ISSN NO. 2320-5407

Journal homepage: http://www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH

RESEARCH ARTICLE

An alternative approach for XML messaging

***P P Abdul Haleem[1], and M P Sebastisan[2]**

1. Assistant Professor, Department of Computer Science, Farook College, Calicut, Kerala, India
2. Professor (Information Technology and Systems), Information Technology, Indian Institute of Management Kozhikode, Kerala, India.

Manuscript Info

Manuscript History:

Received: 05 December 2013
Final Accepted: 23 December 2013
Published Online: January 2014

Key words:
XML Messaging,
Data Formatting Schemes,
XML,
YAML,
Schema,
Message Level Security,
ECC,
Rewriting Attacks

Abstract

Messaging has become a vital part of communication between applications, especially with the wide popularity of web service standards. Data format used to represent information is an important concern in messaging. eXtensible Markup Language (XML), being the default and universal format for structured data, is currently being used as the data format for messaging as well. Traditional networking arena is undergoing a paradigm shift due to the proliferation of wireless mobile devices/networks. When there are devices with constrained resources, the traditional networks have to adopt to the features of the resource constrained networks (RCNs). Demand for resource conservation, which cannot be met with the XML format because of its inherent limitations when used in RCNs give rise to the research into alternative mechanisms. In this paper a format called as Thinned YAML (TYAML) which causes reduction in verbosity based on the YAML Ain't Markup Language (YAML) format is presented. The TYAML format is equipped with message level security also. It is observed that the secure versions of TYAML maintain its less verbose nature in comparison with the XML formats. As security protocols consume considerable resources, the application of lightweight algorithms is of particular interest in RCNs. So TYAML specifications are enhanced to make them compatible with Elliptic Curve Cryptography (ECC) based algorithms. To make TYAML tamper proof while in transit (rewriting attack), a layered mechanism is also proposed. Experiments show clear performance advantages for the proposed structures over the existing formats.

1 Introduction

Data representation and exchange among heterogeneous types of applications and devices is a major concern for technology developers and user communities. XML is widely used now as a medium for this purpose. It is a meta language that allows the developers to create their own markup languages (H. Li 2000). In the traditional wired world, XML is the "lingua franca" for data representation and exchange. But the penetration of wireless applications and devices into the networking arena raises new concerns about the use of XML, especially in web transactions, in the constrained wireless mobile environment. This is due to the inherent limitations of wireless mobile networks and the constrained mobile wireless devices.

1.2 XML Messaging

The meaning of "Messaging" as intended in this paper can be described as a set of data sent or received between applications. Needless to say, these data sets should have header information that describes the "semantics" of the data being transmitted. With the advent of web services and its wide spread use, messaging is a vital part of

communication among applications. It can be observed that the heart of messaging is the format used to represent the information.

XML is widely used as a medium for representing and exchanging data among heterogeneous types of applications and devices. It is particularly useful in exchanging information due to its flexibility and generality over the binary and proprietary formats, developer productivity, provisions for interoperability between applications and widespread tool availability (W. S. Hanslo *et al*., 2003). It is also being used in diversified roles including the generation of programmable web services (J. Kangasharju *et. al* 2007, H. H. Lee *et. al*, 2010) and as a message syntax format in Simple Object Access Protocol (SOAP). It is a markup language for describing documents containing structured information. The most interesting design goals of XML include: (i) straightforward use over the Internet, (ii) support of a wide variety of applications, (iii) human readability, and (iv) easiness to create documents.

Despite its universal acceptance in multiple applications, XML has a constraint which hinders its use in the wireless mobile world - its verbose nature. This verbosity is mainly due to the abundant use of tags. The additional characters which are needed for structuring the data as per the XML specifications lead to an "inflation problem". The estimates in (H. R. Elliotte 2003, J. Kangasharju *et. al* 2005, J. Kangasharju *et. al* 2008) reveals that a XML document can be 3 to 20 times large as compared to its binary or text file representation and so a 1 GB of a database may consume about 20GB when represented in XML. Large files result in slow processing, which needs more memory and increased transmission costs. Increased verbosity also necessitates compression/binary encoding before transmission of data. In addition to this, XML does not have an intrinsic data type support (i.e., there is no specific notion of types such as "integer", "string", "Boolean", "date", and so on). Even though XML is very rich and agile, many web based applications need a simpler messaging than the XML.

This requirement for simple format may seriously hinder the future use of XML (especially in RCNs) for exchanging of data, because of the fact that the data size grows much faster than the communication bandwidth (W. Ng et. al 2006). RCNs include ad hoc networks, sensor networks, mobile networks, and pervasive systems

1.3 Resource Constraint Networks (RCN)

The abundant growth of Internet services and the increased number of wireless mobile users resulted in the penetration of wireless mobile devices into the realm of wired networks. The penetration of constrained wireless mobile devices into the traditional networking arena has reached to the extent of even easily surpassing the number of personal computers. Wireless mobile devices (with limited resources such as battery power, memory, processing power and input and screen) and wireless networks (with constraints such as unreliable channels, low bandwidth, increased latency, increased rate of retransmission of lost packets and weak security features) together impose a number of limitations. In addition, users expect to access much wider range of applications in wireless mobile devices than in the conventional devices.

Fewer numbers of characters for data representation is well suited for an environment with limited input and screen. The largest chunk of power consumed in a wireless mobile device is during the transmission and reception of the messages (M. P. Michael 2005). Packet losses and subsequent retransmissions through the relatively unreliable channels result in further resource crunch. Security protocols for the constrained wireless environment also prefer reduced size of data and less number of packets to be transmitted/received (R. Potlapally *et. al* 2006). All of these constraints cumulatively pointing to the fact that the data size to be transmitted must be as compact as possible.

One of the main concerns with wireless mobile devices is not the slow CPU speed or limited storage capabilities, but the energy (G. Q. Gu *et. al* 2008). As a phone needs to be mobile, its power source must be portable battery that can only contain limited amount of energy backup. Interestingly, unlike the CPU speed and storage capacity, the battery energy density does not follow an exponential growth curve (Paradiso *et. al* 2005). The power limitation is mainly due to the space constraints in accommodating the battery within a mobile device (M. P. Michael 2005).

There is an argument that the designers of wireless mobile devices are competing to pack more facilities into the devices and hence there is no scope for energy conserving measures, especially for messaging. But in reality, these features are used only by the top 10% of the customers. At the same time, low-cost devices are steadily increasing their market share, especially in developing countries. The projected shipments of ultra-low-cost mobile phones is expected to grow to more than 360 million by 2015, representing a 2010-2015 compound annual growth rate

(CAGR) of over 22%, while shipments of low-cost handsets will reach 249 million by the same time period (L. Snol 2010). Mobile phone penetration in developing countries now stands at 68%, where as in developed countries it is reaching saturation levels with on average 116 subscriptions per 100 inhabitants (C. Kounavis 2010). An estimated 80% of the world's population could use a mobile phone, thanks to broad wireless network coverage, but only about 20% actually subscribe to a wireless service, mainly because mobile phones are too expensive, according to the Global System for Mobile communication (GSM) Association. It is also interesting to note that the biggest penetration of wireless networking and devices are from these emerging markets as opposed to developed nations (A. Tella *et. al* 2010, J. C. Aker *et. al* 2010). These facts reveal the importance of low cost wireless mobile devices that are low on features and do not come with many features other than SMS support. Low-cost devices are steadily increasing their market share, especially in developing countries.

The relationship between lack of infrastructure such as power availability, per capital income and demand for low cost entry level devices, especially in emerging markets indicates wide scope for energy conserving applications. The biggest consumer of energy in a mobile device is the RF part of cellular engine, which is responsible for transmission and reception of messages (M. P. Michael 2005). The trend from voice to (mobile) data applications is reflected in the growing number of text messages sent in 2010, which are close to 200000 text messages per second (C. Kounavis 2010). Hence it is clear that both for the entry level devices and powerful new generation devices, a less verbose, energy conserving data formatting scheme is still relevant.

The protocols used for wireless networking such as Bluetooth and Wireless Local Area Network (WLAN) also consume battery power considerably (H. Artail *et. al*. 2009). Another point of concern is the possibility of power drain due to packet losses in the wireless mobile environment. The packet loss in WLAN is estimated to be 30%, which results in heavy drain of battery power (M. P. Michael 2005). The Orthogonal Frequency-Division Multiplexing (OFDM) technology shipped with the IEEE 802.11a and 802.11g standards is known to have high peak-to-average-power ratio (PAPR), expensive transmitter circuitry requirement, and poor power efficiency (OFDM 2008).

The security protocols in the constrained wireless mobile environment must conserve energy to the maximum possible extent. The power drain during a secure wireless session depend upon the number of packets to be transmitted or received and the size of the messages required for establishing a session. Analysis on the energy consumption for processing the various cryptographic algorithms underlines the need to have messages with reduced size and small key sizes (R. Potlapally *et. al* 2006).

Considering the huge verbosity (a difference of about one-third) of an XML-ized document compared to the same information listed in a standard document format (W. Ng et. al 2006), measures for size reduction of messages to achieve energy conservation is very important in constrained wireless mobile devices. Reduction of message size results in reduction in the number of packet transmissions, which in turn reduces the data transmission cost and storage space (Y.X. Lai *et. al* 2008). It is interesting to note that, in the wireless world, optimizing the message size is of primary importance than the processing efficiency gains through an alternative format (J. Kangasharju *et. al* 2008). Hence it is clear that the verbosity of the data format is the factor which directly influences many other constraints, including the energy consumption, particularly in wireless mobile devices and networks.

Remaining portions of the manuscript is organized as follows: Section 2 describes the literature review. Sections 3, 4, 5, and 6 discuss the research contributions. Section 7 discusses the results obtained. Section 8 concludes the paper.

2 Literature Review

Since the format to represent data is the heart of a messaging system, this section makes a survey of the state of the art in data formatting schemes, security specifications and measures for the trusted transmission of messages in RCNs.

2.1 Data Formatting Schemes

Design of XML alternative messaging formats is of interest to the researchers and many schemes are already available in the literature. These schemes can be categorized broadly into (i) standard compression techniques, (ii)

XML alternative binary formats, and (iii) XML alternative non-binary formats. It is worth mentioning that all these methods have yielded good formats for the wired networks.

Since the objective of this paper is to propose a lightweight data formatting scheme which is human readable (as a viable alternative to XML), the focus is on XML alternative non-binary formats. The other categories do not provide self descriptiveness and human readability and hence is not being discussed here.

2.1.1 Non-binary Formats

Several alternative serialization formats are proposed to reduce the verbosity of XML which include YAML (O. Ben-Kiki *et. al* 2010), JavaScript Object Notation (JSON) (D. Crockford 2006), Open Node Syntax (ONX) (S. Jacobs 2005), and Simple Outline XML (SOX) (SOX 2002). These formats suggest methods for conserving memory. They also tend to maintain the agility as in XML. A common feature of these formats is the omission of end tags, which reduces the verbosity and redundancy considerably.

ONX (pronounced as "onyx") is a data-oriented markup language. Data is represented in a compact way, making use of "valueNodes" (for representing the attribute value pair). ONX retains the end tag concept in XML, but care is taken to see that the overall verbosity is less than that of XML. Despite the advantages of being generic and compact, ONX lacks many features of XML which include (i) facility to include comments, (ii) support for Unicode/ISO character sets, and (iii) support for a schema (S. Jacobs 2005).

JSON is a compact, text based, programming language model data interchange format. It is a subset of the JavaScript language. Being a programming language model, JSON is neither a document format, nor a markup language and not even a general serialization format. It is lightweight, human readable, and supports Unicode. The drawbacks of this format include (D. Crockford 2006), (i) poor readability for large documents, (ii) no namespaces, (iii) not extensible, (iv) fear of a latent command-injection security hole, as JSON itself is a script language, and (v) only plain text data can be represented.

SOX is another alternative to XML. SOX supports XML features like elements, attributes, and text. Other parts such as processing instructions, comments, and entities of XML are not supported by SOX as on date (SOX 2002).

YAML (O. Ben-Kiki *et. al* 2010) is data serialization language which is human readable and lightweight. The default data serialization method adopted in YAML is similar to that of XML. However, special care is taken by the developers of YAML to reduce its overall verbosity. Important features of YAML include (i) portability between programming languages, (ii) matching with the native data structures of agile languages, (iii) expressiveness, and (iv) extensibility. It supports the essential features of XML such as human readability, schema awareness, namespaces and extensibility.

Among the above standards, YAML is the best candidate for a base format due to the following reasons (O. Ben-Kiki *et. al* 2010, K. G. Clark 2002, YAML 2007): (i) its versatile block-indent syntax allows formatting of structured data in a human readable style, (ii) it uses the least number of bytes among the existing standards, (iii) YAML is a superset of JSON, (iv) it offers extensible data types beyond the primitives (i.e., beyond strings, floats, ints and bools) which can include class-type declarations or Unicode types, (v) it is resistant to delimiter collision as it relies on outline indentation for structure, (vi) YAML documents allow every embedding document or other types of documents, (vii) it handles indents as small as a single space, and this may offer better compression than markup languages, (viii) it offers a simple relational scheme that allows repeats of identical data to be referenced from two or more points in the tree rather than entered redundantly at those points, and (ix) it lacks an associated command language which is seen as a relative security benefit as parsers at least should be safe to apply to tainted data without the fear of a latent command-injection security hole.

YAML has additional features in comparison with the other non-binary formats, which include (i) provision for comments, (ii) Unicode/ISO character sets, (iii) schema, (iv) namespaces, and (v) facility to include non textual data like images.

2.2. Schema Awareness

Schema awareness is one of the most desirable features of a data formatting scheme. A schema is a model for describing the structure of information. It is a term borrowed from the database world to describe the structure of data in relational tables. In the context of XML, a schema describes a model for a whole class of documents. The model describes the possible arrangement of tags and text in a valid document. A schema might also be viewed as an agreement on a common vocabulary for a particular application that involves exchanging documents.

XML inherits Document Type Definitions (DTDs) from SGML. DTDs are the schema mechanism for Standard Generalized Markup Language (SGML). DTDs are still in use to define content models. But they are not preferred due to their limitations (N. Walsh 1999) such as (i) written in a non-XML syntax, (ii) no support for namespaces, and (iii) offer extremely limited data-typing. In spite of these limitations, DTDs are well understood by a large community of SGML and XML programmers and consultants.

The XML Schema has the following advantages in comparison with the other schemas: (i) its recommendation acknowledges the influence of DCD, DDML, SOX, XML-Data, and XDR in its list of references (appears to have picked up bits and pieces from each of these proposals), (ii) it is supported by the two main contributors (Microsoft for XDR and Commerce One for SOX) for their new developments, (iii) it has a richer data typing system, and (iv) it is a strongly typed schema language which ensures that there is no ambiguity in the determination of the data types. Hence XML Schema is considered as the benchmark for further discussion and comparison.

RELAX NG is an OASIS recommendation. It is observed that XML Schema has the following advantages over RELAX NG: (i) no intrinsic support for data types, (ii) wider vendor support and applications, and (iii) the verbosity of XML Schema is mainly due to its representation style inherited from XML. Even RELAX NG 's representation style is similar to that of the XML Schema.

Hence this manuscript considers XML Schema as the benchmark for further discussion and comparison.

2.2.1. XML Schema

XML Schema (C. M. Sperberg-McQueen *et. al* (2000) was published by the W3C to provide an alternative to XML DTD. It supports namespaces, facilitates the design of open and extensible vocabularies, and meets the requirement of data-oriented applications for a richer data-typing system. It is generally considered as complex, partly because of the number of features, and partly because of the style of the recommendation which gives much stress to the validation process than the modeling features. Its style of representation is very verbose and is identical to XML.

XML Schema documents are XML documents. Even though XML Schema is the leading schema definition in terms of the available tool support features such as support for structures, data types and flexibility, it has some limitations that may hinder its future use in resource constraint environments. The limitations of XML Schema are (i) its XML like representational style increases the verbosity considerably, (ii) the process of defining user defined structures is highly verbose, and (iii) it is generally considered complex and heavyweight due to its variety of features.

2.2.2. Schema Definitions for YAML

A built-in schema does not exist for YAML, unlike for XML in the form of a DTD. However two schema definitions exist for YAML: Kwalify (Kwalify 2010) and Rx (Rx 2008).

Kwalify provides mechanisms to define base data types in YAML such as sequence and mapping. Provisions for the addition of rules and constraints are also supported. Kwalify makes use of the default serialization method in YAML for defining the schema elements. All data types supported in YAML can be used in Kwalify also.

Rx is designed as a schema for lightweight data formatting schemes such as YAML. Data types supported by Rx are of two types, simple types and collection types. Simple types are primitives like “int” and “str”. Collection types are arrays, lists and mappings. The method of representing schema definition is less verbose than XML Schema.

Kwalify and Rx are better than XML Schema in terms of verbosity due to their representational style and use of YAML's data-typing scheme. Any complex data type can be expressed as a mapping or as a sequence. It is observed that, the verbosity of Kwalify and Rx can further be reduced (using the YAML style of representation).

2.3 Message Level Security

Security need to be ensured for the messages in transit either at transport level or at message level. The transport level security is based on the Secure Sockets Layer (SSL) or the Transport Layer Security (TLS) that runs beneath the HTTP. SSL/TLS security features include authentication, data protection, and cryptographic token support for secure HTTP (HTTPS) connections. It is point-to-point, does not work with intermediaries, and is ubiquitous in nature.

Message level security (Bertino *et. al* 2006) includes the security benefits of SSL, with additional features. It is an application layer service and facilitates the protection of messages between the applications. Its main features include the protection of data chunks and provision to work with intermediaries. It can be more effective and has the added flexibility of sending messages over any transport protocol.

The advantages of message level security over transport layer security include the following: (i) provision for end-to-end security as opposed to the point-to-point security (End-to-end security is ideal when multiple intermediary nodes exist between the two endpoints), (ii) message security while in transit and at rest as opposed to only while in transit on the wire in transport layer security, and (iii) support for element-wise signing and encryption.

Message level security specifications need to be included along with the format specification to make it secure.

