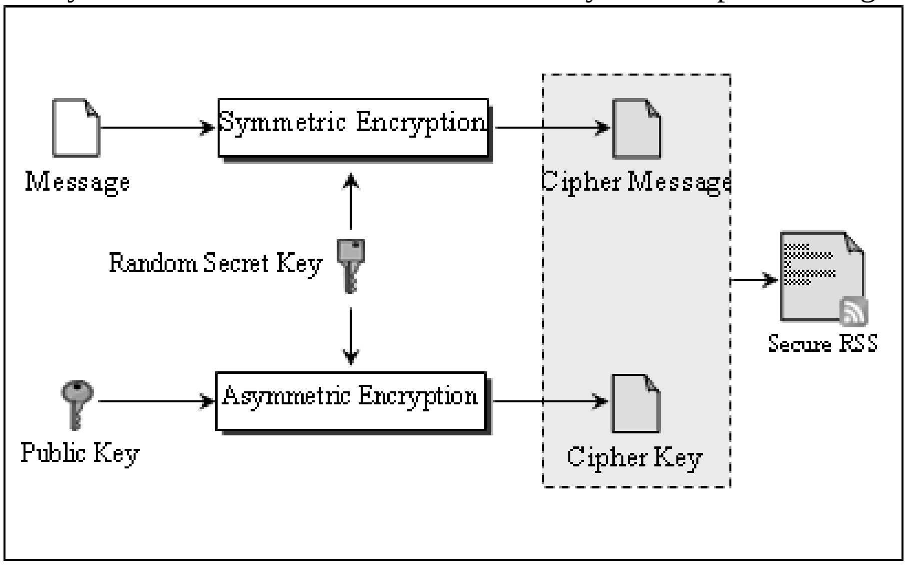


Fig. 5. Encryption process for private information

The random secret key is encrypted using the public key called cipher key.
The cipher message with the cipher key is placed into a <description> element.
3. Create the RSS documents. Three attributes of a <description> element are shown in table 2.
The following examples present how to define them in a <description>.
***Example 1*:** If the information is for public information, the syntax would be:
<description class = "mesg">
Public News</description>
***Example 2*:** If the information is for private information, the syntax would be

<description class = “cipher” username = “John” algorithm = “blowfish”>F663:2e5e </description>

Table 2. Supplemented attributes of a <description>

Attribute	Description
Class	Identify data in a <description> whether “mesg” for public information or “cipher” for private information
username	Name of subscriber
algorithm	The algorithm used to encrypt private information.

Publish the RSS documents

4.2 SInfoNS Aggregator (SInfoNSA)

The SInfoNSA aggregates RSS documents from organizations that may release both public and private information as shown in figure 6, and stores the data to the SInfoNS database. Figure 7 shows a class diagram of the SInfoNS database and the algorithm of SInfoNS Aggregation is illustrated in figure 8.

