XML Security in XML Data Integrity, Authentication, and Confidentiality

Baolong Liu

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Doctor of Philosophy

School of Computing and Engineering
University of Huddersfield

May 2010

Acknowledgements

I would like to thank my first supervisor, Dr. Joan Lu for her supervision. I also acknowledge my second supervisor, Prof. Jim Yip, for his valuable advice to my research.

Many thanks give to Dr. Andrew Campton and Dr. Diane Kitchin for their valuable comments to my progress report.

I want to thank the School of Computing and Engineering at the University of Huddersfield for providing the great opportunity of study and facilitating me throughout the research.

I want to thank my parents, Xinhui Sun and Yingxiang Liu who never give up providing supports to my study. More than anyone else, I want to thank my wife, Yi Guo, for her patience and came to stay with me during my study. I would like to thank all my friends in China for their kind encouragements.

Abstract

The widely application of XML has increasingly required high security. XML security confronts some challenges that are strong relating to its features. XML data integrity needs to protect element location information and context-referential meaning as well as data content integrity under fine-grained security situations. XML data authentication must satisfy a signing process under a dependent and independent multi-signature generation scenario. When several different sections are encrypted within the XML data, it cannot query the encrypted contents without decrypting the encrypted portions. The technologies relating to XML security demand further development.

This thesis aims to improve XML security relative technologies, and make them more practicable and secure. A novel revocation information validation approach for X.509 certificate is proposed based on the XML digital signature technology. This approach reduces the complexity of XKMS or PKI systems because it eliminates the requirement for additional revocation checking from XKMS or CA. The communication burden between server and client could be alleviated.

The thesis presents the context-referential integrity for XML data. An integrity solution for XML data is also proposed based on the concatenated hash function. The integrity model proposed not only ensures XML data content integrity, but also protects the structure integrity and elements' context relationship within an XML data. If this model is integrated into XML signature technology, the signature cannot be copied to another document still keeping valid.

A new series-parallel XML multi-signature scheme is proposed. The presented scheme is a mixed order specified XML multi-signature scheme according to a dependent and independent signing process. Using presented XML data integrity-checking pool to provide integrity-checking for decomposed XML data, it makes signing XPath expression practicable, rather than signing XML data itself.

A new labeling scheme for encrypted XML data is presented to improve the efficiency of index information maintenance which is applied to support encrypted XML data query processing. The proposed labelling scheme makes maintenance index information more efficient, and it is easy to update XML data with decreasing the number of affected nodes to the lowest. In order to protect structural information for encrypted XML data, the encrypted nodes are removed from original XML data, and structural information is hidden.

A case study is carried out to demonstrate how the proposed XML security relative approaches and schemes can be applied to satisfy fine-grained XML security in calibration certificate management.

List of Figures

List of Tables

List of Abbreviations

CA	Certification Authority
CI	Content Integrity
CRI	Context Referential Integrity
CRL	Certificate Revocation List
CRS	Certificate Revocation System
CRT	Certificate Revocation Tree
DD	Document Dispatcher
DL	Discrete Logarithm
DoD	Department of Defence
DOM	Document Object Model
DSA	Digital Signature Algorithm
DSI	Discontinuous Structural Index
DSS	Digital Signature Standard
DTD	Document Type Definition
HMAC	Hash-based Message Authentication Code
IETF	Internet Engineering Task Force
ITU	International Telecommunication Union
MAC	Message Authentication Code
NLBILS	Number List based Interval Labeling Scheme
OASIS	Organization for the Advancement of Structured Information Standards
OCSP	Online Certificate Status Protocol
PI	Processing Instruction
PKI	Public Key Infrastructure
RSA	Rivest, Shamir and Adleman
SA	System Authority
SC	Signature Collector
SHA	Secure Hash Algorithm
SOAP	Simple Object Access Protocol
STI	Structure Integrity

XDD	XML Data Decomposition
XDM	XPath data model
XHTML	eXtensible Hypertext Markup Language
XKMS	XML Key Management Specification
X-KISS	XML Key Information Service Specification
X-KRSS	XML Key Registration Service Specification
XML	eXtensible Markip Language
XSL	eXtensible Stylesheet Language
XSLT	XSL Transformations
URIs	Uniform Resource Identifier Strings
W3C	World Wide Web Consortium