2.3.1. Security Specifications

A generic security specification should have a uniform format and standard to specify the parameters, algorithm and other related information across different applications. The security processes that need to be considered are (i) encryption, (ii) signature, and (iii) signcryption (Y. Zheng 1997). The signcryption process is of particular interest due to its advantages in resource conservation. It is a method to reduce the cost of the signature-then-encryption method, by combining the functions of digital signature and public key encryption in a logically single step (Y. Zheng 1997). It costs on the average 50% less in the computation time and 91% less in the message expansion than the signature-then-encryption with RSA (Zheng 1997).

W3C has published the specifications for XML encryption (T. Imamura *et. al* 2002) and signature (M. Bartel *et. al.* 2008). These specifications include syntax and processing rules for creating and representing encrypted values and digital signatures. As on date, W3C has not standardized any specification for signcryption. The XML security specifications are highly verbose as they use the XML syntax, and hence they need tailoring to suit the requirements for a data format.

The existing XML Digital Signature specification needs refinement with respect to signing of the encrypted part of a document. When verifying a signature, one must know whether the signature was computed over the encrypted or unencrypted document/fragment. The basic requirement is to communicate to the recipient any dependency between a given signature element and the encryption element(s) applied to a common XML document(s). Such an arrangement is not there at present in the W3C specification for XML Digital Signature (B. LaMacchia *et. al* 2000).

2.3.2. Specifications for ECC Compatibility

Security algorithms and protocols generally occupy the major chunk in the data formats for secure messaging. The main sources of power drain during a secure wireless session are the number of packets transmitted/received and the size of messages required for establishing a session.

Elliptic Curve Cryptography (ECC) is attractive in the wireless mobile security arena due to its dramatic decrease in the key size without compromising the security. A 160-bit ECC key offers the same level of security as a 1024-bit RSA key (W. Chou 2003). The energy analysis on various cryptographic algorithms and key exchange protocols reveal considerable savings with ECC (N. Potlapally *et. al.* 2003). Advantages of ECC include gain in the running

time (W. Chou 2003), fewer number of bits for the key size, (W. Chou 2003) and advantage in the energy consumption (N. Potlapally *et. al.* 2003).

ECC is used in the design of the elliptic curve variations of Integrated Encryption Scheme (IES) and signcryption. Elliptic Curve Integrated Encryption Scheme (ECIES) is a public-key encryption scheme based on ECC which provides semantic security against chosen-plaintext and chosen-ciphertext attacks (Certicom-Research 2000). ECC based techniques also consume less power compared to Digital Signature Algorithm (DSA) and RSA. For e.g., with a key size of 163 bits, ECDSA consumes less power than its RSA and DSA equivalents with a key size of 1024 bits, with an advantage of 255.52 mJ and 386.52 mJ, respectively (N. Potlapally *et. al.* 2003). Thus, IES, DSA and signcryption with ECC appear to be better mechanisms in the constrained wireless mobile environment. It can be concluded that the generic security specifications are to be tailored to make them compatible with ECC based algorithms.

2.4. Rewriting Attacks

A sliced serialization format is helpful, but not efficient for the secure transfer of data. The format also needs to withstand common attacks to ensure the trusted transmission of messages. XML documents are vulnerable to rewriting attacks. Many solutions were proposed in the literature to overcome this attack which include XML Digital Signature (Zheng 1997), WS Policy (S. Bajaj et. al. 2004), WS Security (A. Nadalin *et. al* 2004), SOAP Account (M. A. Rahaman *et. al.* 2006), and WS Policy Advisor (K. Bhargavan *et. al.* 2005).

XML Digital Signature (Zheng 1997) is the default and natural tool to secure the information in XML format. W3C has published the detailed specifications for signature processing and it is extensively in use now. However, XML Digital Signature has a security hole (Zheng 1997). XML Digital Signature specification allows a non-contiguous object of an XML data set to be signed separately. Each signed object is referenced by a Uniform Resource Identifier (URI) indirection from the "Reference" element of the signature. This indirect referencing is by means of an Id, without giving any clues about the actual location of the signed element. This leads to rewriting attacks, where an object can be relocated (A. Benameur *et. al.* 2008, S. K. Sinha *et. al.* 2008).

When WS Security Policy (S. Bajaj et. al. 2004) is used, security is ensured by preparing security policy files. The author and the implementer of the security policy need to be very careful in writing and implementing the policy. Flaws in the policy files can lead to holes which provide the chance for unauthorized access to system resources (M. McIntosh *et. al.* 2005, M. A. Rahaman *et. al.* 2007). Also the security loophole reported for XML Digital Signature is inherited by WS Security policy files (M. A. Rahaman *et. al.* 2007).

As the XML Digital Signature specification is used in the WS Security (A. Nadalin et. al 2004), the security hole reported for XML Digital Signature is inherited by WS Security. Multiple security headers with same name are allowed by WS Security specifications. This feature can be exploited by the attacker in gaining unauthorized access to the resources. The WS Security standard includes XML Digital Signature, XML Encryption, X.509 certificate, Kerberos ticket, etc., making it complex and heavyweight, especially for the RCNs.

WS Policy Advisor (K. Bhargavan *et. al.* 2005) is a novel approach making use of the policy configuration files. It is a rule-based tool for detecting typical errors in the configuration of policy files. It can be used to run several static queries on the policy and configuration files of Web Service Enhancements (WSE). Then it generates security reports and remedial actions for security flaws. The WS Policy Advisor shows performance degradation when the policy configuration is complex. Hence it is not a recommended tool for the constrained wireless environment (A. Benameur *et. al.* 2008).

SOAP Account (M. A. Rahaman *et. al.* 2006, M. A. Rahaman *et. al.* 2007) approach tracks the physical location of the signed objects. This is done with the introduction of an account structure consisting of the following information: (i) the number of child elements under the root element, (ii) the number of header elements in the message, (iii) the number of references in each signature, and (iv) the successor and predecessor relationship of each signed object as parent element and sibling elements.

This mechanism helps in detecting the rewriting attacks only partially due to the following limitations (A. Benameur *et. al.* 2008, S. K. Sinha *et. al.* 2008): (i) no mechanism to detect the replay attack (although MessageID or

Timestamp are proposed by the WS-Security, they are optional elements), (ii) no information that can uniquely identify the parent of the signed element (this unawareness may cause a rewriting attack), and (iii) a flaw in the SOAP Account mechanism in a typical message structure. This flaw is due to the absence of the actual hierarchical position of the signed objects in the SOAP Account structure (S. Gajek *et. al* 2007).

3 TYAML: An Alternative Format for Messaging

In this section the following facts are discussed: (i) schema formation for the proposed data format, (ii) TYAML – the proposed data formatting scheme, (iii) generic security specifications for message level security, (iv) enhancing message level security specifications for ECC compatibility, and (iv) mechanisms to check rewriting attacks.

3.1. YASchema – A Lightweight Schema Definition for the Proposed Data Format

Unlike XML Schema, YASchema supports all data types supported by YAML. Steps for YASchema creation is briefly outlined in the following subsections.

3.1.1. YASchema Creation - Bifurcation of Structure from the Content

In YAML the structure of the data is embedded in the data representation itself. If a schema definition can be prepared from the data contents, it reduces the verbosity considerably, especially when there is a set of messages which can be transmitted with a common structure. In addition to this, schema has an important role in the validation of a message after the transmission. The YASchema creation has two stages (P P Abdul Haleem *et. al* 2009): (i) preparation of data in the YAML format and (ii) bifurcation of structure and content.

a. Preparation of Data in YAML Format along with a meta file

Document creation is easy in YAML compared to XML and is created using any simple text processor. YAML streams are encoded using a set of printable Unicode characters, either in the Unicode Transformation Format (UTF) - 8 or in UTF-16. The structure is denoted by white space indentation and tab characters are not allowed for this purpose. List members are denoted by a leading hyphen (-) with one member per line, or enclosed in square brackets ([ ]) and are separated by comma space (,). Comments are denoted by the hash sign (#) and continue until the end of the line (M. McIntosh *et. al* 2005).

Metafile

When data is represented as per the YAML specifications, there can be some fields that are not mandatory, or some fields without any values populated. With multiple records in the document, there is a chance of non-uniformity in the data sets due to this. For e.g., one record may contain an optional element populated with/without a value, or without any optional element. This can hinder the schema extraction process. To overcome this, the user needs to prepare a "metafile". All fields of the document structure, with values populated are entered in the metafile. If there are no specified values, the default values are to be initialized. In addition to this, the number of occurrences of an element and whether the elements are to be squeezed or not are also to be mentioned in the metafile.

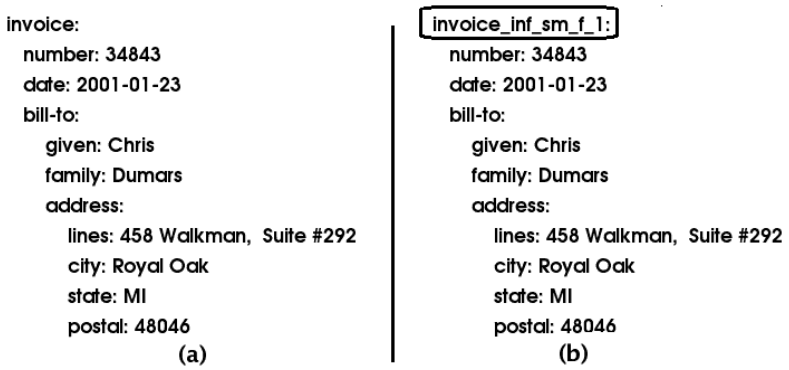

Figure 3.1: Sample YAML data and its metafle

A YAML message example and its metafile are shown in Figure 3.1. It can be observed from Figure 3.1 that in a metafile, the element name "invoice" is appended with a "_inf" part. The "_inf" part contains the type of the document (short, small, etc.), whether the element is to be squeezed or not and the number of occurrences of the element in the data. The metafile is used for the bifurcation of structure and content. It can be discarded later as the required information about the structure will be maintained in the YASchema.

b. Bifurcation of Structure and Content

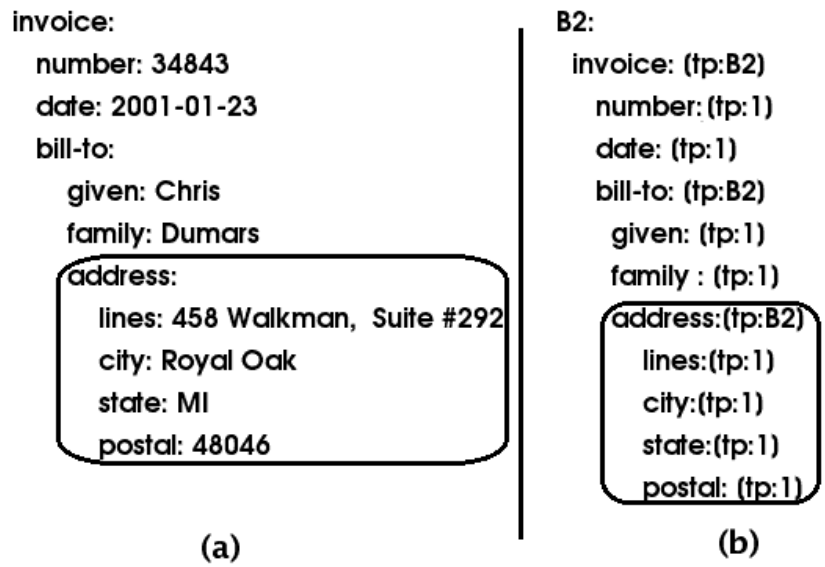

Figure 3.2: Sample YAML data and its schema definition

The metafile prepared by the user is to be scanned node-by-node to determine the schema elements. A DFS (depth-first-search) of the contents of the meta file is needed for this. For this, the document is scanned one node at a time. There are three kinds of nodes in YAML - scalar, sequence, and mapping. Apart from this, there are mappings of sequences and sequences of mappings. The message structure is to be evaluated to decide the type of schema to be constructed. The type can be of any of the three kinds - scalar, sequence and mapping. Mappings of sequences and sequences of mappings are considered as special cases. Primitive details like name of the "field", its data type and an ID value are added to the schema for every member in the node. Further details to the schema could be added in the validation and verification stages. Once this process is completed, it results in a schema description of the message to be transmitted. The YASchema definition for a sample YAML document is as shown in Figure 3.2.

3.2. Thinned YAML (TYAML) – A Lightweight Data Format Based on YAML

This section proposes a lightweight data format by applying some thinning techniques to the plane YAML format. The resulting data format is called Thin YAML (TYAML). TYAML must be an alternative to XML, but with less verbosity. The objectives of the work in this section can be summarized as follows: development of a data formatting scheme (as an alternative to XML), with the following properties: (i) less verbose, (ii) schema aware, (iii) secure, but with less overhead, (iv) self descriptive, and (v) platform independent.

YAML is the base format for the proposed work. YASchema is the primary step for the development of TYAML. Verbosity of YAML is reduced in two phases to get the thinned format. In the first phase, verbosity is reduced using the inherent properties of plain YAML, and a method for serialization is proposed. In the second phase, verbosity reduction is achieved with the help of the features in YASchema. The process of developing TYAML from base YAML format is as shown in Figure 3.3.

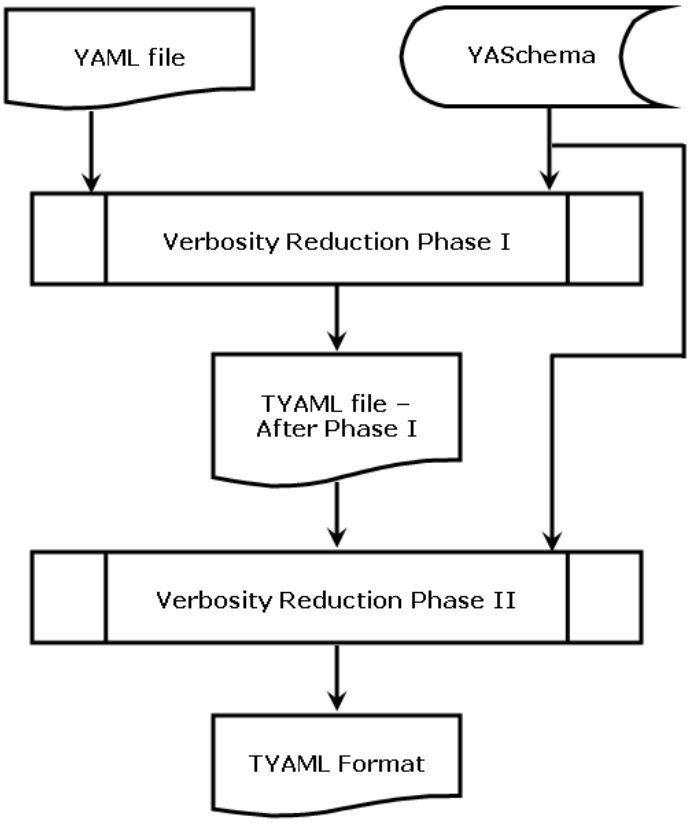

Figure 3.3: Development of TYAML format from plain YAML

3.2.1. Verbosity Reduction Phase I

One can observe that YAML's natural approach for representing information is as per the guidelines of the legacy XML. Even though this arrangement provides high degree of readability, it also demands more space for representing the information. The "flowstyle" method (A. Nadalin *et. al* 2004) helps one to squeeze the verbosity considerably. Phase I uses the flowstyle method for verbosity reduction. In certain cases, reorganization of the contents to this style yields better performance. For e.g., a complex message consisting of a customer invoice (containing many product details).

The usual way of converting a mapping to flowstyle is as shown in Figure 3.5 (the original YAML message is shown in Figure 3.4). Here, the mappings are squeezed into flowstyle, by repeating the element names with each mapping. This increases the redundancy due to the repeated occurrences of the element names. When there are many such mappings in a message with a common structure, it is better to list the element names of the mappings once, followed by the values of the mappings in the successive lines as shown in Figure 3.6. This approach is used while squeezing the message contents. This helps in the reduction of bytes needed to represent a message. In the case of small messages, the effect of this technique may not be visible, as they may not have repeated occurrences of data with similar structures. But this method is more effective in the case of large messages. To achieve this task, the entire document is scanned first. A Depth-First-Search (DFS) is employed for this purpose, as YAML follows a tree structure. Every node is visited one at a time and the "squeezable" node portions are squeezed into the flowstyle in two stages.

```
invoice:
  - number: 34841
    date: 2001-01-23
    bill-to:
      given: Chris
      family: Dumars
      address:
        city: Royal Oak
        state: MI
        postal: 48046
  - number: 34842
    date: 2001-01-23
    bill-to:
      given: Duke
      family: Francis
      address:
        city: Flee Ghar
        state: MI
        postal: 48047
```

Figure 3.4: Data in YAML Format

First, the element names of the node are serialized in the flowstyle. A DFS is done with all nodes under the root node, and the nodes are stored initially in a list. Subsequently, each node in the list is visited. Element names are collected separated by "," during the visit to a node. To close the braces properly, the current node list is then pushed into a stack during a visit. Once the visit to the current node or its child nodes are recorded, the node lists are popped off and the visit is repeated for the remaining nodes in the list. The visited nodes are getting removed from the list. The stack arrangement helps to properly place the closing braces in the case of mappings within mappings.

```
invoice:
   - {number: 34841, date: 2001-01-23, bill-to: {given: Chris, family:
Dumars, address: {city: Royal Oak, state: MI, postal: 48046}}}
   - {number: 34842, date: 2001-01-23, bill-to: {given: Duke, family:
Francis, address: {city: Flee Ghar, state: MI, postal: 48047}}}
```

Figure 3.5: Data in flowstyle

```
invoice:
   - {<number, date, bill-to: {<given, family: {<address: {<city, state,
postal>}>}>}
   - {34841, 2001-01-23, {Chris, Dumars, {Royal Oak, MI, 48046}}}
   - {34842, 2001-01-23, {Duke, Francis,  {Flee Ghar,  MI, 48047}}}
```

Figure 3.6: Message in flowstyle (enhanced)

In the second stage of scanning, the nodes are visited again to list the element values in the flowstyle. First, the node contents are listed in the natural method of the flowstyle (as shown in Figure 3.5). Then, all element names are removed from the serialization.