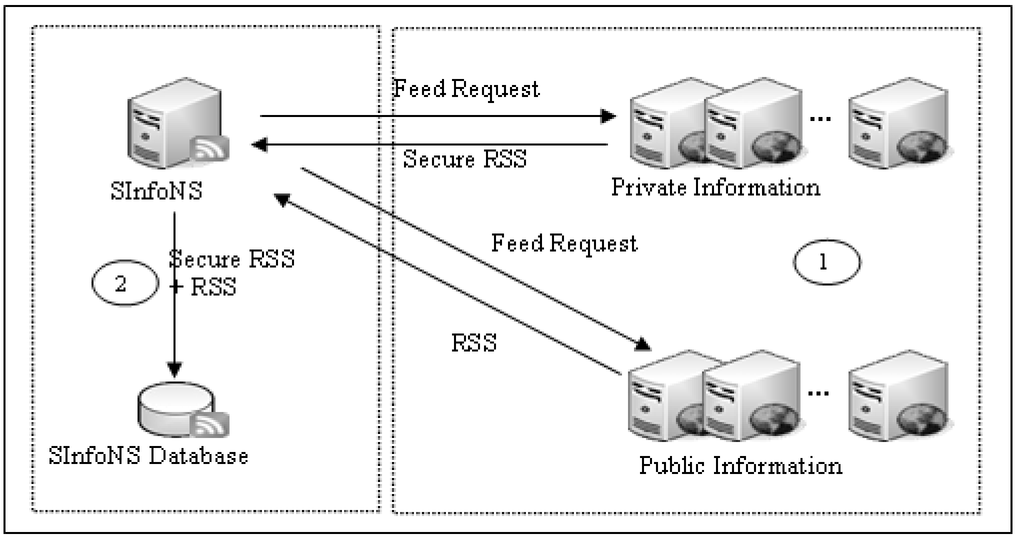


Fig. 6. The SInfoNS Aggregator [15]

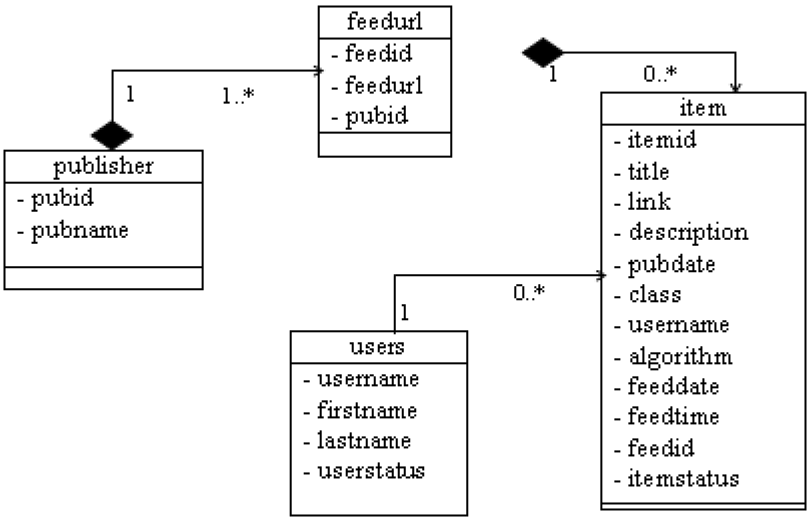


Fig. 7. Class Diagram of SInfoNS Database

```
Method SInfoNSA (feedUrl, PubInfo,
                SecInfo)
 for each feedurl
  Download feed content(PubInfo and
     SecInfo)from feedurl
  for each item in the feed
    Get data in <title>,
        <description>,<link>,
        class, username, algorithm
    Store in SInfoNS database
  end for
 end for
end method
```

Fig. 8. The SInfoNSA Algorithm

4.3 SInfoNS Generator (SInfoNSG)

The SInfoNSG generates an RSS document for each user. The SInfoNSG will activate when a user accesses it with his or her username and password given by the organization. Then, SInfoNS creates a user's profile on his mobile device, and generates an RSS document for that user as shown in figure 9. The process to obtain the RSS document runs as follows:

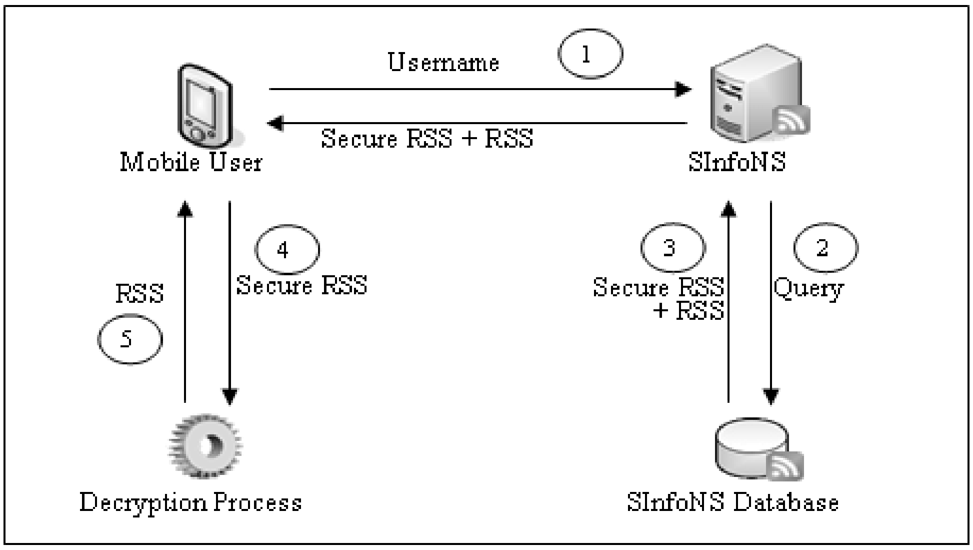


Fig. 9. The SInfoNS Generator [15]

1. The SInfoNS verifies the username from the client's profile on the user's mobile device for authentication.
2. The SInfoNS queries the SInfoNS database for retrieving a user's latest RSS document.
3. The SInfoNS generates the user's RSS documents with public and private information and sends it to that user.
4. In case of private information, the decryption process in figure 9 is invoked.
5. When finishing the decryption of the information, the result will be displayed on the user's mobile device.

The SInfoNSG algorithm is explained in figure 10.

```
Method SInfoNSG (username)
 Search username in SInfoNS
  database to generate RSS document
 Send RSS document to user
 if (class = "cipher")
  Call decryption function (cipher
    key, cipher message, algorithm)
 end if
 Show content to user
end method
```

Fig. 10. The SInfoNSG Algorithm

4.4 SInfoNS Validator (SInfoNSV)

The SInfoNSV is devised to validate the RSS document according to a set of structural and content rules expressed in secure RSS schema based on XML schema [16]. However, one of the XML schema limitations is co-constraints as reported in [17]. Co-constraints or co-occurrence constraints mean one attribute is present if and only if the other attribute is (or is not) present in the same element. We applied SchemaPath [17], a light extension of XML schema to handle conditional constraints on our RSS documents. Fig. 11 illustrates a process to validate the RSS document with secure RSS schema. The process in Fig. 11 runs as follows:

1. A secure RSS schema written in SchemaPath (SP) format is transformed to XML schema (secure RSS schema′) using T′.
2. An MT, a metadata, is created to verify a content rule expressed in the secure RSS schema and generates a template (T″) for each alternative of every conditional declaration.
3. A secure RSS document is transformed to a secure RSS′ using T″.
4. The secure RSS′ is verified with the secure RSS schema′ to ensure that it is valid or not using the XML schema validator.