List of Mathematical Symbols

$h(M)$	The result of a hash computation on message M using an approved hash function
K	A secret key used in symmetric cryptography
$A_s(M,K)$	An encryption algorithm (symmetric cryptography)
$A_s^{-1}(C,K)$	A decryption algorithm (symmetric cryptography)
C	Encryption result (Cipher text)
K_{pub}	A public key
K_{priv}	A private key
$A_e(M,K_{pub})$	An encryption algorithm (Asymmetric cryptography)
$A_d(C,K_{priv})$	A decryption algorithm (Asymmetric cryptography)
V	An element set of XML data
X_D	An XML data
p	A prime number that defines the $GF(p)$ and is used as a modulus in the operations of $GF(p)$
q	A prime factor of $p-1$
G	A broadcast signing group
x_i, y_i	Private key and public key for each member in G
X,Y	The private and public key pair for G
Γ	A set of decomposed sub-message of M
r	One component of a DSA digital signature
s	One component of a DSA digital signature
R	One component of multi-signature
S	One component of multi-signature
$Cert$	An X.509 digital certificate
C_{xml}	An X.509 digital certificate based on XML data format
SN	A set of revoked X.509 digital certificate

T	A time-stamp for signature
v	An XML node in XML data
v_r	The node of XML data root
I_i	An initial vertex of an edge i in directed graph
T_i	A terminal vertex of an edge i in directed graph
φ	A directed graph
$L(\varphi)$	A set of edges in graph φ
$E(\varphi)$	A set of vertices in graph φ
ϕ	An empty set
SG	A series-parallel signing group
st	Any sub-tree in an XML data
R_k	One component of multi-signature in a broadcast subgroup
S_k	One component of multi-signature in a broadcast subgroup
NL	Number list

List of Contents

Chapter 1 Introduction

With eXtensible Markup Language (XML) (Bray et al., 2008) widely applied to different areas, security is necessary to be integrated into XML solutions. XML is based on text format designing and has tree structure, and it is easier to access portions of data using XPath (Berglund et al., 2007). It is natural that data integrity, data authentication, information confidentiality, and other security benefits should be applied to entire XML data or portions of XML data. Traditional security systems can only handle the entire document or message. A new requirement of security is needed. XML security should provide security assurance of information represented using XML format. XML security must be combined with XML data features to keep the advantages and flexibility of XML while integrating essential security technologies. This is very important in XML-based protocols, e.g. the protocol of Simple Object Access Protocol (SOAP) (Gudgin et al., 2007), and it relies on XML as its message format to provide message negotiation and transmission.

Based on the new features and security requirements of XML data, the reasons of having XML security mechanism for XML are: XML data provides fine-grained access, and it needs to sign, and encrypt portions of XML data rather than in whole, e.g. multi-signature to different portions of XML data. Traditional security technologies cannot be used directly within XML data and do not provide methods relative to XML data content management (Sun and Li, 2005), e.g. using XPath to locate portions of XML data content or specifying XML data content using Uniform Resource Identifier string (URIs). Traditional security technologies play an important role in XML security to provide a set of necessary security algorithms and techniques which can be deployed in XML security. However, the representation of traditional security is not suitable to XML security (Sun and Li, 2005). The format of traditional security technologies is in binary, and it requires specialized software for interpretation and extracting the security information (Hirsch, 2002).

The specifications relative to XML security published by W3C define the basic framework and rules that can be utilized by across applications. The basic idea for XML security is to reuse the algorithms, approaches, concepts, and techniques of traditional security systems. The tools and methods which can be used to support extensible integrating XML have been introduced. This idea enables existing infrastructures and across security deployment to interoperate with XML security. By using existing technologies and XML relative tools, XML security minimizes additional applications to satisfy security requirements for XML data (Hirsch, 2002).

There are four major topics relative to XML security.

- XML data integrity

  XML data integrity ensures that both XML data structure and data content are not destroyed or changed during transition or storage, and this can be ensured using hash value checking. This may happen when XML data is transmitted over the internet, such as from a server to a browser, and XML data is stored in a database system, or processed by intermediaries.

- XML data authentication

  Identity authentication provides assurance about the claimed identity of an entity. In other words, it is to prove the claimed identity to a verifier. XML data authentication is that the entity is responsible for the creation of a set of XML data, which is the whole XML data or portion of XML data, is the one claimed. XML data authentication is usually ensured using digital signature.