3.2.2 Verbosity Reduction Phase II

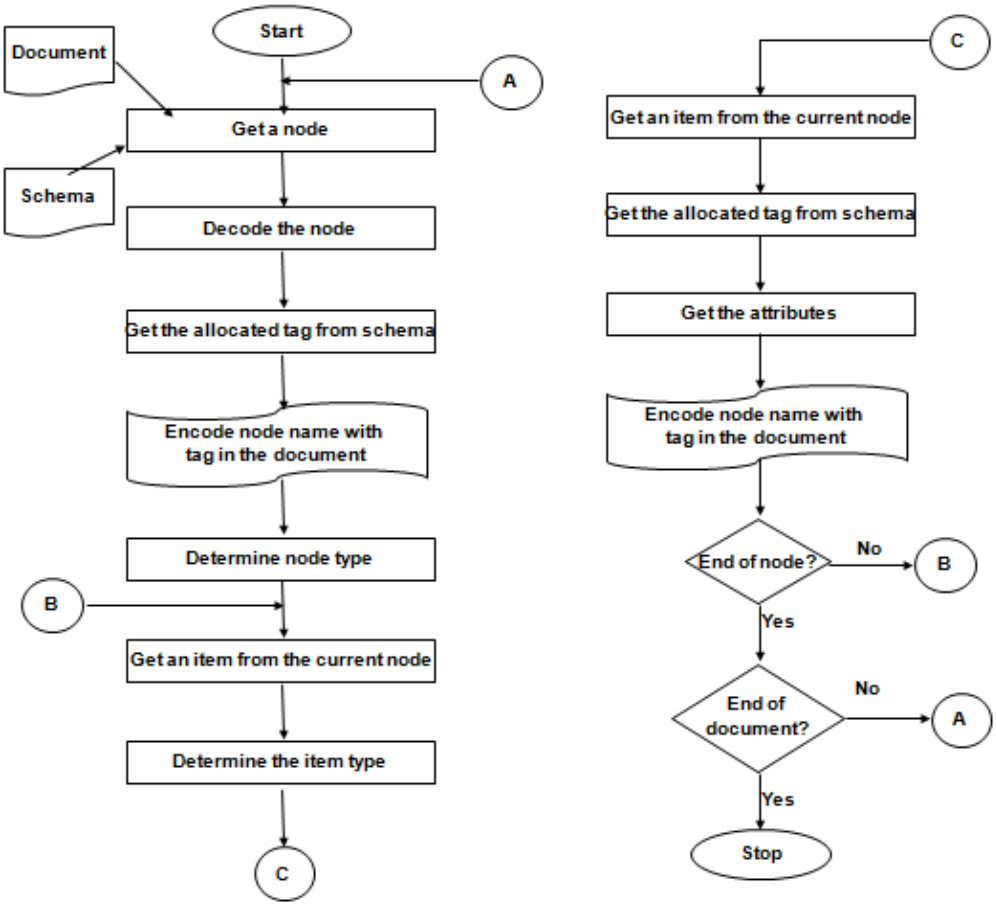

Figure 3.7: Verbosity Reduction (Phase II)

This phase uses YASchema for reducing the message verbosity. All primitive details regarding each and every element in the message are defined in the schema creation stage itself. In addition to this, special ID codes are provided for each element. The message is scanned and the elements are encoded with the ID codes. The ID codes could be referred from the accompanying schema, or if the sender likes to reuse a schema that had been sent earlier, the schema to be referred could be specified in the message itself. This makes the decoding easy, making the message schema-aware, and is a useful value addition to the plain YAML.

The procedure for Phase II (Figure 3.7) is also DFS. YASchema has the ID code to be assigned for each element. Every element of YASchema is scanned, and the occurrences of the scanned element name in the message are encoded with its corresponding ID code. It is a search and replace procedure.

4 Message Level Security Specifications in TYAML

In addition to optimizing the message verbosity, provisions for secure transfer of messages is also a must for a data format. Exchanging the information without disclosing its contents is vital in the present-day networked world. The proved mechanism for protecting the confidentiality of information across different systems and networks are based on cryptography. Apart from ensuring the confidentiality of messages, authentication, data integrity and support for non-repudiation also must be ensured. Incorporating these mechanisms makes TYAML safe and secure.

There is a need for a uniform specification to describe and exchange the parameters and associated information. The security specification defines the formulation of document structure and the associated schema for a particular purpose. This is essential to maintain uniformity among the different platforms and application program interfaces. The generic standard proposed in this section includes specifications for (i) encryption, (ii) digital signature, and (iii) signcryption. Specification for a canonicalization process is discussed in (P P Abdul Haleem *et al* 2011).

Signcryption is a technique that simultaneously performs the functions of both digital signature and encryption (Y. Zheng 1997). Using the RSA cryptosystem, it costs on the average 50% less in computation time and 91% less in message expansion than the signature-then-encryption does (Zheng 1997).

4.1. Generic Security Specifications

The guidelines specified by W3C (D. Mundy *et. al* 2004) for defining the specifications are (i) the specification must provide for the separation of information about the process (Encryption/Signature/Signcryption) from the processed (Encrypted/ Signed/Signcrypted) data, (ii) the specifications must be adaptable to different cryptographic algorithms, and (iii) facilities for processing at the element level (within a document) and at the document level are to be incorporated.

Provisions to include the following most important information are to be made in the specifications (i) type of "process" at element level or on the whole document, (ii) algorithm applied for the "process", and (iii) details about the key being used. The goal behind the generic specifications is that different APIs should be able to secure the data and further process it in a uniform format. *CipherData* element is a child of *EncrData* element. It encompasses the *CipherValue* element to enclose the encrypted string.

The proposed YASchema maintains all the facilities existing in XML (the facility to specify the required and optional elements and their attributes, for instance). Considerable value additions are also made in the schema definition. The ability to define shorter tags for the element names is the prominent among these. This is an optional arrangement to further reduce the size of the final message. The order of occurrence of the attributes can also be specified. Details like the schema being used and whether the name encoding is applied are specified along with the document with the help of a special directive.

4.1.1. Specification for Encryption

The minimal structure of the encryption specification and the corresponding YASchema are as shown in Figure 4.1. It can be observed that the encrypted part of the document is initiated with the *EncrData* mapping. It is the core element where all encrypted data are enclosed. When the entire document is encrypted, the structure node starts with this element. As in XML, the *EncrData* should neither be enclosed within another *EncrData* element, nor it can house another mapping of the same type. *Type* indicates the nature of encryption (1 = entire document, 2 = Element and 3 = Content of an element).

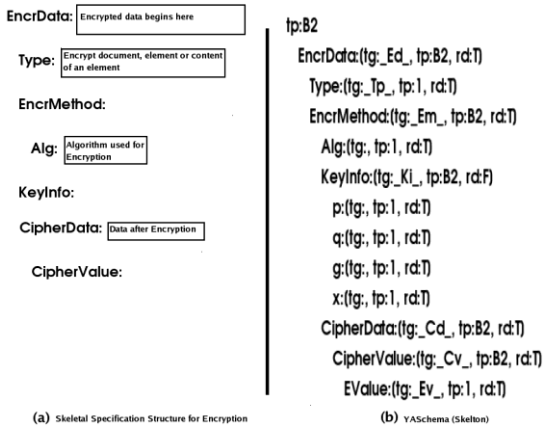

Figure 4.1: Encryption Specification and YASchema: Skeletal Structure

4.1.2 Specification for Digital Signature

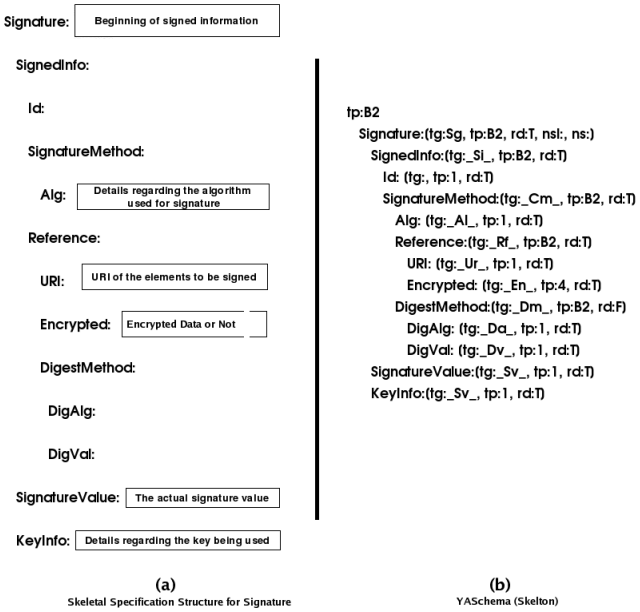

Figure 4.2: Signature Specification and YASchema: Skeletal Structure

The minimal structure for signature specification is as shown in Figure 4.2. The *Signature* mapping identifies the data being signed. The *SignedInfo* element is mandatory. It houses all information related to the signed resources. The *SignatureMethod* lists the algorithm used to create the digital signature. The actual signature (in *SignatureValue* element) is outside this element.

The proposed specification takes care of an aspect overlooked by the XML Digital Signature Working Group: signing of the encrypted part of a document. When verifying a signature one must know whether the signature, was computed over the encrypted or unencrypted document/fragment. The basic requirement is to communicate to the recipient any dependency between a given Signature element and the encryption element(s) applied to a common XML document(s). Such an arrangement is not visible at present in the W3C specification for the XML Digital Signature (Bertino *et. al* 2006). A flag is proposed (along with part of the documents being signed) to indicate whether it (the portion to be signed) is encrypted or not. This flag indicates that the ciphertext should not be decrypted prior to the signature verification.

4.1.3. Specification for Signcryption

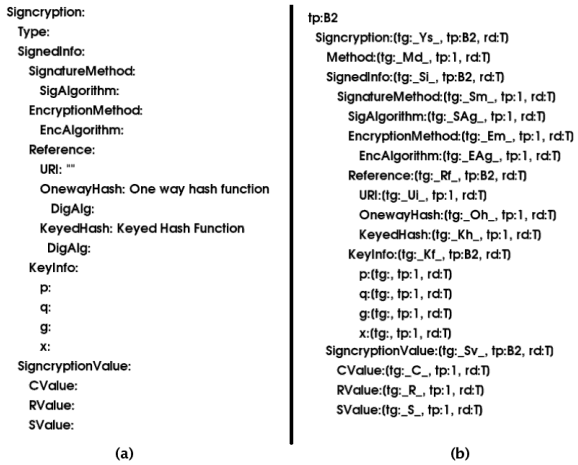

Figure 4.3: Signcryption Specification and YASchema: Skeletal Structure

The proposals for signcryption and associated YASchema definition are as shown in Figure 4.3. The skeleton specification for signcryption is self descriptive. Fields are provided to specify encryption and signature processing. The corresponding YASchema construct makes use of the verbosity reduction features to reduce the size of the messages. The first entry (*tg*) in this list is a tag that can be used for encoding of the descriptive name of the field. The second entry (*tp*) indicates the data type and the third entry (*rd*) indicates whether this field is mandatory or not.

5 Enhancing TYAML Message Level Security for ECC Compatibility

Security protocols and algorithms play a limiting or negative role in process efficiency and energy efficiency. So, restricting the number of packets transmitted or received and using smaller encryption keys are important to overcome the above limitations.

The application of Elliptic Curve Cryptography (ECC) based techniques are promising in this regard. ECC is already in use in the wireless mobile security arena due to its dramatic decrease in the key size and running time without any compromise on the security. A 160-bit ECC key offers the same level of security as a 1024-bit RSA key (Zheng 1997). The energy analysis on various cryptographic algorithms and key exchange protocols reveals considerable savings with ECC (T. Imamura *et. al* 2002). ECC is used in the design of the elliptic curve variations of Integrated Encryption Scheme (IES), Digital Signature Algorithm (DSA), and signcryption. The Elliptic Curve Integrated Encryption Scheme (ECIES) is a public-key encryption scheme based on ECC which provides semantic security against chosen-plain text and chosen-ciphertext attacks (M. Bartel *et. al.* 2008). The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve version of the DSA (G. White *et. al* 2007). ECDSA is superior to RSA and DSA in terms of the total time needed for key generation, signature creation and verification (CISCO Tech Note 2005). A convincing advantage in the energy consumption for key generation, signature and verification processes with a key size of 163 bits against 1024 bit equivalents of RSA and DSA is recorded in (T. Imamura *et. al* 2002). Thus, EC version of IES, DSA and signcryption appears to be better mechanisms in RCNs.

5.1. Specifications Specification for ECC based Algorithms

The generic specifications discussed in Section 4 are enhanced to make it compatible with EC versions of (i) IES, (ii) DSA, and (iii) signcryption.

5.1.1. Specification for ECIES

```
EncrData:
  Type:
  EncrMethod:
    Algorithm:
  KDF:
    Algorithm:
  MAC:
    Algorithm:
  SharedInfo:
    S1:
    S2:
  KeyInfo:
    n:
    g:
      x:
      y:
    a:
    b:
    p:
    h:
  CipherData:
    CipherValue:
      vValue:
      cValue:
      tValue:
```

(a)

```
tp:B2
  EncrData:(tg:Ed, tp:B2, rd:T, nsl:, ns:)
  Type:(tg:_Tp_, tp:1, rd:T)
  EncrMethod:(tg:_Em_, tp:B2, rd:T)
    Algorithm:(tg:_Al_, tp:1, rd:T)
    KDF:(tg:, tp:B2, rd:T)
      Algorithm: (tg:_Al_, tp:1, rd:T)
    MAC:(tg:, tp:B2, rd:T)
      Algorithm: (tg:, tp:1, rd:T)
    SharedInfo:(tg:_Shi_, tp:B2, rd:T)
      S1: (tg:, tp:1, rd:T)
      S2: (tg:, tp:1, rd:T)
    KeyInfo:(tg:_Ki_, tp:B2, rd:T)
      n: (tg:, tp:1, rd:T)
      g: (tg:, tp:B2, rd:T)
        x: (tg:, tp:1, rd:T)
        y: (tg:, tp:1, rd:T)
      a: (tg:, tp:1, rd:T)
      b: (tg:, tp:1, rd:T)
      p: (tg:, tp:1, rd:T)
      h: (tg:, tp:1, rd:T)
    CipherData:(tg:_Cd_, tp:B2, rd:T)
      CipherValue:(tg:_CV_, tp:B2, rd:T)
        vValue: (tg:_vv_, tp:1, rd:T)
        cValue: (tg:_cv_, tp:1, rd:T)
        rValue: (tg:_rv_, tp:1, rd:T)
```

(b)

Figure 5.1: (a) ECIES Specification (b) Corresponding YASchema

The skeleton structure of the specifications for ECIES is as shown in Figure 5.1. *EncrData* is the root element. The encrypted part of the document is initiated with the *EncrData* mapping. It is the core element where all encrypted data are enclosed. Elements specific to *Ecies* are *KDF*, *MAC* and *SharedInfo*. These elements house the key derivation function, message authentication code function and the shared info called *S1* and *S2*, respectively.

Also, the *CipherValue* element encompasses the results *v* (the public key), *c* (the cipher key) and *t* (the message tag).

5.1.2. Specification for ECDSA

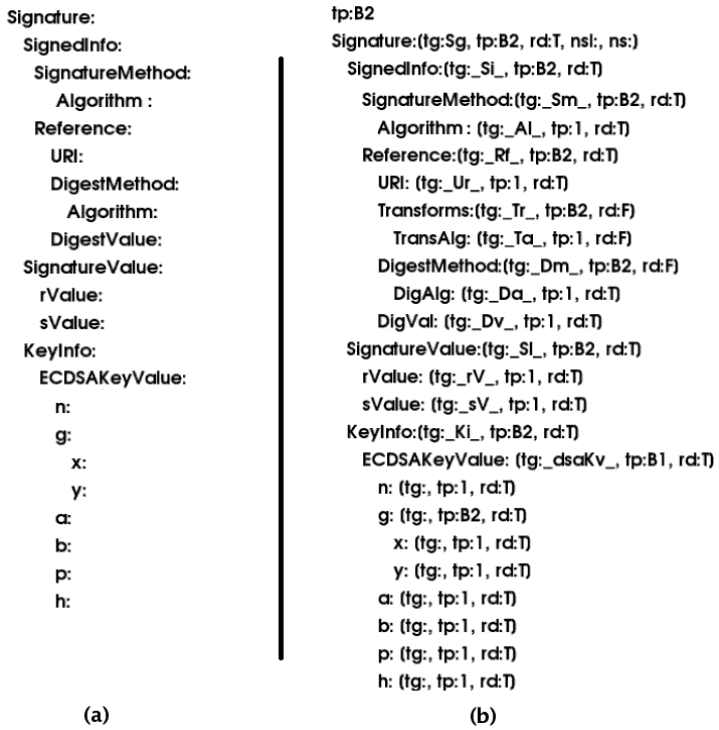

Figure 5.2: (a) ECDSA Specification (b) Corresponding YASchema

The proposed specification for digital signature is modified to include the elements needed for ECDSA operations. The minimal structure is shown in Figure 5.2. The specification includes provisions to maintain information such as signature algorithm, digest method, signature value and key information.

5.1.2 Specification for EC Signcryption

The skeleton structure of the specification and YASchema for EC Signcryption is shown in Figure 5.3. The specification has provision to include all the parameters needed to perform EC Signcryption - *Signature Algorithm, Encryption Method, Oneway Hash, Keyed Hash, Key Information and Signcryption Values*.

The skeleton structure of the specification and YASchema for EC Signcryption is shown in Figure 5.3. The specification has provision to include all the parameters needed to perform EC Signcryption - Signature Algorithm, Encryption Method, Oneway Hash, Keyed Hash, Key Information and Signcryption Values.