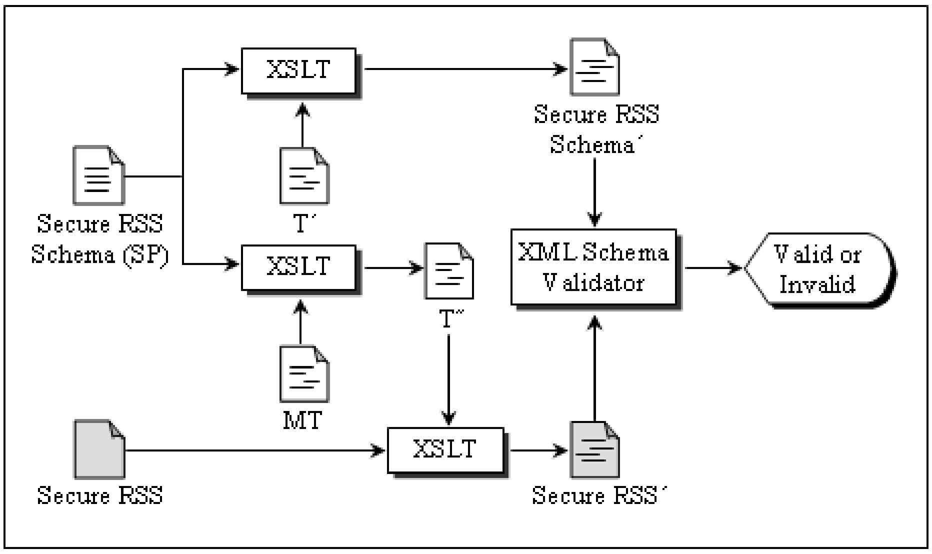


Fig. 11. Process to validate RSS documents with Secure RSS Schema

Since the RSS schema [18] does not support the attribute that we created in Section 4.1, then we create a format and data type added on a <description> in the RSS schema as shown in figure 12.

```
<xsd:element name="description" >
   <xsd:alt cond="@class='mesg'"
    type="rssMesgType" />
   <xsd:alt cond="@class='cipher'"
    type="rssCipherType"/>
</xsd:element>
```

Fig. 12. SchemaPath Syntax for type declaration in <description>

A supplemental syntax of a <description> in RSS is based on a simple Extended Backus-Naur Form (EBNF) notation in [19]. The formal grammar of a supplemental syntax is shown in figure 13.

```
DescItemElem ::=   DescItemSTag DescItemType DescItemETag
DescItemSTag ::=   '<description' S
DescItemETag ::=   '</description' S? '>'
DescItemType ::=   MesgAttr MesgContent
             | CipherAttr CipherContent
MesgAttr   ::=     'class' Eq MesgValue S? '>'
MesgValue  ::=     '"' mesg '"' | "'" mesg "'"
MesgContent  ::=   Char (Char)*
UserNameSChar      ::=   [a-z] | [A-Z] | [0-9] | "_"
             | "-"
UserNameChar ::=   UserNameSChar | "."
UserNameValue      ::=   '"' UserNameSChar (UserNameChar)* '"'
             | "'" UserNameSChar (UserNameChar)* "'"
CipherAttr::=      'class' Eq CipherValue S 'username' Eq
             UserNameValue S
             'algorithm' Eq AttValue S? '>'
CipherValue  ::=   '"' cipher '"'
             | "'" cipher "'"
CipherSChar  ::=   [a-z] | [A-Z] | [0-9] | "+"
             | "/"
CipherEChar  ::=   [a-z] | [A-Z] | [0-9] | "_"
             | "-" | "="
CipherChar::=      CipherSChar | "=" | #x20
CipherContent      ::=   CipherSChar (CipherChar)* CipherEChar
             ':' CipherSChar (CipherChar)* CipherEChar
```

Fig. 13. Supplemental syntax of a <description> using EBNF notation

The examples in figures 14 and 15 show the derivations according to the supplemental syntax.

5. Implementation

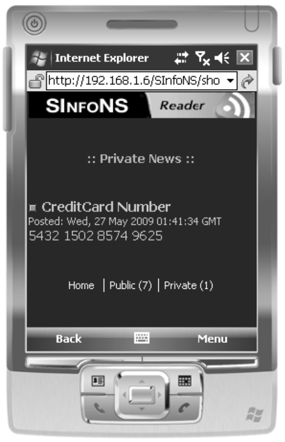
The system has been developed for the use of PHP and JavaScript. The result is shown in Internet Explorer Mobile on Windows Mobile 6.1 Emulator. The main function of SInfoNS can be divided into 4 modules: Administration, Publication, Subscription, and Secure RSS Schema Validation.

```
Example element: <description class="mesg">Public News</description>

Derivation:
DescItemElem => DescItemSTag DescItemType DescItemETag
            => <description DescItemType DescItemETag
            => <description MesgAttr MesgContent DescItemETag
            => <description class Eq MesgValue> MesgContent DescItemETag
            => <description class = MesgValue> MesgContent DescItemETag
            => <description class = "mesg"> MesgContent DescItemETag
            => <description class = "mesg"> Char (Char)* DescItemETag
            => <description class = "mesg"> P (Char)* DescItemETag
            => <description class = "mesg"> Public News DescItemETag
            => <description class = "mesg"> Public News </description>
```