- XML data confidentiality

  XML data confidentiality ensures that XML data structure, data content or other sensitive information in XML data may only be accessed by legitimate parties. Confidentiality is generally associated with access control mechanisms or encryption technologies. Compared to access control mechanism, the encryption technology is essential in application, for example, utilizing encryption technology to protect data transmitted in an untrusted channel.

- Accountability

Accountability is used to record the responsibility of the individuals belonging to the organization for which a policy regarding XML data security has been established (Brandt and Bonte, 2000). This can be ensured using access control mechanism, such as assigning the role to control user's accountability.

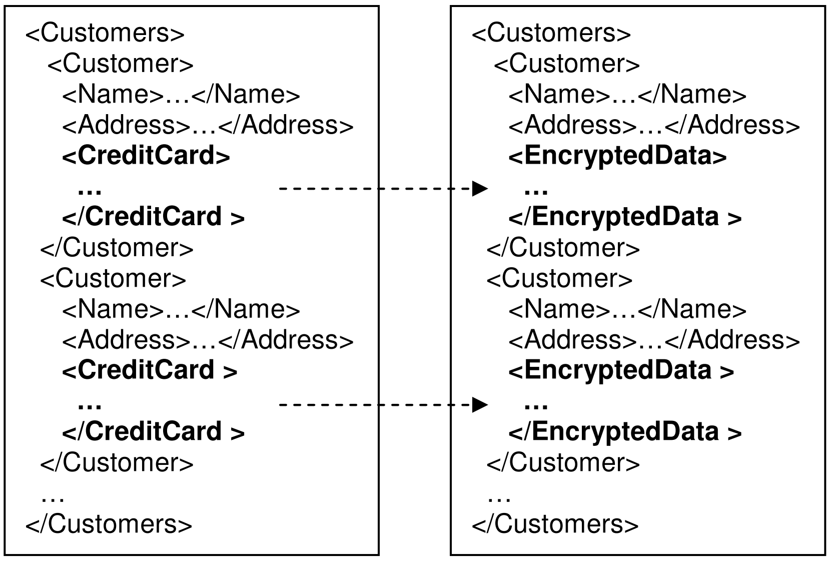
These topics are not separate. XML key management (Hallam-Baker and Mysore, 2005) provides the basic key requirements for XML data integrity, authentication, and confidentiality. XML data integrity is used to generate hash value which actually is signed in XML signature. XML data integrity is the fundamental for XML data authentication. Based on the specification of "Decryption Transform for XML Signature" (Hughes et al., 2002), XML signature and XML encryption can be implemented independently, or encrypt entire or portions of a signed XML data.

The thesis focuses on XML security in XML data integrity, authentication, and confidentiality. In particular, it mainly focuses on improving technologies relative to XML data integrity, XML signature, and XML encryption.

1.1 Motivation and challenges

XML security specifications published by W3C have addressed XML data integrity, XML digital signature, and XML encryption (Bartel et al., 2008; Imamura et al., 2002). XML data integrity is the basis for XML signature generation. W3C adopts DOM-HASH (Maruyama et al., 1999) to generate hash values for ensuring XML data integrity. Without considering XML data structure integrity, and context-referential integrity, it will result that a signature can be copied to another document still keeping valid signature verification. XML data integrity is the main reason leading to limitation in XML signature. Existing integrity solutions for XML data have not considered the features relative to XML data. Most of XML data integrity models describe controls for achieving hash values, but no attempt is made to define a model for XML data integrity combined with XML data features.

XML signature specification supports to build a single signature or multiple signatures. With XML signature, users can sign the same content or different portions of XML data. With the single digital signature generation and verification, XML signature is useful and practicable. When several users participate in multi-signature generation, XML signature specification cannot handle a multi-signature generation with a mixed dependent and independent way since it only supports a parallel multi-signature generation. Existing XML multi-signature schemes only provide broadcast (parallel) signature-generation scenarios. It cannot satisfy the signing process under a dependent multi-signature-generation situation.



```
<Customers>
  <Customer>
    <Name>…</Name>
    <Address>…</Address>
    <CreditCard>
      ...
    </CreditCard >
  </Customer>
  <Customer>
    <Name>…</Name>
    <Address>…</Address>
    <CreditCard >
      ...
    </CreditCard >
  </Customer>
  …
</Customers>
```

```
<Customers>
  <Customer>
    <Name>…</Name>
    <Address>…</Address>
    <EncryptedData>
      ...
    </EncryptedData >
  </Customer>
  <Customer>
    <Name>…</Name>
    <Address>…</Address>
    <EncryptedData >
      ...
    </EncryptedData >
  </Customer>
  …
</Customers>
```