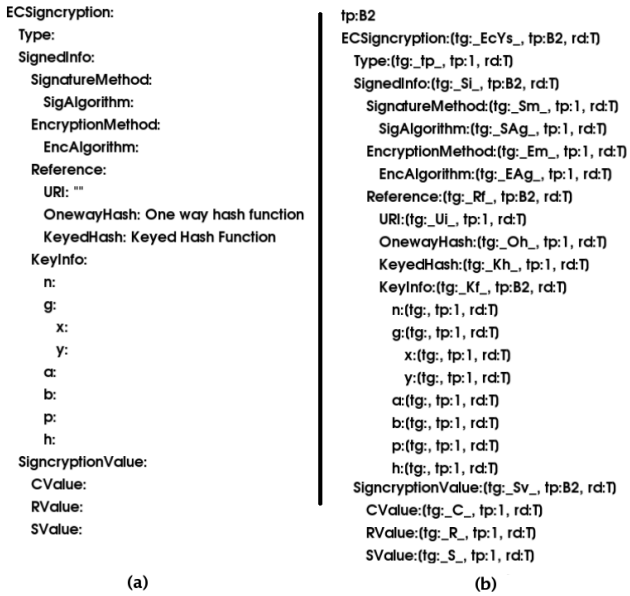

Figure 5.3: (a) EC Signcryption Specification (b) Corresponding YASchema

6 Lightweight Schemes to Check Rewriting Attacks in TYAML

In this section a comprehensive approach to make TYAML documents free from rewriting attacks are proposed. The proposed lightweight scheme to check rewriting attacks consists of the following: (i) refinement of the specification for Digital Signature, (ii) inclusion of policy assertions in YASchema, and (iii) formulation of an improved accounting structure called YAccount to keep track of the signed objects in the messages. By combining the YASchema and YAccount, a layered architecture mechanism (P P Abdul Haleem *et. al* 2012) is also proposed to check the rewriting attacks.

6.1. Refinement of the Digital Signature Specification

The specification for signature processing, as shown in Figure 6.1, inherits the security hole mentioned in Section 2. To eliminate the trapdoor for a possible rewriting attack, an additional structure is added to the signature specification which captures the physical location of each signed object (its ID, immediate predecessor, siblings and depth in the document hierarchy). This data, which is named as YLocation as depicted in Figure 6.2 is appended after the existing signature specification structure as a sibling of SignedInfo.

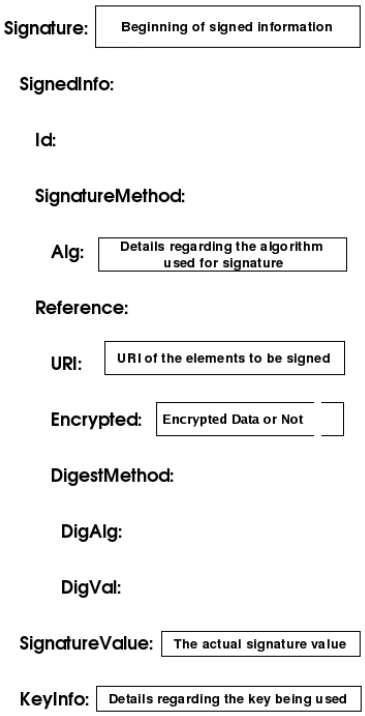

Figure 6.1: Signature Specification

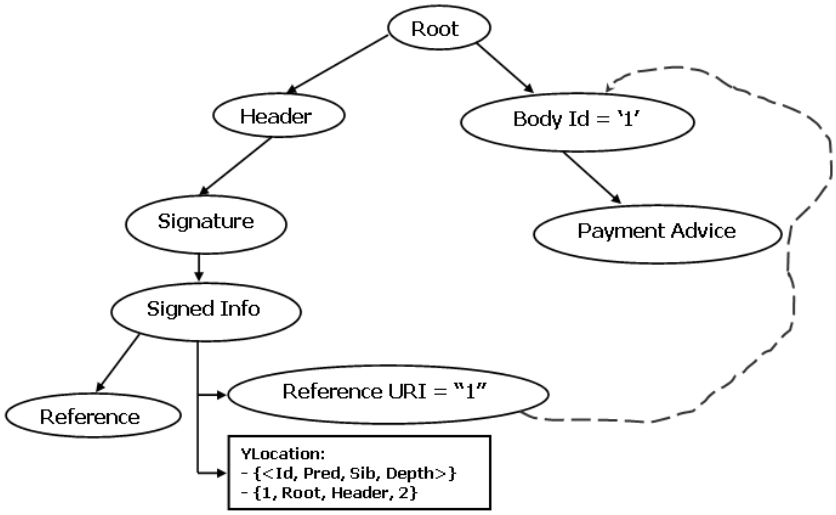

Figure 6.2: Addition of Location of Signed Objects in the Signature Specification

This additional information cannot be made mandatory always with the signature specification due to the compatibility issues with the existing applications. Hence additional mechanisms are to be proposed to ensure trusted transmission of data.

6.2. Inclusion of Policy Assertions with YASchema Elements

The schema definition, YASchema, is extended to include policy assertions as well. The policy assertions stipulated in WS-Policy (B. LaMacchia *et. al* 2000), WS-Policy Assertion (L. M. Feeney *et. al* 2001) and WS-Security Policy (G. Allard *et. al.* 2006) are prepared in separate policy files. Mapping of policy assertions with the inherent properties of YASchema eliminates the need to maintain a separate policy file.

The three basic assertions (N. Potlapally *et. al*. 2003) commonly in use in the policy files are: (i) message predicate assertion (to list the mandatory parts of the message), (ii) integrity assertion (to list the parts of the message to be signed), and (iii) confidentiality assertion (to list the parts of the message to be encrypted). In addition to this, "age assertion" (B. LaMacchia *et. al* 2000) (which specifies the acceptable time period before messages are declared "stale" and discarded) is also included in the proposal. A combination of these preconditions, in the form of a declarative XML format, prevents the injection of bogus entries and alteration of message contents.

```
B2:
  # ****  Age Assertion (age:) ****
  # ****  Mandatory Assertion (rd:) ****
  invoice:(tg:_1_, cl:B2, tp: B2, rd: T, sz: 1, no:25, age:)
    invno:(tg:_1_, cl:B2, tp:1, rd:T)
    date:(tg:_2_, cl:B2, tp:01, rd:T)
    # ****  Integrity Assertion (sd:) ****
    billto:(tg:_3_, cl:B2, tp:B2, rd:T, no:3, sd:T)
      given:(tg:_4_, cl:B2, tp:01, rd:T)
      family:(tg:_5_, cl:B2, tp:01, rd:T)
      address:(tg:_6_, cl:B2, tp:B2, rd:T, no:4)
        lines:(tg:_7_, cl:B2, tp:01, rd:T)
        city:(tg:_8_, cl:B2, tp:01, rd:T)
        state:(tg:_9_, cl:B2, tp:01, rd:T)
        postal:(tg:_10_, cl:B2, tp:01, rd:T)
    tax:(tg:_11_, tp:03, rd:T)
    total:(tg:_12_, tp:03, rd:T)
  # ****  All Policies Puttogether ****
policy:
  mand (_ALL_)
  sid (_3_)
```

Figure 6.3: Mapping of Base Assertions with YASchema constructs

Now, mapping of these assertions to the YASchema constructs can be done as follows: (i) the required elements are marked (this helps in checking whether any of the required elements are absent after receiving the message and it is a message predicate assertion), (ii) the order of occurrence of the elements in the message is listed (this helps us to check whether any of the bogus elements are injected to the message structure by the intruder), and (iii) the data type of the elements are specified. In addition to this, the list of elements to be signed (integrity assertion), the list of elements to be encrypted (confidentiality assertion), and age assertion are also added to the YASchema. A description about the mapping of policy assertions with YASchema constructs is shown in Figure 6.3.

The policy assertions can check the rewriting attacks, but the flaws in preparing policy files can hinder the trustworthiness of the method. This happens due to the lack of proper understanding of threats and security mechanisms. For e.g., if some of the message parts are not included in the message predicate, either a new bogus header can be inserted or the message can be routed to another user unintended by the sender (W. Chou 2003). Thus, a modified accounting structure is to be proposed to supplement the trust level.

6.3. Formulation of the *YAccount* Structure to Track the *SignedObjects*' Location

An account structure is proposed which keeps track of the actual location of all the signed objects in the message. This structure is to be bundled with the data in transit. The structure which is called "YAccount" consists of the following information: (i) the number of child elements of the root, (ii) the number of header elements, (iii) the number of references for the signing element, (iv) the predecessor of the signed object, (v) sibling relationship of the signed object, and (vi) depth of each signed element. In addition to this, a unique ID is also assigned to each signed object. This ID is part of the YAccount. Though all elements of YAccount are important in tracking the signed objects, inclusion of ID and depth can prevent the security holes discussed in (K. Bhargavan *et. al.* 2005). This is because there is a chance that the attacker may be able to retain the present values of all the other elements while modifying the message structure. But the ID is unique for each element and the depth information is changed when the position of the element is changed in the document hierarchy.

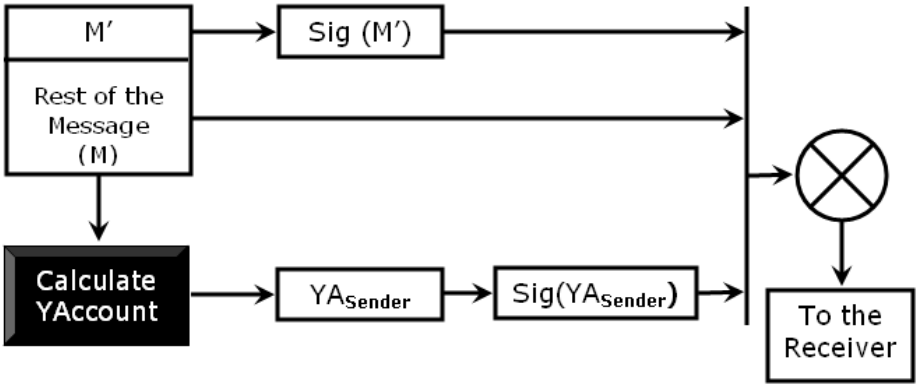

Figure 6.4: *YAccount* before the Rewriting Attack

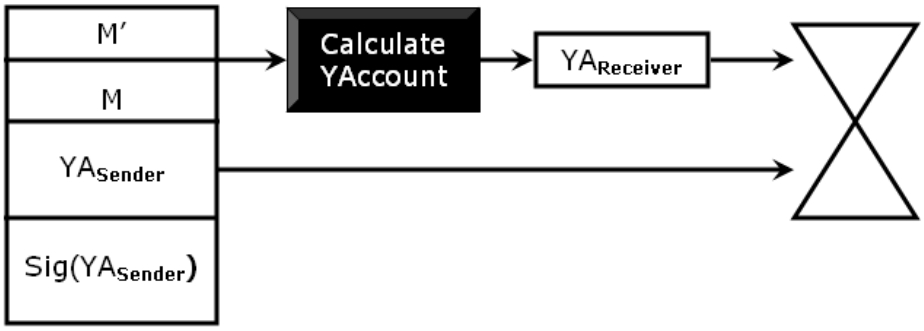

Figure 6.5: *YAccount* after the Rewriting Attack

The functionality of the approach is illustrated in Figures 6.4 and 6.5. The YAccount is calculated by the sender before sending the message which is included in the message to be transmitted in the signed format. M' denotes the portion of the data that is to be signed. The signed part, YAccount calculated and signed, and the rest of the data are concatenated and sent to the receiver (Figure 6.4). The receiver re-calculates the YAccount and compares the calculated YAccount with the account information already received from the sender as in Figure 6.5.

6.4. Combining YASchema and YAccount: A Layered Architectural Mechanism to Check the Rewriting Attacks

A layered approach is needed to prevent rewriting attacks because (i) the addition of YLocation to the signature specification cannot be introduced to all situations in the existing applications, (ii) to strengthen the overall trust level of the mechanism so that security holes in one level can be exposed in the other level, and (iii) to make the methodology compatible with the constraints of the wireless mobile environment. The choice of making use of any/all of the proposed mechanisms is left to the user, depending upon the importance of the message and the situation of the surroundings. The simplest approach in the stack is designated as Layer 0.

In Layer 0, the inherent properties of YASchema are used to check the attacks. In addition, the most common policy assertions in the WS Policy specifications are mapped with the YASchema constructs. In Layer 1, the proposed accounting structure called YAccount is employed to check the attacks. These two approaches in succession make the proposed method better in tracking the rewriting attacks.

6.4.1. Layer 0

As explained earlier, the inherent properties of the YASchema are useful for checking the attacks. In addition, the most common policy assertions in the WS Policy specification are mapped with YASchema constructs. YASchema can be exchanged between senders and receivers in two ways: sender and receiver agree upon a predefined schema, that was exchanged in advance, and YASchema is sent along with the message.

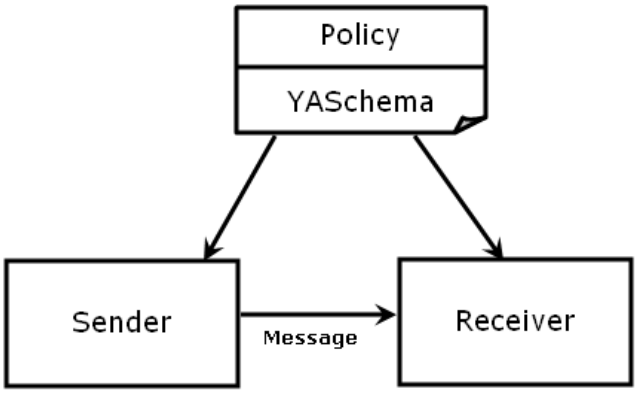

Figure 6.6: Layer 0 - With a common YASchema

In the first case, as shown in Figure 6.6, the structure at the receiver's side is checked as per the assertions and the specifications mentioned in YASchema. Anomalies of the document structure, especially in the modified bogus header part, are an indication that the message has been intercepted. As there will not be any specifications for the added bogus header entry in YASchema, the received document is considered as tampered.

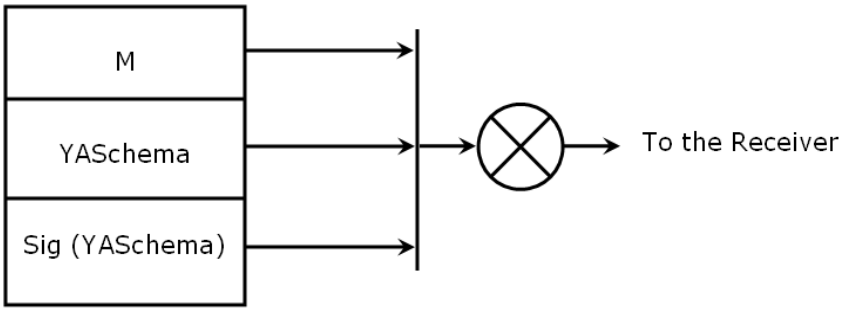

Figure 6.7: Layer 0 - With YASchema being part of the message (sender's side)

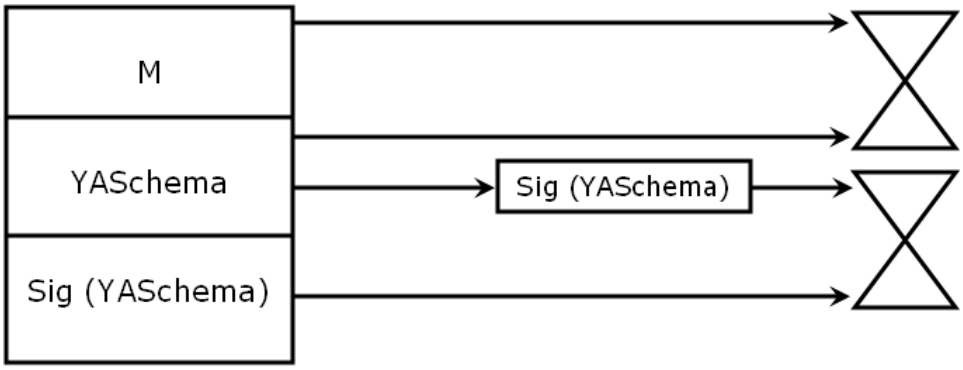

Figure 6.8: Layer 0 - With YASchema being part of the message (receiver's side)

In the second case, the digest value of the YASchema is also attached with the message by the sender (Figure 6.7). This digest value is calculated again by the receiver and compared against the digest value received (Figure 6.8). In any case, if the YASchema is modified by an attacker, that will be reflected in the digest values. Hence at this stage itself, the attack is detected. If there is no mismatch, the received structure is verified against the YASchema for further anomalies.

6.4.2. Layer 1

The functionality of this level is as explained in Section 6.3. The process of sending policy assertions with the document is summarized as follows: (i) the sender and receiver mutually agree over the trust level (Layer 0, Layer 1 or both) to be adopted for the data transfer, (ii) if Layer 0 is chosen, the sender and receiver must agree upon whether to share an existing YASchema or to append a new YASchema with the message, (iii) in the first case, the structure at the receiver's side is checked against the YASchema constructs and as per the assertions mentioned in the YASchema. If any irregularities are found, it will result in the rejection of the message, (iv) in the second case, the digest value of the YASchema that is attached with the message by the sender is compared against the calculated value at the receiver's end, (v) if there is no tampering of the message, the check will be done as per the constructs and policy assertions mentioned in the YASchema, (vi) if Layer 1 is chosen, the YAccount received is compared against the calculated YAccount for anomalies, and (vii) all of the steps (ii to vi) are done, if both trust levels are chosen.

6.5. Attack Case Studies

This section presents two case studies to check the schemes for the detection of rewriting attacks. A classical attack case discussed in related papers is considered first. Then a particular case that breaks the inline approach as demonstrated in (K. Bhargavan *et. al.* 2005) is discussed. If both the attacks are detectable by the proposed approach, that confirms the capability of the approach. Since the methodology adopted for YLocation and YAccount are identical, only the steps with YAccount are presented here.