Fig. 14. A derivation of public information according to the supplemental syntax

```
Example element: <description class="cipher" username="John"
                    algorithm="blowfish">F633:2e5e</description>

Derivation:
DescItemElem =>  DescItemSTag DescItemType DescItemETag
             =>  <description DescItemType DescItemETag
             =>  <description CipherAttr CipherContent DescItemETag
             =>  <description class Eq CipherValue username Eq UserNameValue
                 algorithm Eq AttValue> CipherContent DescItemETag
             =>  <description class = CipherValue username Eq UserNameValue
                 algorithm Eq AttValue> CipherContent DescItemETag
             =>  <description class = "cipher" username Eq UserNameValue
                 algorithm Eq AttValue> CipherContent DescItemETag
             =>  <description class = "cipher" username = UserNameValue algorithm
                 Eq AttValue> CipherContent DescItemETag
             =>  <description class = "cipher" username ="UserNameSChar
                 (UserNameChar)*" algorithm Eq AttValue> CipherContent
                 DescItemETag
             =>  <description class = "cipher" username ="J (UserNameChar)*"
                 algorithm Eq AttValue> CipherContent DescItemETag
             =>  <description class = "cipher" username ="John" algorithm Eq
                 AttValue> CipherContent DescItemETag
             =>  <description class = "cipher" username = "John" algorithm =
                 AttValue> CipherContent DescItemETag
             =>  <description class = "cipher" username = "John" algorithm =
                 "blowfish"> CipherContent DescItemETag
             =>  <description class = "cipher" username = "John" algorithm =
                 "blowfish"> CipherSChar (CipherChar)* CipherEChar : CipherSChar
                 (CipherChar)* CipherEChar DescItemETag
             =>  <description class = "cipher" username = "John" algorithm =
                 "blowfish">F (CipherChar)* CipherEChar : CipherSChar (CipherChar)*
                 CipherEChar DescItemETag
             =>  <description class = "cipher" username = "John" algorithm =
                 "blowfish">F66 CipherEChar : cipherSChar (CipherChar)*
                 CipherEChar DescItemETag
             =>  <description class = "cipher" username = "John" algorithm =
                 "blowfish">F663: cipherSChar (CipherChar)* CipherEChar
                 DescItemETag
             =>  <description class = "cipher" username = "John" algorithm =
                 "blowfish">F663:2 (CipherChar)* CipherEChar DescItemETag
             =>  <description class = "cipher" username = "John" algorithm =
                 "blowfish">F663:2e5 CipherEChar DescItemETag
             =>  <description class = "cipher" username = "John" algorithm =
                 "blowfish">F663:2e5e DescItemETag
             =>  <description class= "cipher" username= "John"
                 algorithm="blowfish">F663:2e5e</description>
```

Fig. 15. A derivation of private information according to the supplemental syntax

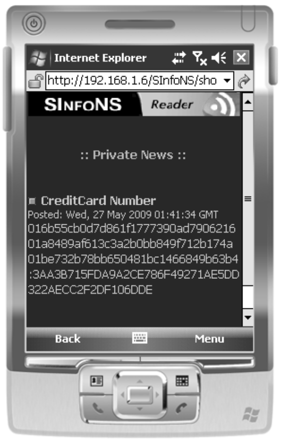


:: Private News ::

CreditCard Number
Posted: Wed, 27 May 2009 01:41:34 GMT
5432 1502 8574 9625

Home | Public (7) | Private (1)

a)