(a) (b)

Figure 1.1 An example of XML encryption result

XML encryption is used to ensure XML data confidentiality, and it provides a flexible approach to encrypt any portions of XML data. Figure 1.1(a) shows a document of customer information. In order to protect credit card information, the elements <CreditCard> have to be encrypted. By using XML encryption technology, the encrypted result is shown in Figure 1.1(b). XML encryption supports to encrypt portions of XML data, and this can be found in Figure 1.1(b). However, the query of the information which resides in the cipher blocks has not been addressed in XML encryption technology when XML data is encrypted. For instance, only the cipher blocks in Figure 1.1 (b) can answer information relating to bank information.

Existing approaches are index-based scheme for encrypted XML data querying. Management of index information is not considered by researchers. It is a time-consuming task to maintain index information for frequently changed XML data. It needs to consider efficiency of index information updating when an index scheme is deployed (Ünay and Gündem, 2008). In addition, the structural information leakage has not been considered within existing solutions.

1.2 Aim and objectives

The aim of this research is to improve XML security related technologies, and make it more practicable and secure. In order to reach this aim, there are several objectives:

- To present an approach to easily validate X.509 digital certificate revocation information.

- To present the XML data integrity requirements combined with XML data features.

- To present a solution for XML data integrity protection, and improve the efficiency of hash value generation for XML data.

- To build an XML multi-signature scheme, which is a mixed-signing order scheme including both series and parallel to satisfy a dependent and independent signing process.

- To present an index mechanism which can exactly locate a block of cipher text while users submit a query, and eliminate unnecessary decryption when query an encrypted XML data. The index information can be updated efficiently. In addition, this index mechanism will not disclose the structural information of XML data.

- To implement a prototype of proposed approaches and schemes combining with existing XML security specifications.

1.3 Research approach

This research started with an extensive literature review of the sate-of-the-art in XML security in XML data integrity, entity authentication, XML data authentication and XML confidentiality.

- Approaches or schemes development

To address the aim and objectives highlighted in section 1.2, this research decided to use the concatenated hash function to model XML data integrity. The traditional hash functions, such as SHA1 or SHA2 cannot protect the relationship of different XML elements in XML data, e.g. parent-child relationship, sibling relationship. However, the traditional hash functions can be used to generate hash value for individual XML element. It needs a mechanism to assemble the hash value of individual XML element to protect the relationship between different XML elements. Similar to Merkle hash function (Merkle, 1989), concatenated hash function also focuses on hash value generation processing for tree-based data structure. Merkle hash function is based on binary tree, in contrast, concatenated hash function is based on arbitrary tree structure, and it is more suitable to handle XML data. Digital signature is used to ensure XML data authentication in a hierarchical network. Index-based mechanism is adopted in encrypted XML data query processing.

- Correctness proving

The correctness of proposed approaches or schemes is proved to confront the proposed aim and objectives. The correctness proofs of XML data integrity approach are expressed by three theorems. As proposed XML multi-signature scheme, the correctness is proved by using strict mathematic method.

- Security analysis

The security of proposed approaches or schemes is also analyzed. The proposed XML data integrity approach is based on concatenated hash function. The security issue in proposed integrity approach is avoided because the approach is based on collision-resistant one-way hash function. The proposed XML multi-signature scheme is based on discrete logarithm (DL) problem, so it has a high security. The thesis also describes how the proposed encrypted XML data query scheme avoids inference attack.

- Testing and evaluation

The proposed approaches or schemes are strictly tested and verified to evaluate its performance and efficiency over existing solutions. The testing and evaluation cases are generated from XMark and DBLP dataset. The testing mainly focuses on correctness and functionality proving of proposed approaches or schemes. The evaluation mainly focuses on efficiency comparison between existing solutions. In this research, testing and evaluations were continuously being undertaken during every major phase to ensure that it has a good functionality and stability. Researchers in the School of Computing & Engineering have given some advices for evaluation, as well as whether it meets the aim and objectives of the research. Further revisions for the proposed approaches or schemes might take place based on the feedback from these tests and evaluation.

- Prototype implementation

  A prototype for XML security was developed in the C#.net language. This has facilitated the refinement and completion of the approaches and schemes with improved understanding on some implementation issues. The system also served as a demonstration of capabilities of the final system with feedbacks from various tests.