6.5.1. Case I

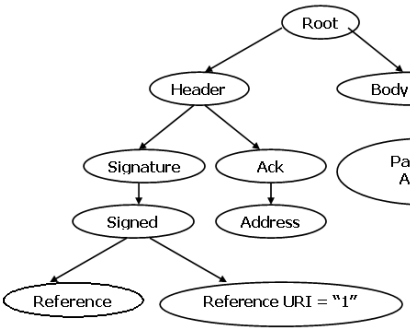

Figure 6.9: Payment Advice - Before the Attack

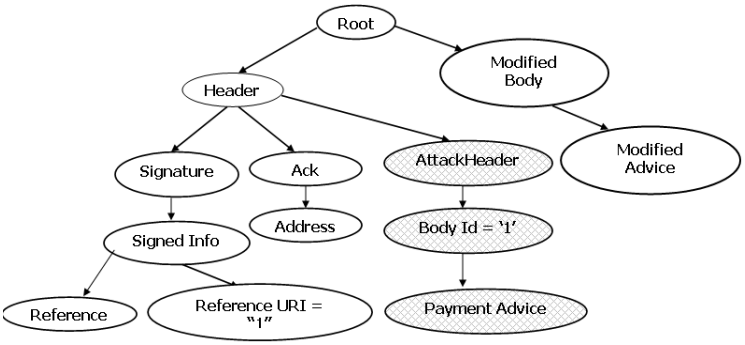

Figure 6.10: Payment Advice - After the Attack

This attack modifies the message structure by inserting bogus entries. The scenario for this case is as follows: A customer, Alice, issues a payment voucher for 1000 Euros to be transferred from her account to the supplier's (Bob's) account as shown in Figure 6.9. Some malicious attacker intercepts this message and updates it stating to transfer 5000 Euros instead of 1000 Euros as shown in Figure 6.10. The elements related to the payment amount are not explicitly shown.

At first, Layer 0 mechanism checks the payment advice. If a common YASchema is agreed upon by the sender and receiver, the structure at the receiver's side is checked as per the YASchema constructs. Anomalies of the document structure, especially in the modified bogus header part, are the indication that the message has been intercepted. If confirmation with the structure is successful, then the received document is checked as per the assertions and specifications mentioned in the YASchema. As there cannot be any specifications for the added bogus header entry in the YASchema, the document is declared as tampered.

YASchema is often allowed to be part of the message being transferred. In such cases, the digest value of the YASchema is also attached with the message by the sender. This digest value is calculated again by the receiver and compared against the digest value received. If YASchema is modified by the attacker, it will be revealed in the digest value comparison. Hence at this stage itself, the attack is detected. If there is no mismatch, the received structure is verified against the YASchema for anomalies.

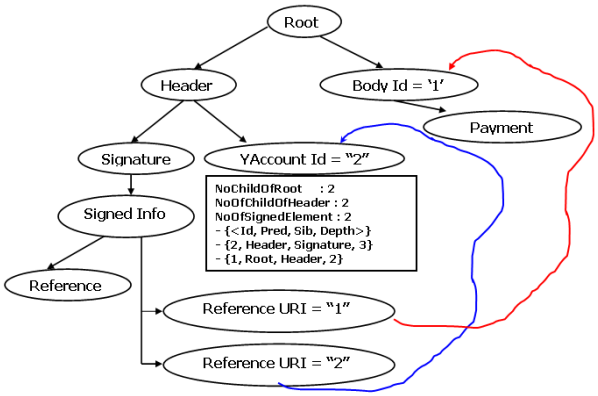

Figure 6.11: Payment Advice - YAccount before the Attack

Now, if the Layer 0 test is successfully completed, then Layer 1 test is carried out. The sender prepares the YAccount for the payment advice as shown in Figure 6.11. The existing structure of the payment advice is reflected in the YAccount information. This information is sent to the receiver. The attacker intercepts the message, modifies it, and sends it to the receiver in the next hop. The receiver now has to detect the attack on the message.

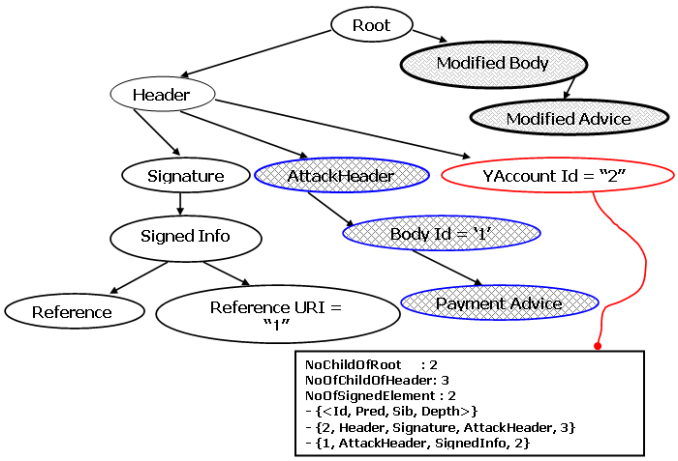

Figure 6.12: Payment Advice - YAccount after the Attack

When the message is received, the receiver does the validation of the YAccount information. The YAccount is calculated again as shown in Figure 6.12. The number of children of the root and the number of signed elements remain the same. But the number of header elements increases to 3 after the attack due to the addition of bogus

entries. Dissimilarities are also evident in the predecessor, sibling, and depth information of the signed elements. Due to the clear mismatch in the scenario, the receiver discards the message as invalid. In addition, if an attacker changes the YAccount information according to the modified message contents, then this message will be invalidated by the receiver while validating the signature of the signed YAccount by the originator.

6.5.2. Case II

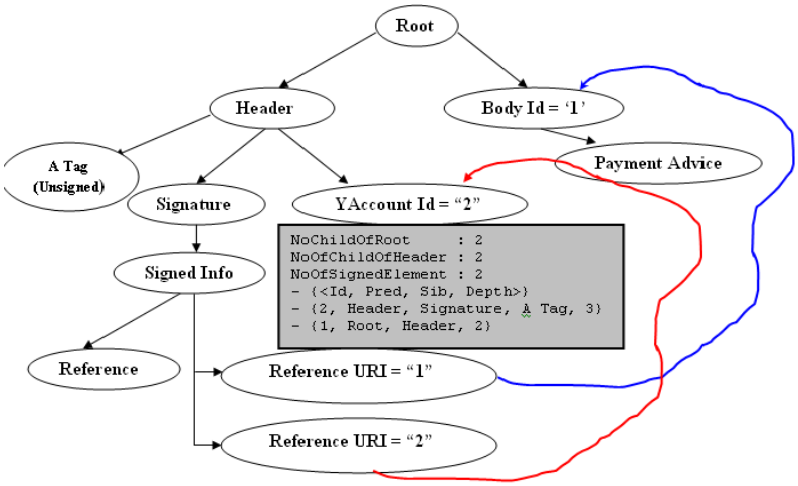

Figure 6.13: Payment Advice with an unsigned tag in the header - YAccount before the Attack

There is a possibility of at least one unsigned element existing within the Header. An attacker can take advantage of this situation, as the YAccount structure is not taking care of the unsigned elements. Such a scenario is shown in Figure 6.13. The YAccount information calculated is also shown in Figure 8.16. An attacker moves the body part (which contains the actual payment to be made) including its parent and sibling elements beneath "A Tag", and adds a bogus element at the original position as shown in Figure 6.14.

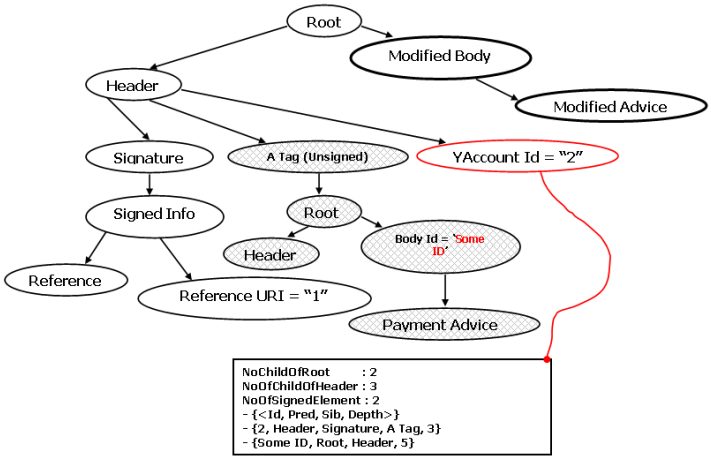

Figure 6.14: Payment Advice with an unsigned tag in the header - YAccount after the Attack (Possibility I)

As described in the earlier case, verification with a shared YASchema commonly agreed upon by the sender and receiver will definitely report the anomalies in the received document, if any.

Tracing the attack in Layer 1 is described as follows: From figure 6.13, it can be noted that the moved element preserves its relationship with the predecessor and siblings. Hence the newly calculated YAccount information provides the same information as before for (i) the number of children under root as 2, (ii) the number of children of the Header as 3, and (iii) number of signed elements as 2. But the list of ID, Predecessor, Siblings and Depth

information are of importance here. If the attacker directs the reference to the forged element so that he can be safe from a signature anomaly (as in Figure 6.14), then also, both the ID and Depth information will be different. The ID value of the forged element will be different, as the ID needs to be unique. If this restriction cannot be forced due to any reason, the depth information will be different as the structure is altered.

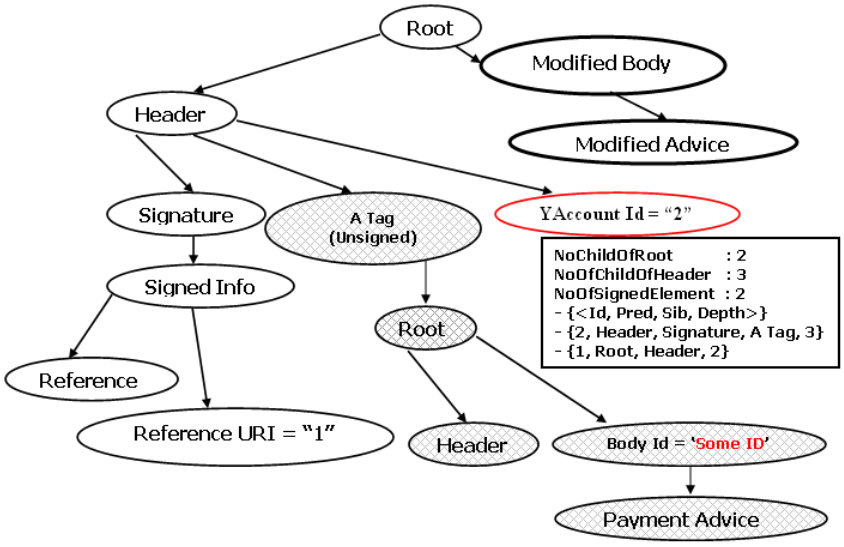

Figure 6.15: Payment Advice with an unsigned tag in the header - YAccount after the Attack (Possibility II)

The second possibility is that the attacker still uses the forged element for the YAccount calculation. In this case, the entries in the YAccount received and sent by the sender will be the same as shown in Figure 6.15. Still the anomaly will be traced in the signature comparison process. These steps address the hole discussed in (K. Bhargavan *et. al.* 2005).

7 Results and Discussion

In this section the results obtained for each of the phases mentioned in Section to is discussed.

7.1. Datasets Categories

Table 7.1: Data sets - Categories

Type	Remarks
Short	Size upto 50 bytes
Small	1 Record - String, Float, DateTime
Medium	25 Records - String, Float, DateTime
Large	1 Customer with 10 products each

Messaging in web services, especially in e-business, uses XML messages and several XML-based message templates that can be used between entities engaged in Business-to-Business e-business transactions over the Internet. These messages provide support for (S. Lanka *et. al* 2000) (i) purchase orders, (ii) charge orders, (iii) acknowledgments, (iv) status updating, (v) shipment notifications, and (vi) payment transactions. Based on this and categorization of data in papers such as (D. Mundy *et. al.* 2004) and (A. Ng 2006), the messages are classified into five categories for the performance evaluation, as listed in Table 7.1. The Short category represents typical short messages that are exchanged between the users (a simple messaging format with only text). The Small category represents a sophisticated version of the Short category. It represents a typical e-business message of an invoice or an inquiry form. The Medium category typically holds the details about 25 customer records. It represents a batch inquiry. The Large and Composite categories are constituted as invoice records with higher degrees of complexity and with more number of repeating record structures. These categories are designed to provide consistent test data. Among these categories, the Medium, Large and Composite categories are likely to contain repeated occurrences of data with the same structure. In the case of Small, Short and Medium categories, all elements are subjected to

verbosity reduction measures, whereas in the case of Large and Composite categories, parts of the data set that repeatedly occur are subjected to the verbosity reduction measures. The basis for the categorization is to make sure that the types of messages of different structures and complexities are used for performance comparison. The factors that are considered for data categories are size, structure and complexity.

7.2. Results

7.2.1. YASchema

The two existing YAML schema definitions Kwalify (Kwalify 2010) and Rx 2008) and the XML Schema (C. M. Sperberg-McQueen *et. al.* 2000) are used as benchmarks for comparison for YASchema. The comparison is done in two stages with varying degrees of constraints. First, the comparison is done using a commonly used set of constraints available in all formats and the second comparison is by adding the basic constraints in YASchema.

a. Verbosity: YASchema vs. Kwalify, Rx and XML Schema

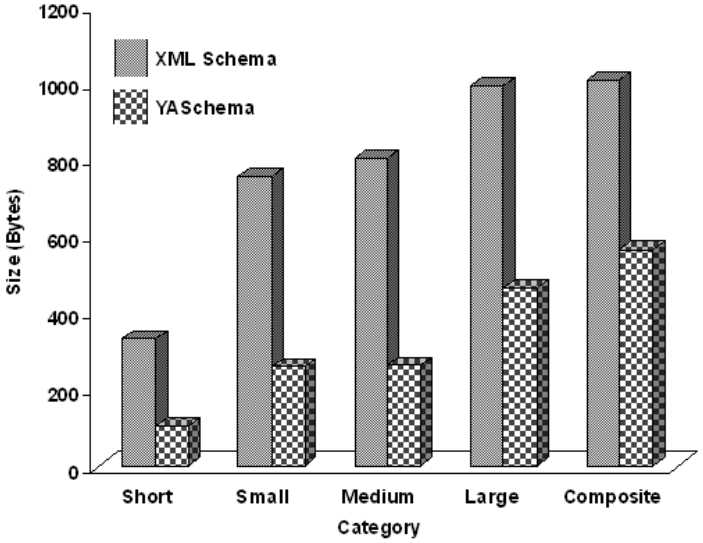

Figure 7.1: Verbosity Comparison: XML Schema vs. YASchema

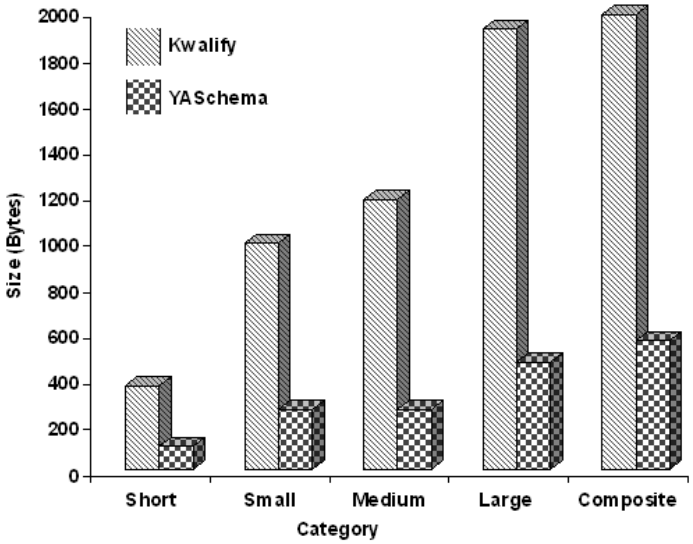

Figure 7.2: Verbosity Comparison: Kwalify vs. YASchema

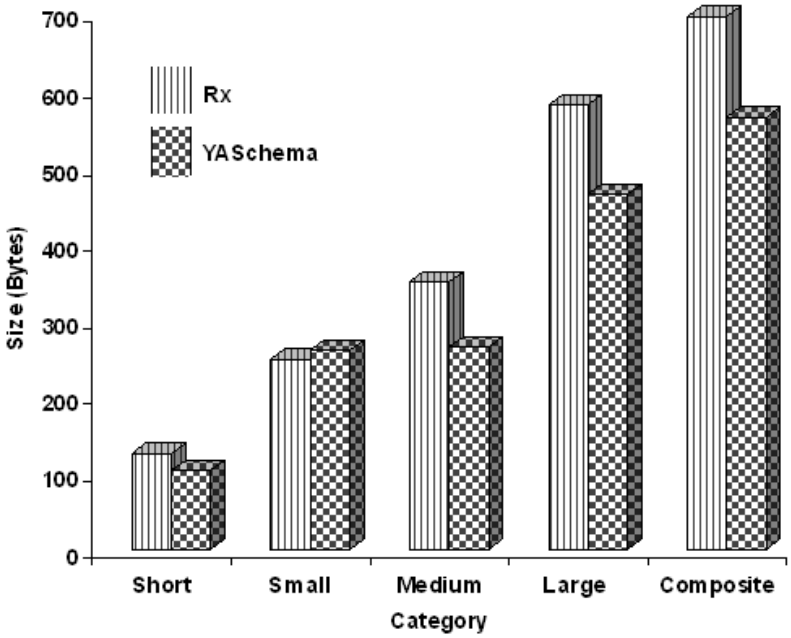

Figure 7.3: Verbosity Comparison: Rx vs. YASchema

The verbosity comparison of YASchema against XML Schema, Kwalify Schema and Rx Schema are as shown in Figures 7.1, 7.2 and 7.3, respectively. It is observed that YASchema outperforms the rest in all categories of data. Even though Rx Schema is less verbose than XML Schema and Kwalify, it is more verbose than YASchema. Thus, it is clear that while the commonly existing features are taken into consideration, YASchema is at advantage than the existing schemas in terms of the verbosity of messages.

b. Verbosity: YASchema (after the addition of specific features) vs. Kwalify, Rx and XML Schema