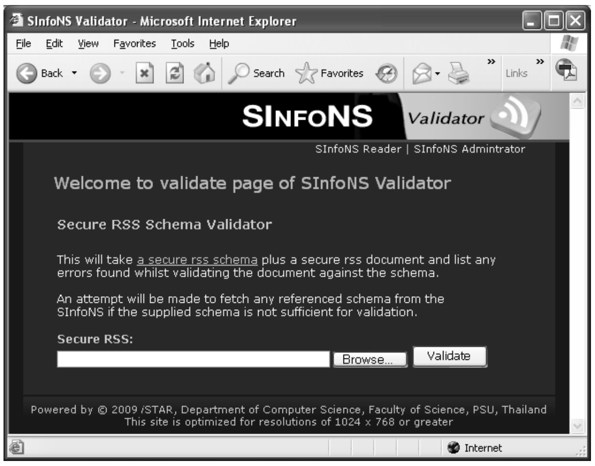


:: Private News ::

CreditCard Number
Posted: Wed, 27 May 2009 01:41:34 GMT
016b55cb0d7d861f1777390ad7906216
01a8489af613c3a2b0bb849f712b174a
01be732b78bb650481bc1466849b63b4
:3AA3B715FDA9A2CE786F49271AE5DD
322AECC2F2DF106DDE

b)

Fig. 16. Private information with and without decryption


SInfoNS Validator - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Links
SInfoNS Validator
SInfoNS Reader | SInfoNS Admintrator
Browse...
Validate
Internet

Welcome to validate page of SInfoNS Validator

Secure RSS Schema Validator

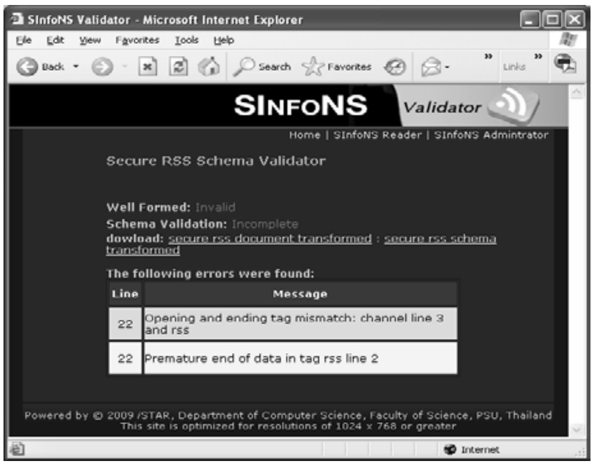
This will take a secure rss schema plus a secure rss document and list any errors found whilst validating the document against the schema.

An attempt will be made to fetch any referenced schema from the SInfoNS if the supplied schema is not sufficient for validation.

Secure RSS:

Powered by © 2009 /STAR, Department of Computer Science, Faculty of Science, PSU, Thailand
This site is optimized for resolutions of 1024 x 768 or greater

a)


SInfoNS Validator - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Links
SInfoNS Validator
Home | SInfoNS Reader | SInfoNS Admintrator
Internet

Secure RSS Schema Validator

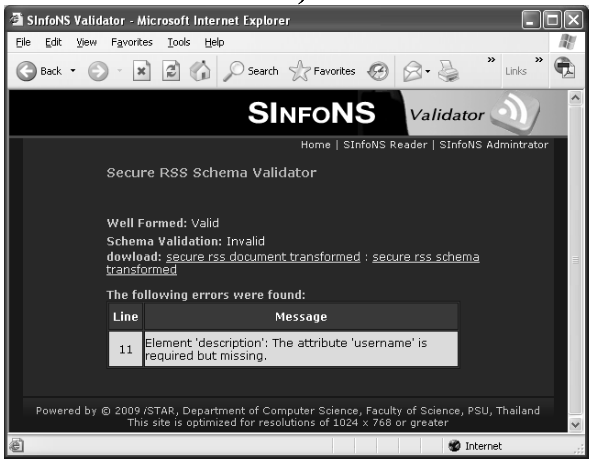
Well Formed: Invalid
Schema Validation: Incomplete
dowload: secure rss document transformed : secure rss schema transformed

The following errors were found:

Line	Message
22	Opening and ending tag mismatch: channel line 3 and rss
22	Premature end of data in tag rss line 2