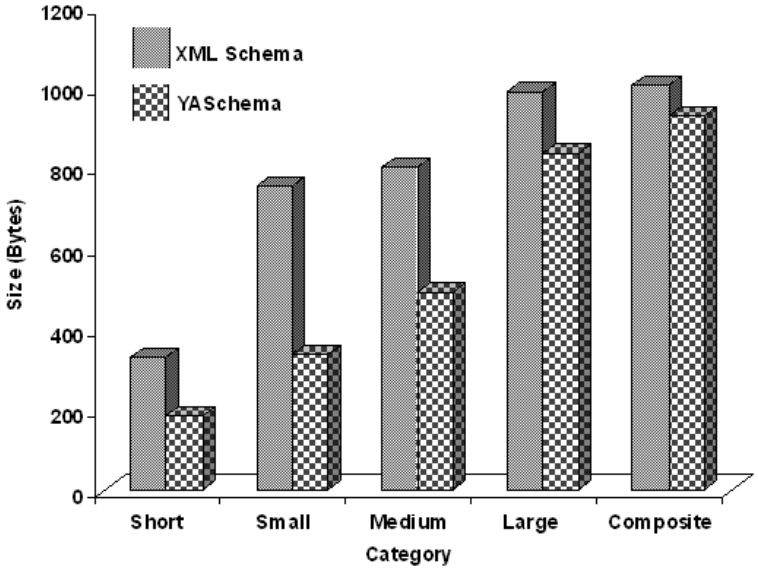

Figure 7.4: Verbosity Comparison (after the addition of specific features to YASchema): XML Schema vs. YASchema

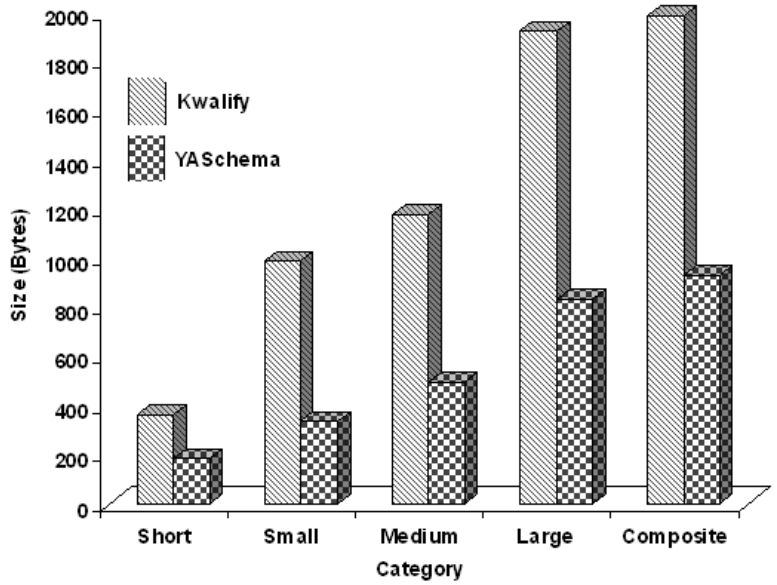

Figure 7.5: Verbosity Comparison (after the addition of specific features to YASchema): Kwalify vs. YASchema

Now, with added constraints for tag, base class, number of elements, number of occurrences and squeezable, YASchema is compared with other formats. The above constraints are needed for the thinning of YAML. These constraints do not have the equivalent constraints in XML Schema, Kwalify or Rx, except for the constraint "number of occurrences".

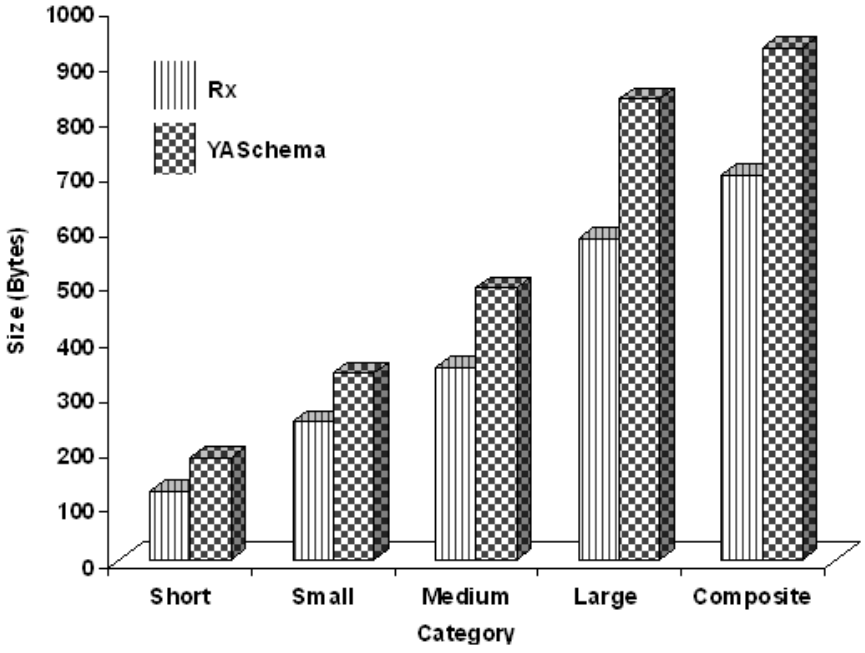

Figure 7.6: Verbosity Comparison (after the addition of specific features to YASchema): Rx vs. YASchema

It can be seen that even after the addition of more constraints, YASchema is performing better than the XML Schema and Kwalify (Figures 7.4 and 7.5, respectively). Rx is now performing better than YASchema (Figure 7.6). If similar constraints (currently not existing in Rx) are added to Rx, its verbosity will increase and then YASchema is likely to perform better.

7.2.2. TYAML

The performance comparison is done with non binary formats, and with XML and YAML, in particular. For each category of data sizes (listed in Table 7.1), the data sets are serialized in YAML and TYAML formats.

The parameters used for the performance evaluation include verbosity, content density, transmission speed, number of packets needed for the transmission, and the energy consumption. These parameters are indicative of the resource usage and the viability of the proposed scheme.

a. Verbosity: TYAML vs. other Non Binary Formats

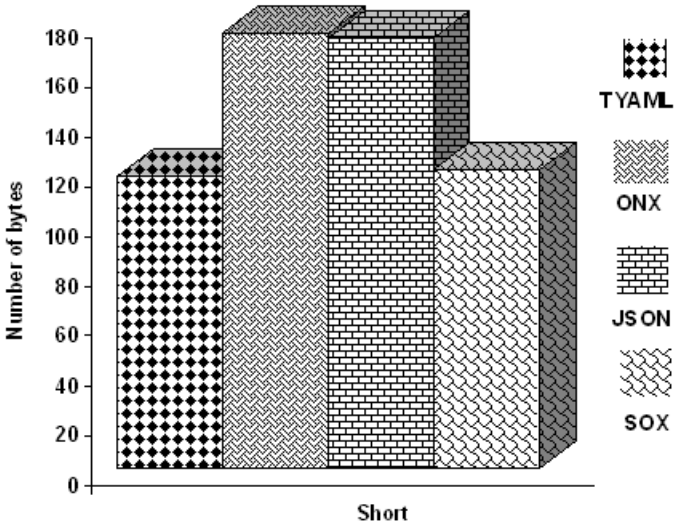

Figure 7.6: Verbosity: TYAML vs. other Non Binary Formats (*Short* category)

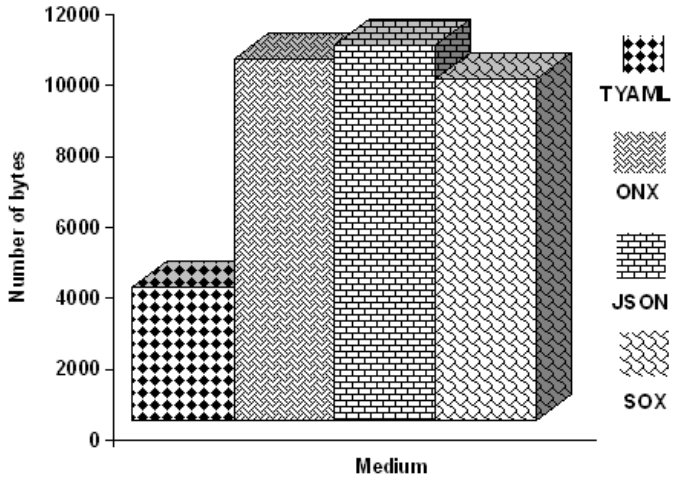

Figure 7.7: Verbosity: TYAML vs. other Non Binary Formats (*Medium* category)

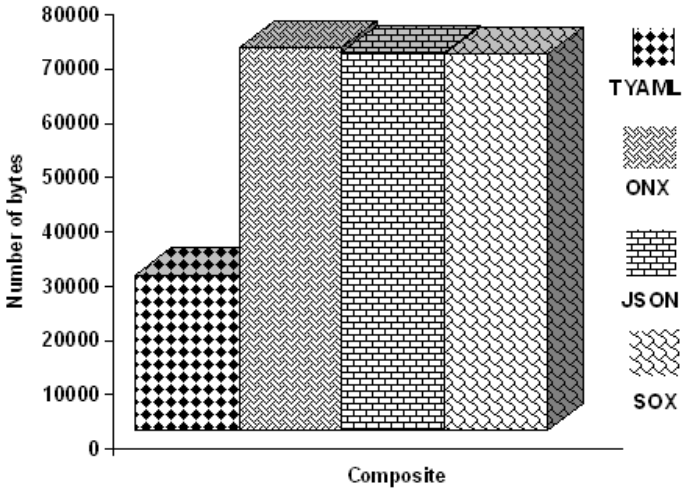

Figure 7.8: Verbosity: TYAML vs. other Non Binary Formats (*Composite* category)

Verbosity of TYAML data sets are compared against ONX, JSON and SOX formats (Figures 7.6, 7.7 and 7.8, respectively). These three formats are chosen as representatives of the alternative non-binary serializations. The observations are: (i) SOX format has a slight advantage over TYAML in Small category, and (ii) TYAML needs only less number of bytes than all other serializations, in other categories.

b. Verbosity: TYAML vs. XML (XML without end tags and extra spaces)

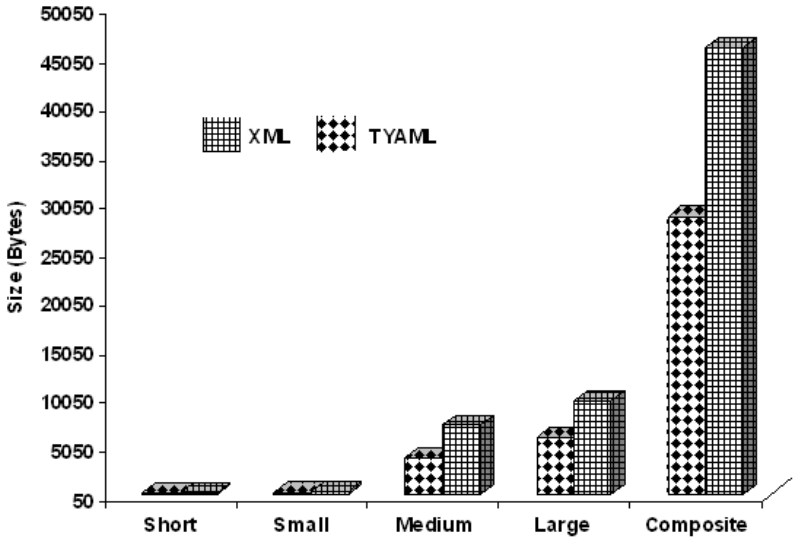

Figure 7.9: Verbosity: TYAML vs. XML (without end tags and extra spaces)

Verbosity of XML is the maximum in WFF due to the abundant use of extra spaces and end tags for all element names. If extra spaces and end tags are removed from the XML data sets, the readability of XML is affected. From Figure 7.9, it is observed that TYAML still perform better than XML. TYAML maintains this advantage by retaining the readability of the documents.

c. Verbosity: (TYAML + YASchema) vs. (XML + XML Schema)

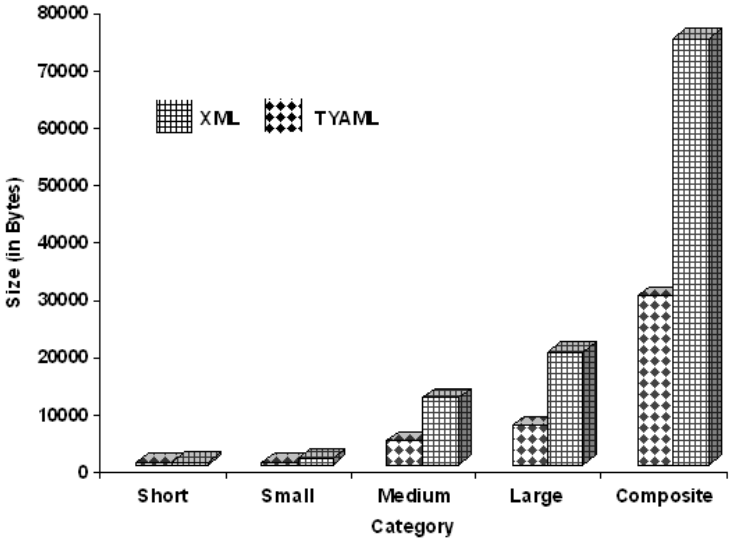

Figure 7.10: Verbosity: (TYAML + YASchema) vs. (XML + XML Schema)

Schema is part of the data being transmitted, at least for the first transmission. Also, it will have to be accompanied with the document for security applications. The verbosity of the data sets are compared with their schema sizes for XML and TYAML data sets in Figure 7.10. It is observed that the TYAML/YASchema combination performs better than the XML/XML Schema combination in all categories of data sets.

d. Verbosity: TYAML vs. XML (after compression)

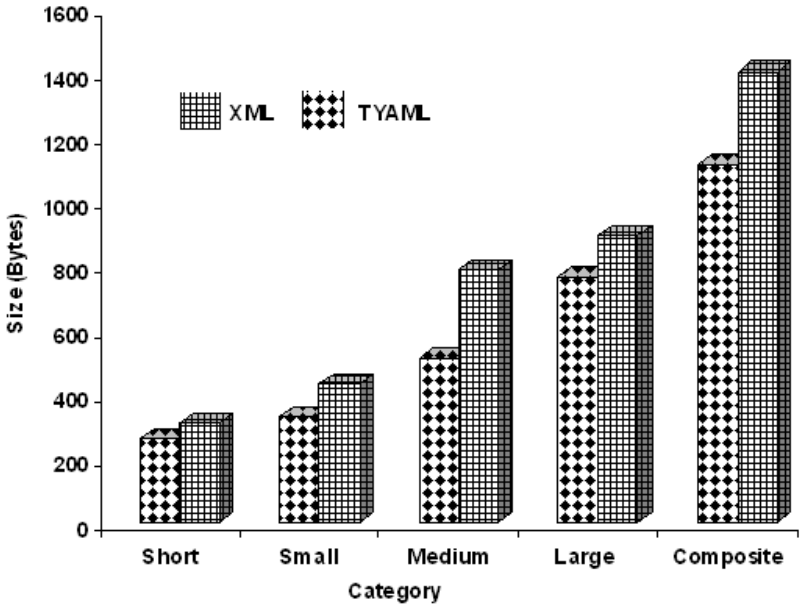

Figure 7.11: Verbosity: TYAML vs. XML (after compression)

Data, especially in Large and Composite categories may have to be compressed, depending upon the capability of the devices before data transmission. The effect of compression over the data sets is as shown in Figure 7.11. It is observed that TYAML is performing better than XML after the compression. In the Small category, the impact is minimal and it is more visible in the *Large* and *Composite* categories.

e. Verbosity: TYAML vs. YAML (both without schema)

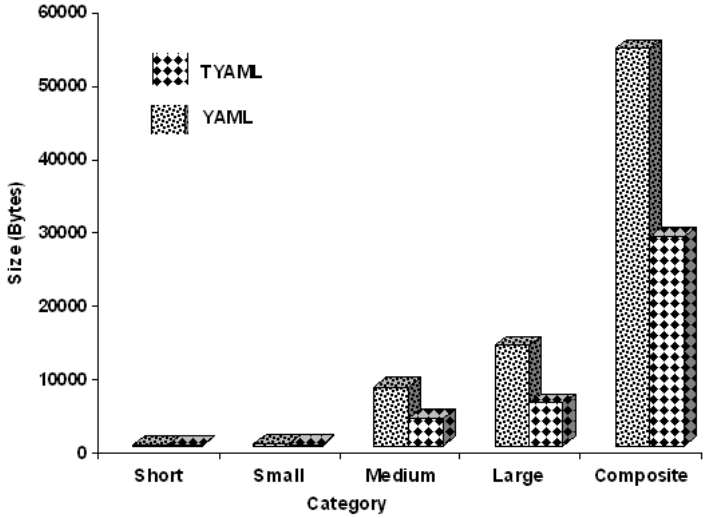

Figure 7.12: Verbosity: TYAML vs. YAML (both without schema)

TYAML and YAML are compared for verbosity in different categories. It can be observed from Figure 7.12 that TYAML has a marginal advantage in *Short* and *Small* categories than YAML. In the other categories, TYAML has a clear advantage than YAML. This demonstrates the usefulness of the thinning measures for reducing the verbosity of YAML.

f. Content Density: TYAML vs. XML

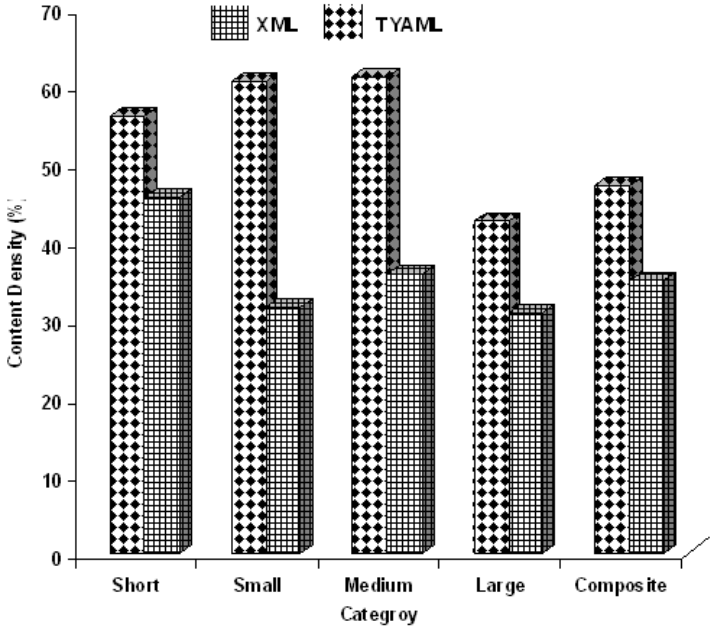

Figure 7.13: Content Density: TYAML vs. XML

Content Density (CD) is a measure to assess the amount of structure in an XML document (G. White et. al 2007). Higher the CD value, the more compact the document. A comparison of CDs of TYAML and XML is as shown in Figure 7.13. All categories of data sets report higher CD values in TYAML. Hence it can be concluded that the verbosity reduction measures have helped in increasing the CD for TYAML.

g. Transmission Speed: TYAML vs. XML

This analysis assumes ideal links with speeds of 64kbps. The actual performance will be inferior to this because of the limitations such as packet loss, retransmission, noise, etc., which are neglected here. Apart from this, limitations imposed by the protocols can also degrade the transmission speed (for e.g., the Maximum Segment Size (MSS) in Transmission Control Protocol/Internet Protocol (TCP/IP) on an Ethernet network is generally set to 1460 bytes (CISCO Tech Note 2005).

Table 7.1: Data Sets - Size in Bytes

Type	YAML	TYAML	XML
Small	145	117	191
Medium	320	260	445
Composite	2171	1144	5881

It is observed from Table 7.1 that each of the TYAML data sizes, particularly the Composite type, fits well into the limits of a single TCP/IP MSS after the two phases of verbosity reduction. The record sizes of XML in the Composite category are greater than the MSS, which forces the data to be segmented into multiple packets.

The transmission time needed for the record sets are tabulated in Table 7.2. A steady improvement in performance is observable in the case of TYAML compared to YAML and XML.

Table 7.2: Transmission Speed (in msecs) using a 64kbps link

Type	YAML	Phase I	TYAML	XML
Small	17.7	11	10.71	23.32
Medium	39.06	34.3	31.74	54.32
Composite	265.01	144.23	139.65	717.9

Table 7.3: Comparison (YAML vs. TYAML) - Transmission Speed (in msecs) using 64kbps link

Record Set	YAML (a)	TYAML (b)	Gain (a/b)
Small	17.7	10.71	1.65
Medium	39.06	31.74	1.23
Composite	265.01	139.65	1.90

The transmission speed gains of TYAML against YAML are tabulated in Table 7.3. These gains are significant as they are achieved without applying any binary encoding to the original message.

h. Packets Needed: TYAML vs. XML

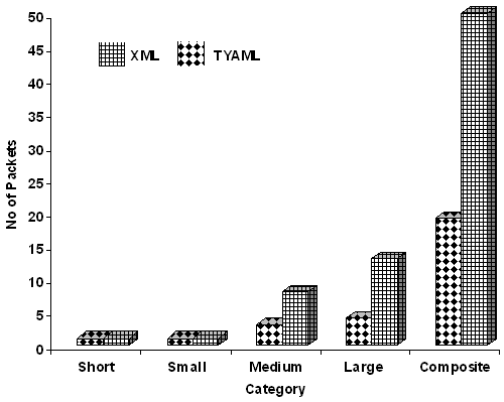

Figure 7.14: Packets Needed: TYAML vs. XML

MSS in TCP/IP on an Ethernet network is taken as the reference for the calculation of the number of packets (1460 bytes). The number of packets needed for transmission of the data is crucial in constrained wireless mobile environment because of the limitations such as packet loss, retransmission, energy consumption etc. It can be observed from Figure 7.14 that TYAML data sets require minimum number of packets, especially in the larger categories of data (Large and Composite).

i. Energy Consumption: TYAML vs. XML

Since wireless mobile devices depend upon the limited battery power attached to it, minimizing the energy consumption is very important in RCNs. The minimization efforts should consider the cost of transmitting, receiving, and even discarding of packets (L. M. Feeney *et. al* 2001). The states in which the energy is consumed are: (i) transmit (when the node is transmitting packets), (ii) receive (when the node is receiving packets), (iii) idle (when the node is waiting for any packet transfers), and (iv) sleep (when the node can neither receive nor transmit). Among these, only (i) and (ii) are the causes of useful energy consumption.

The energy model discussed in (G. Allard *et. al.* 2006) is used to calculate the energy spent at each node due to a flow. The actual energy consumption can also be affected by factors such as overhearing (reception of packets intended for other stations) and unsuccessful (colliding) transmissions. Since the objective is to find the impact of verbosity over energy consumption, only the energy consumption for transmission of data packets in a flow is considered. The energy needed for acknowledgments is ignored for simplifying the calculation. Accordingly, the base formula in (G. Allard *et. al.* 2006) is modified for the calculations. The energy consumption in transmitting packets at node N due to another node M (depending upon whether node N belongs to a flow or not, and where in the flow node M is situated) in the network is

$$E_{N=M} = T_M{>}0(T_{M=N}E_{Tpck})$$

where $E_{N/M}$ = Energy spent at node N due to node M, E_{Tpck} = Energy spent for transmission of one data packet, and $T_P = 1$ if P is true, 0 otherwise.

Table 7.4: Simulation Parameters

Parameter	Values
Channel type	Channel/WirelessChannel
Radio propagation model	Propagation/TwoRayGround
Network Interface Type	Phy/WirelessPhy
Link-/MAC-Layer	IEEE 802.11
Interface Queue	Queue/DropTail/PriQueue
Antenna model	Antenna/OmniAntenna
Max packet in IFQ	50
Topology	500x400
Transmission range (m)	250
Bandwidth	10Mbits/sec

The Network Simulator (NS-2) simulations are used to measure the packets generated during transmission. To simulate the scenario, a network configuration needs to be defined. The node configuration setting is as shown in Table 5.4. The node configuration for a wireless node sets the The Ad hoc On Demand Distance Vector (AODV) routing algorithm as its ad hoc routing protocol, the link layer type as LL and the Medium Access Control (MAC) protocol as IEEE 802.11. The queue between the MAC layer and the link layer is used. A "tcp" agent is attached to the source node, and a connection is established to a tcp "sink" agent attached to the destination node. A "ftp" traffic generator is attached to the "tcp" agent. The size of the data packets is set at 1.5 kilobytes. For the simulation, typical transmission and reception costs for the Lucent Silver card as specified in (L. M. Feeney *et. al* 2001) are used. The transmission power used is 1.3 W, and the reception power is 0.9W. Simulation time varies according to the size of the record sets as tabulated in Table 7.1. For e.g., in the case of Small data set, size in the case of YAML original is 145 bytes.

Simulation is run for 116 secs, which is the theoretical time to transfer 145 terabytes. This variation is applied to study the effect of number of packets being generated for different message categories.

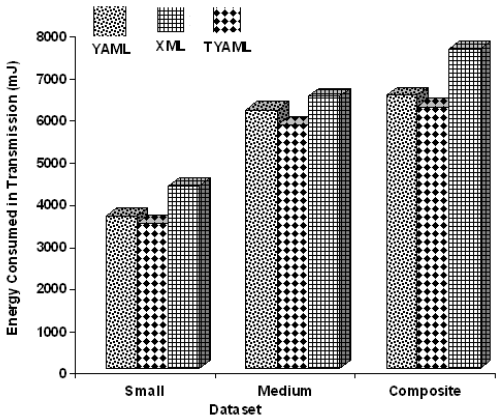

Figure 7.15: Energy Consumption while Transmission: XML vs. YAML and TYAML

It is observed from Figure 7.15 that (i) the Small data set reports major gains than the other two categories, (ii) TYAML displays gains in all categories, (iii) XML reports the maximum consumption in all categories, and (iv) TYAML demonstrates better performance than the plain YAML format in all categories. This shows that the verbosity reduction (of the message) results in the reduction in the number of packets generated. It also helps in reducing the network traffic, energy spent for message transmission and depletion of memory space.

7.2.3. TYAML Message Level Security – General Security Specifications

The verbosity of TYAML security specifications are compared with that of XML. The Advanced Encryption Standard (AES) version with 128 bits is used for the encryption and Digital Signature Algorithm (DSA) is used for signature processing. For signcryption operations, AES is used for the encryption part and Secure Hash Algorithm - 1 (SHA-1) is used for hashing. The processed values and the related parameters are represented as per the generic specifications for verbosity comparison.

a. Verbosity of Encryption Specification: TYAML vs. XML

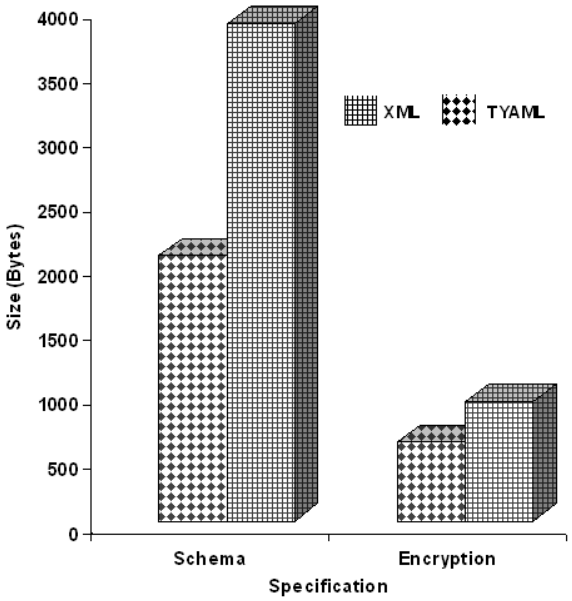

Figure 7.16: Verbosity of Encryption Specification: TYAML vs. XML

The verbosity of the XML specification and associated XML Schema (for encryption) are compared with the corresponding TYAML specification and YASchema in Figure 7.16. It is observed that the TYAML/YASchema combination performs better than the XML/XML Schema combination.

b. Verbosity of Signature Specification: TYAML vs. XML

The verbosity of the signature specifications (XML and TYAML) and the corresponding schemas (XML Schema and YASchema) are compared in Figure 7.17. It is observed that the TYAML/YASchema combination performs better than the XML/XML Schema combination.

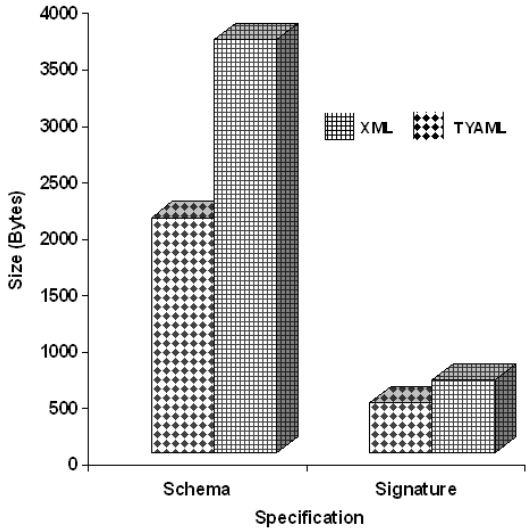

Figure 7.17: Verbosity of Signature Specification: TYAML vs. XML

c. Verbosity of Signcryption Specification: TYAML vs. XML

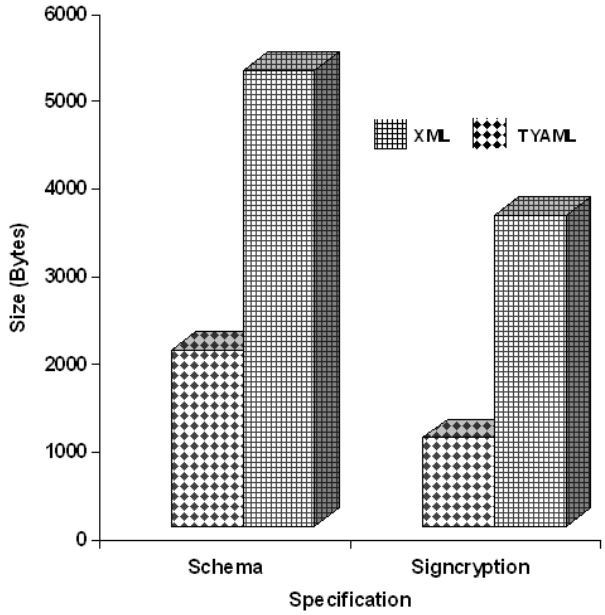

Figure 7.18: Verbosity of Signcryption Specification: TYAML vs. XML

The verbosity of the XML signcryption specification and XML Schema are compared with the corresponding TYAML signcryption and YASchema in Figure 7.18. It is observed that the TYAML/YASchema combination performs better than the XML/XML Schema combination.

7.2.4 Enhancing TYAML Format for ECC Compatibility

The performance comparison of TYAML specifications are done with XML. A key size of 283 bits is used for the EC algorithms. The parameters and other details of the process and processed values are represented as per the specifications in TYAML and XML, for comparison.

a. Verbosity for ECIES Specification: TYAML vs. XML

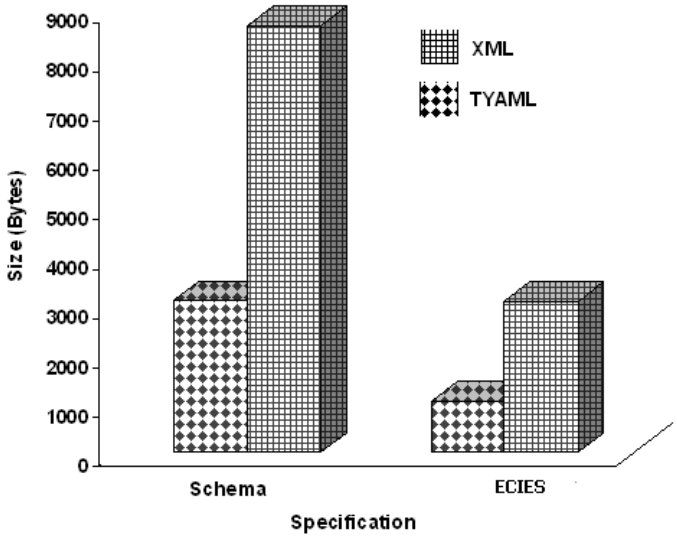

Figure 7.19: Verbosity for ECIES Specification: TYAML vs. XML

The verbosity of the XML specification and XML Schema are compared with the corresponding TYAML specification and YASchema in Figure 7.19. It can be seen that TYAML/YASchema combination is performing better than the XML/XML Schema combination.

b. Verbosity for ECDSA Specification: TYAML vs. XML

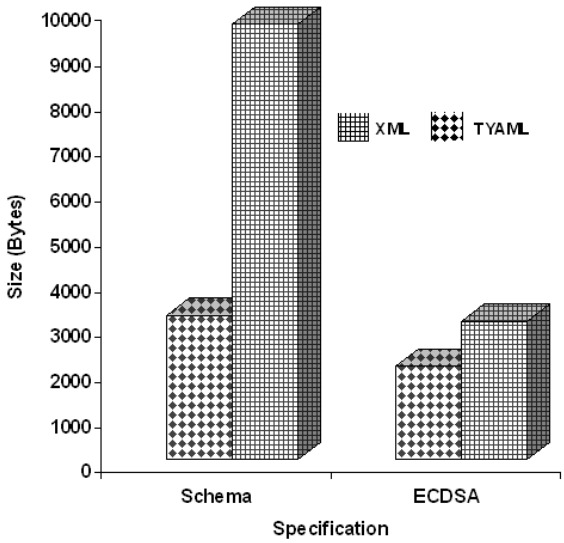

Figure 7.20: Verbosity for ECDSA Specification: TYAML vs. XML

The verbosity of XML specification and XML Schema are compared with the corresponding TYAML specification and YASchema as shown in Figure 7.20. In this case also, TYAML/YASchema combination is performing better than the XML/XML Schema combination.

c. Verbosity of TYAML: Generic vs. EC based Specifications

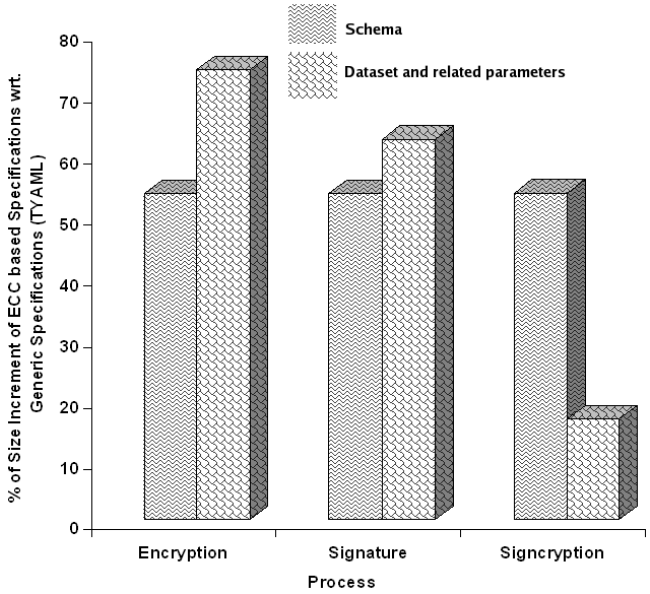

Figure 7.21: Verbosity of TYAML: Generic vs. EC based Specifications

A comparison of generic and EC based specifications are done to study the effect of verbosity in TYAML formats. For generic specifications, RSA with 3072 bit key size is used for equivalent level of security of 283 bits of ECC. The verbosity of generic and EC based specifications in TYAML format are compared in Figure 7.21.

It can be observed that the introduction of EC based specifications resulted in the increase of verbosity in the cases of encryption, signature and signcryption. This is primarily due to the increase in the number of elements in the EC based specifications. There is an increase in the number of parameters needed for EC based algorithms and processed values (encrypted/signed/signcrypted values). Anyway, this cannot be taken as a drawback due to the fact that when there are a set of information to be transmitted, then the "common" details such as the algorithm, parameters and associated YASchema need to be sent only once. Hence, for the subsequent transfers, only the processed information need to be sent.

In spite of the fact that ECC based TYAML specifications are more verbose than the RSA based generic specifications, they are of value due to the proved advantages of ECC based algorithms.