Powered by © 2009 /STAR, Department of Computer Science, Faculty of Science, PSU, Thailand
This site is optimized for resolutions of 1024 x 768 or greater

b)


SInfoNS Validator - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Links
SInfoNS Validator
Home | SInfoNS Reader | SInfoNS Admintrator
Internet

Secure RSS Schema Validator

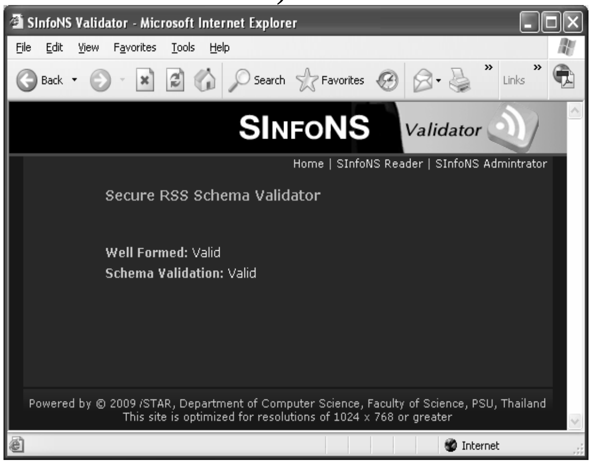
Well Formed: Valid
Schema Validation: Invalid
dowload: secure rss document transformed : secure rss schema transformed

The following errors were found:

Line	Message
11	Element 'description': The attribute 'username' is required but missing.

Powered by © 2009 /STAR, Department of Computer Science, Faculty of Science, PSU, Thailand
This site is optimized for resolutions of 1024 x 768 or greater

c)


SInfoNS Validator - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Links
SInfoNS Validator
Home | SInfoNS Reader | SInfoNS Admintrator
Internet

Secure RSS Schema Validator

Well Formed: Valid
Schema Validation: Valid

Powered by © 2009 /STAR, Department of Computer Science, Faculty of Science, PSU, Thailand
This site is optimized for resolutions of 1024 x 768 or greater

d)

Fig. 17. Check validity of each secure RSS document
a) SInfoNS Validator
b) Not Well-formed
c) Well-formed but Invalid
d) Well-formed and Valid

An administration module takes care of data such as RSS documents, locations to feed (feed URL), and subscriber information.

A publication module publishes RSS documents for both public and private information.

A subscription module creates and displays the RSS documents relevant to each subscriber.
A secure RSS schema validation module is used to check the validity of each secure RSS document.

5.1 Study Results

We implemented the SInfoNS and generated a web site to notify both public and private information such as credit card information to users. The results in Fig. 16 a) and b) show private information with and without decryption in a mobile device. The Secure RSS Schema Validator shown in Fig. 17 a) is created to check the validity of each secure RSS document. Fig. 17 b) shows that the secure RSS document is not well-formed. Fig. 17 c) shows that the secure RSS document is well-formed but invalid, whereas Fig. 17 d) shows that the secure RSS document is well-formed and valid.

6. Conclusions

The proposed SInfoNS helps organizations to ensure that information applicable to each user, especially sensitive information, will be secure. We applied the RSS technology together with the cryptography algorithm to make any RSS document secure before disseminating it to relevant users. The results displayed on a user's mobile device give users, the latest information.
The results of this study confirm that our system is able to aggregate RSS documents and disseminate information to each user. Additionally, the Secure RSS Schema Validator we have created checks the validity of each secure RSS document with respect to syntactic rules and conforms to the validation language.
The SInfoNS meets the requirements for securely distributing private information using RSS technology with cryptography algorithm. The system is designed to support TCP/IP-based users, thus not only for mobile users, but also Internet-based users.

References

[1] E. Finkelstein, *Syndicating Web Sites with RSS Feeds for Dummies*, Wiley Publishing, Inc., Hoboken, NJ, 2005.

[2] W3C, *Extensible Markup Language (XML)*, July 2008. Available at: http://www.w3.org/XML/ [May 28, 2008].