7.2.5 Lightweight Schemes to Check Rewriting Attacks in TYAML

In this section, effect of the increased verbosities due to the introduction of data structures used for checking the rewriting attacks is compared.

a. Verbosity: Data set vs. Metafile

Table 7.5: Verbosity: Data set vs. Metafile (in Bytes)

Parameter	Dataset	Metafile
Small	117	19
Small	215	68
Medium	3727	91
Large	5872	105
Composite	28642	112

Metafile is an additional payload used for verbosity reduction. The sizes of data sets and the corresponding metafiles are as shown in Table 7.5. It can be seen that the metafile size is relatively very small compared to the data set size. Thus, it can be concluded that the introduction of the metafile structure does not have a considerable impact on the overall payload to be transmitted.

b. Verbosity: WS Policy, SOAP Account and YAccount

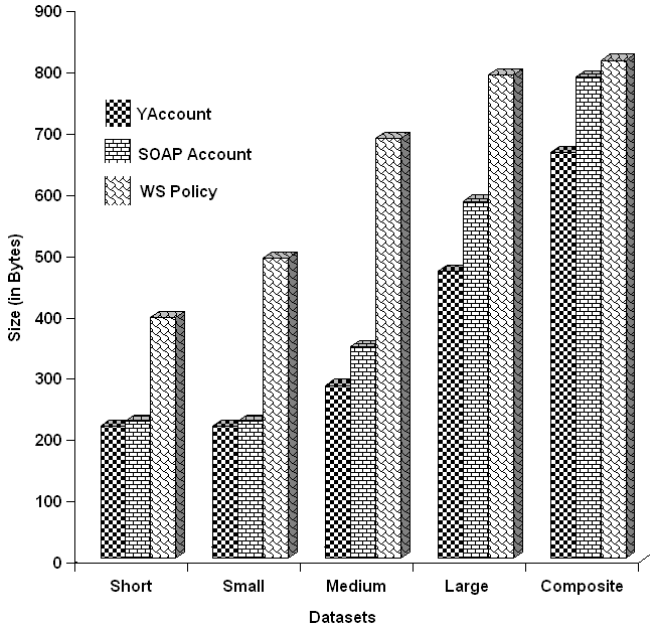

Figure 7.22: Verbosity: Comparison of WS Policy, SOAP Account and YAccount

The WS Policy file, SOAP Account and YAccount are compared in Figure 7.22. It is observed that (i) WS Policy files are larger in size compared to SOAP Account and YAccount (ii) for smaller categories, SOAP Account and YAccount are comparable in size, and (iii) YAccount files are the shortest in all categories of data sets.

c. Verbosity: WS Policy vs. YASchema

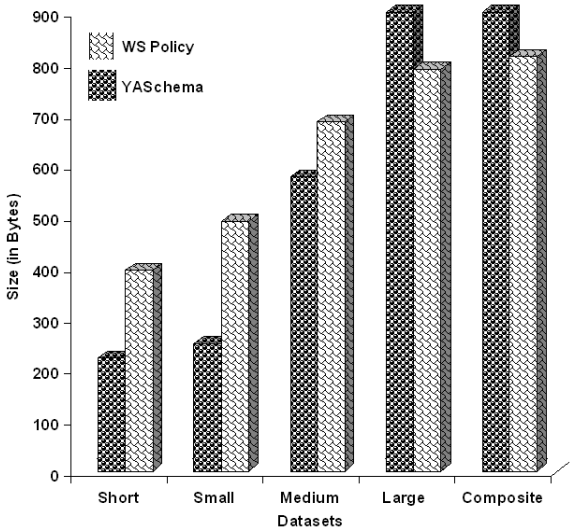

Figure 7.23: Verbosity: WS Policy vs. YASchema

Figure 7.23 compares the verbosities of WS Policy files and YASchema. For Short, Small and Medium data sets, YASchema takes less number of bytes than WS Policy. But for the remaining categories, WS Policy file outperforms YASchema. YASchema has additional entries other than the policy assertions like data types and order of occurrences. In spite of that YASchema is performing better than WS Policy files in the first three categories. The drawback in the larger categories stresses for the need for careful choice of the policy rules in YASchema.

YASchema has additional entries other than the policy assertions like data types and order of occurrences. In spite of that YASchema is performing better than WS Policy files in the first three categories. The drawback in the larger categories stresses for the need for careful choice of the policy rules in YASchema.

d. Relative Verbosity: WS Policy, YASchema and YAccount (with respect to the data set)

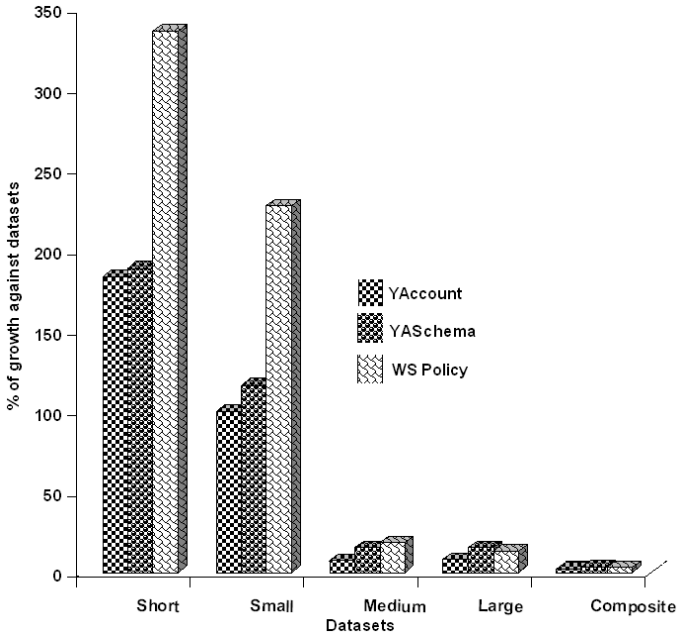

Figure 7.24: Verbosity (Relative to data set size): WS Policy, YASchema and YAccount

The growth % of WS Policy, YASchema and YAccount constructs with respect to the size of the data sets are as shown in Figure 7.24. It is observed that YAccount is more advantageous in all categories, particularly in the smaller categories of data sets, and YASchema, in spite of the additional constructs, performs better than the WS Policy file in smaller categories (*Short*, *Small* and *Medium*).

8 Conclusion

The alternative method for XML messaging is presented in this paper. This can address the "inflation problem" of information in the Internet/Intranet, and particularly in the RCNs. The lightweight approach reduces the verbosity and facilitates secure and trusted information, and is a right step towards conserving the resource usage.

YAML, a lightweight data format, is used as the basic framework. A schema definition called YASchema, makes the proposed format schema aware. It is observed that YASchema needs less storage space than the existing schemes. The proposed lightweight message format TYAML is derived from the base YAML format through two phases of thinning. The thinning processes are DFS based methods, aimed at reducing the verbosity of the base format. TYAML inherits all the data types and properties of YAML. Performance analysis showed that TYAML reduces the message size by 56.66% and 41.78%, respectively, in comparison with XML and plane YAML.

Message level security is added to the TYAML format, introducing security specifications and schema definitions for encryption, signature and signcryption. Together with these specifications, TYAML found to be less verbose than the corresponding XML specifications. The generic specifications are refined further to make it compatible

with ECC with a view of facilitating secure TYAML in the constrained mobile wireless devices and RCNs. ECC based schemes such as IES, DSA and signcryption are chosen for this.

Trusted exchange of information is ensured with the introduction of refined signature specifications with indicators to the actual location of the signed objects. Since this measure has compatibility issues, a separate accounting structure is devised to prevent the rewriting attacks. Policy rules are embedded into the YASchema, so that there is no need to maintain separate policy files. A layered architecture is also proposed, combining the YASchema and the accounting information to check the rewriting attacks more efficiently. All the proposed structures are less verbose than the corresponding XML structures.

Performance evaluation in terms of verbosity, transmission speed, packets needed, energy consumed and content density confirmed that TYAML performs better than the existing popular formats. The limitations of TYAML include compatibility issues for the signature (considering the abundant use of the XML format) and the need for careful framing of the policy assertions for larger messages.

TYAML format can perform well as an alternative to XML, in its applications such as in configuration files, log files, inter-process messaging, chatting, cross-language data sharing, etc.

The proposed future work include incorporating measures such as "schema hardening" to make the format more resilient to rewriting attacks and incorporating attribute based access control mechanism into TYAML/YASchema, as envisaged in policy languages like eXtensible Access Control Markup Language (XACML).

References

H. Li (2000). XML and industrial standards for electronic commerce, Knowledge and Information Systems (KAIS), 2(3): 487–497.

W. S. Hanslo and K. J. MacGregor (2003). Using XML messaging for wireless middleware communication, SATNAC '03: Proceedings of the South African Telecommunication Network Applications Conference, George, South Africa. North-Holland Publishing Co.

J. Kangasharju et. al (2007). XML messaging for mobile devices: From requirements to implementation," The International Journal of Computer and Telecommunications Networking, 51(16): 4634–4654.

H. H. Lee and W.S. Lee (2010). Consistent collective evaluation of multiple continuous queries for filtering heterogeneous data streams, Knowledge and Information Systems (KAIS), 22(2): 185–210.

H. R. Elliotte (2003). XML Bible, New York, NY, USA: John Wiley & Sons, Inc.

J. Kangasharju et. al (2005). Requirements and design for XML messaging in the mobile environment," Second International Workshop on Next Generation Networking Middleware, 29–36.

J. Kangasharju et. al (2008). XML security with binary XML for mobile web services, International Journal of Web Services Research (IJWSR), 5(3): 1–19.

W. Ng et. al (2006). XCQ: A queriable XML compression system, Knowledge and Information Systems (KAIS), 4: 421–452.

M. P. Michael (2005). Energy Awareness for Mobile Devices, Research Seminar on Energy Awareness, University of Helsinki.

R. Potlapally et. al (2006). Heterogeneous grid computing for energy constrained mobile device, IEEE Transactions on Mobile Computing, 5: 128-143.

G. Q. Gu and J.-Z. Lu (2008). Some issues of computer networks: Architecture and key technologies, Journal of Computer Science and Technology (JCST), 21:5: 708-722.

Paradiso and Starner (2005). Energy scavenging for mobile and wireless electronics, IEEE Pervasive Computing, 4: 18-27.

L. Snol (2010). The rise of the ultra-low-cost mobile phone, Available at http://www.itworld.com/personal-tech/130741/the-rise-ultra-low-cost-mobilephone.

C. Kounavis (2010). Access to mobile networks available to over 90% of world population 143 countries offer 3G services, Available at http://www.wirelessresearch.eu/archives/349.

A. Tella and D. Y. Oluyemisi (2010). The future of ICT in developing world: Forecasts on sustainable solutions for global development, Indian Journal of Library and Information Science, 4(2): 115-131.

J. C. Aker and I. M. Mbiti (2010). "Mobile phones and economic development in Africa", Journal of Economic Perspectives, 24(3): 207-232.

H. Artail et. al. (2009). A distributed mobile database implementation on pocket PC mobile devices communicating over Bluetooth, Journal of Network and Computer Applications (JNCA), 32: 96-115 [Online], Available at http://portal.acm.org/citation.cfm?id=1460932.1461106

OFDM (2008). Orthogonal frequency-division multiplexing, Available at http://en.wikipedia.org/wiki/OFDM.

Y.X. Lai et. al (2008). PEJA: Progressive energy-efficient join processing for sensor networks, Journal of Computer Science and Technology (JCST), 23(6): 957-972.

O. Ben-Kiki et. al (2010). YAML aint Markup Language (YAML) version 1.2, 3rd edition, Available at http://www.yaml.org/spec/1.2/spec.html.

D. Crockford (2006). The application/JSON media type for Javascript Object Notation (JSON), Available at: http://tools.ietf.org/html/rfc4627.

S. Jacobs (2005). Open Node Syntax Version 0.6.9, Available at: http://www.seairth.com/web/onx/onx.html.

SOX (2002). Simple Outline XML: SOX (2002), Available at: http://www.langdale.com.au/SOX/.

K. G. Clark (2002). Look ma, no tags, Available at: http://www.xml.com/pub/a/2002/07/24/yaml.html.

YAML (2007). YAML, Available at: http://en.wikipedia.org/wiki/YAML.

N. Walsh (1999). Understanding XML schemas, Available at: http://www.xml.com/pub/a/1999/07/schemas/index.html?page=1.

C. M. Sperberg-McQueen and Henry Thompson (2000). W3C XML Schema, Available at http://www.w3.org/XML/Schema.

Kwalify (2010). Kwalify, Available at http://www.kuwata-lab.com/kwalify/.

Rx (2008). Rx - Simple, Extensible Schemata, Available at: http://rjbs.manxome.org/rx/index.html.

Bertino and Martino (2006). Security in SOA and web services, IEEE SCC '06: IEEE International Conference on Services Computing. New York, NY, USA, Pp. 41-55.

Y. Zheng (1997). Digital Signcryption or how to achieve cost (signature & encryption) << cost (signature) + cost (encryption), Lecture Notes in Computer Science (LNCS), 1294: 165-179, Available at: citeseer.ist.psu.edu/zheng97digital.html.

Zheng (1997), Signcryption and its applications in efficient public key solutions, ISW 97: Proceedings of 1st International Information Security Workshop, Pp. 291-312, URL: citeseer.ist.psu.edu/zheng97signcryption.html.

T. Imamura, B. Dillaway and E. Simon (2002). XML encryption syntax and processing, W3C Recommendation, Available at: http://www.w3.org/TR/xmlenc-core/.

M. Bartel et. al. (2008). XML signature syntax and processing (second edition), W3C Recommendation, URL: http://www.w3.org/TR/xmldsig-core/.

B. LaMacchia et. al (2000). Review Draft, Available at: http://lists.w3.org/Archives/Public/xml-encryption/ 2000Dec/att-0024/01-XMLEncryption v01.html.

W. Chou (2003). Elliptic Curve Cryptography and its applications to mobile devices, Tech. Rep., University of Maryland, College Park, Maryland.

N. Potlapally et. al. (2003). Analyzing the energy consumption of security protocols, ISLPED '03: Proceedings of the 2003 International Symposium on Low Power Electronics and Design, Pp. 30-35.

Certicom-Research (2000), Standards for efficient cryptography, SEC 1: Elliptic Curve Cryptography, Certicom Research, Tech. Rep.

S. Bajaj et. al. (2004), Web services policy framework (WS-Policy), Available at: http://www.ibm.com/developerworks/library/specication/ws-polfram/.

A. Nadalin et. al (2004). Web services security: SOAP message security 1.0 (WS-Security 2004), Available at: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-messagesecurity-1.0.pdf.

M. A. Rahaman et. al. (2006). An inline approach for secure SOAP requests and early validation, OWASP '06: Proceedings of the OWASP Europe2006 Conference, Pp. 19-33.

K. Bhargavan et. al. (2005). An advisor for web services security policies, SWS '05: Proceedings of the 2005 workshop on Secure web services, New York, NY, USA, ACM Pp. 1-9.

A. Benameur et. al. (2008). XML rewriting attacks: Existing solutions and their limitations, IADIS 2008: International Conference on Applied Computing. IADIS Press, Available at: http://arxiv.org/abs/0812.4181.
S. K. Sinha and A. Benameur (2008). A formal solution to rewriting attacks on SOAP messages, SWS '08: Proceedings of the 2008 ACM workshop on Secure web services, New York, NY, USA, ACM, Pp. 53-60.

M. McIntosh and P. Austel (2005). XML signature element wrapping attacks and countermeasures, SWS '05: Proceedings of the 2005 workshop on Secure web services, New York, NY, USA, ACM, Pp. 20-27.

M. A. Rahaman and A. Schaad (2007). SOAP-based secure conversation and collaboration, ICWS 2007: Proceedings of the 2007 IEEE International Conference on Web Services, IEEE Computer Society, Pp. 471-480.

S. Gajek et. al (2007). Breaking and fixing the inline approach, SWS '07: Proceedings of the 2007 ACM workshop on Secure web services, New York, NY, USA, ACM, Pp. 37-43.

P P Abdul Haleem and M P Sebastian (2009). An Efficient Approach for Thinning Wireless Mobile Messages Alternative Approach for Slicing Down the Message Size and Enhancing the Security in Wireless Mobile Networks, ICIC Express Letters, Japan, 3(1): 87-94, ISSN 1881-803X.

P P Abdul Haleem and M P Sebastian (2011). Optimizing Resource Consumption for Secure Messaging in Resource Constrained Networks, Innovations in Mobile Multimedia Communications and Applications: New Technologies, IGI Global, USA, Pp 21-36, DOI: 10.4018/978-1-60960-563-6, ISBN13: 9781609605636.

D. Mundy and D. W. Chadwick (2004). An XML alternative for performance and security: ASN.1, IT Professional, 6: 30-36, [Online]. Available at: http://portal.acm.org/citation.cfm?id=1435621.1436239

G. White et. al (2007). Efficient XML interchange measurements note, Available at : http://www.w3.org/TR/eximeasurements/.

CISCO Tech Note (2005). Adjusting IP, MTU, TCP, MSS, and PMTUD on Windows and Sun systems, August 2005, Available at: http://www.cisco.com, CISCO Technical Note, Document ID: 13709

P P Abdul Haleem and M P Sebastian (2012). An Energy Conserving Approach for Data Formatting and Trusted Data Exchange in Resource Constrained Networks, Knowledge and Information Systems, Springer-Verlag, London, Vol 32, Issue 3, Pp. 559-587, 2012, DOI 10.1007/s10115-011-0450-0, Print ISN: 0219-1377

L. M. Feeney and M. Nilsson (2001). Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, INFOCOM 2001, Proceedings of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, 3: 1548 - 1557.

G. Allard et. al. (2006). Evaluation of the energy consumption in MANET, Lecture Notes in Computer Science (LNCS), 4104/2006: 170-183.

S. Lanka and P. Parikh (2000). XML shredding, Midterm Project, New York University, Courant.
A. Ng (2006). Evaluation of the serialisation and deserialisation performance of Table Driven XML, Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW '06), Washington, DC, USA, IEEE Computer Society, Pp. 178-185. [Online]. Available at: http://portal.acm.org/citation.cfm?id=1116162.1116345.