[3] W3C, *XML Encryption Syntax and Processing*, December 2002. Available at: http://www.w3.org/TR/xmlenc-core/ [May 28, 2008].

[4] W3C, *XML Signature Syntax and Processing (Second Edition)*, June 2008. Available at: http://www.w3.org/TR/xmldsig-core/ [May 28, 2008].

[5] B. Benz and J. Durant, *XML Programming Bible*, Wiley Publishing, Inc., Indianapolis, Indiana, 2003.

[6] L. Dykes and E. Tittel, *XML For Dummies 4th Edition*, Wiley Publishing, Inc., Hoboken, NJ, 2005.

[7] W3Schools, *RSS Tutorial*, Available at: http://www. w3schools.com/rss/ [April 20, 2008].

[8] A. Kahate, *Cryptography and Network Security*, McGraw-Hill Publishing Company, 2003.

[9] R. J. Glotzbach, J. L. Mohler and J. E. Radwan, "RSS as a Course Information Delivery Method", *International Conference on Computer Graphics and Interactive Techniques*, San Diego, California, August 05-09, 2007.

[10] S. J. Cold, "Using Really Simple Syndication (RSS) to Enhance Student Research", *ACM SIGITE Newsletter*, Vol.3, No.1, January, 2006, pp. 6-9.

[11] H. Maruyama and T. Imamura, "Element-wise XML Encryption", April 2000. Available at: http://lists.w3.org/Archives/Public/xmlencryption/2000Apr/att-0005/01-xmlenc [January 5, 2008].

[12] R. G. Bartlett and M. W. Cook, "XML Security Using XSLT," The *36th Hawaii International Conference on System Sciences (HICSS'03)*, IEEE, 2002.

[13] T. Chang and G. Hwang, "The design and implementation of an application program interface for securing XML documents", *The Journal Systems and Software*, August, 2007, pp. 1362–1374.

[14] J.Gregorio, *Secure RSS Syndication*, July 2005. Available at: http://www.xml.com/pub/a/2005/07/13/secure-rss.html [May 28, 2008].

[15] L. Preechaveerakul and W. Kaewnopparat, "A Novel Approach: Secure Information Notifying System using RSS Technology," *2009 International Conference on Future Networks (ICFN 2009)*, Bangkok, Thailand, Mar 7-9, 2009, pp. 95-99.

[16] C. M. Sperberg-McQueen and H. Thompson, *XML Schema*, April 2000, Available at: http://www.w3.org/XML/Schema [October 10, 2008].

[17] P. Marinelli, C. S. Coen, and F. Vitali, "SchemaPath, a Minimal Extension to XML Schema for Conditional Constraints," *The 13th International conference on WWW*, New York, NY USA, May 17-22, 2004, pp. 164-174.

[18] J. Thelin, *RSS 2.0 Schema*, Available at: http://www.thearchitect.co.uk/schemas/rss-2_0.xsd [December 10, 2008].

[19] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler and F. Yergeau, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, November 2008, Available at: http://www.w3.org/TR/REC-xml/ [January 12, 2009].

Ladda Preechaveerakul received the Ph.D. degree in Computer Science from Chulalongkorn University, Thailand. Currently, she is a lecturer in Department of Computer Science, Faculty of Science, Prince of Songkla University, Thailand. Her research interests include Internet Computing, Mobile Computing, and Information Security.

Wichuta Kaewnopparat received the M.Sc. degree in Computer Science from Prince of Songkla University, Thailand. Currently, she is a programmer in Computer Center, Pattani Campus, Prince of Songkla University. Her research interests include Web Intelligent, Information Retrieval.

Dararat SaeLee received the M.Sc. degree in Computer Science from Chulalongkorn University, Thailand. She currently is a lecturer in Department of Computer Science, Faculty of Science, Prince of Songkla University, Thailand. Her research interests include Grid Computing, Operating System, and Information Retrieval.