1.4 Arrangements of this thesis

Chapter 1 introduces the motivation and challenges of XML security in XML data integrity, authentication and confidentiality. The aim and objectives are described. The research approaches are also demonstrated.

Chapter 2 introduces the background knowledge relative to traditional security technologies, the common tools which has been deployed in XML security specifications, especially XML security specifications or standards including XML key management, XML signature syntax and processing, XML encryption syntax and processing published by W3C or OASIS.

Chapter 3 is literature review. The contents of literature review mainly focuses on existing ideas and solutions relative to revocation information validation approaches for X.509 digital certificate, XML data integrity, the theories and schemes of multi-signature for ensuring XML data authentication. The investigation of schemes for index-based encrypted XML data query processing is also illustrated.

XML key management is the basic requirements for XML security technologies. In order to alleviate the burden of revoked certificate validation, Chapter 4 introduces an improved X.509 digital certificate based on XML signature technology. The improved X.509 digital certificate can be utilized combined with XML key management specification with a high efficiency.

Chapter 5 analyzes the XML data integrity. Based on presented XML data integrity requirements, an integrity model CSR (‘C’ for content integrity, ‘S’ for structure integrity, and ‘R’ for context-referential integrity) for XML data is proposed in this chapter. The functionality of this model is tested to meet the XML data integrity requirements. The efficiency of this model is also evaluated.

Chapter 6 introduces the series-parallel signing group, which intends to generate a multi-signature with a dependant and independent signing process. In order to make XML data integrity-checking possible, the XML data integrity checking-pool is presented in this section. Based on Lu’s XML multi-signature scheme (Lu and Chen, 2004), combined with series-parallel signing group and XML data integrity-checking pool, a series-parallel multi-signature scheme for XML data authentication is proposed.

Chapter 7 introduces a number list based interval labeling scheme for XML data. Based on presented labeling scheme, an index-based scheme for encrypted XML data query processing is proposed with considering the efficiency of index information maintaining.

Based on approaches and schemes proposed in previous chapters, Chapter 8 implements a case study of XML security in calibration certificate management. The system architecture for calibration certificate management is introduced. The detailed algorithms or processes relative to XML security are also described.

Chapter 9 focuses on the summary of this research and contribution to knowledge. A discussion for the future work is also described.

Chapter 2 Background

This chapter firstly introduces the traditional security techniques and protocols, and then briefly introduces the XPath language. XML security specifications published by W3C and OASIS are also described. In particular, XML key management, Canonical XML, XML signature and XML encryption specifications established by the W3C are discussed.

2.1 Security techniques and protocols

2.1.1 Cryptographic hash functions

Cryptographic hash functions are modelled based on one-way functions, which is easy to generate an authentication code. A cryptographic hash function is

$$y = h(M) \tag{2.1}$$

where, h is a hash function, M is a message, y is a hash value (Stallings, 2006). A cryptographic hash function has some properties as follows.

- The input of h can be a block of data of any size
- h produces a fixed-length output, and it is called hash value, or message digest.
- For any given value y, it is computationally infeasible to find M such that $h(M) = y$. This is referred to as the one-way property.
- Given a message M, it is difficult to find $M^{'}$ such that $M \neq M^{'}$ and $h(M) = h(M^{'})$ (Stallings, 2006).
- It is difficult to find any pair (M, $M^{'}$) such that $h(M) = h(M^{'})$ (Stallings, 2006).

Widely used cryptographic hash functions are MD5, and the SHA series of functions. However, the collision has been found in hash function MD5, SHA-1 and RIPEMD-160, and they are now considered insecure. SHA-256 and other hash functions are believed to be secure. A summary of existing hash algorithms are listed in Table 2.1.

Table 2.1 Hash algorithms

Hash algorithms	Block size (bits)	Output size (bits)	Rounds	Collision
MD5 (Rivest, 1992)	512	128	64	2^{39} (Wang et al., 2005a)
SHA-1 (FIPS180-2, 2002)	512	160	80	2^{63} (Wang et al., 2005b)
SHA-256 (FIPS180-2, 2002)	512	256	64	No
SHA-512 (FIPS180-2, 2002)	1024	512	80	No
RIPEMD-160 (Dobbertin et al., 1996)	512	160	80	2^{51} (Mendel et al., 2006)

It is shown that MD5, SHA-1 and RIPEMD-160 are not good choice for generating message digest because of collision attacks (Cid 2006). Although the drawback of SHA-256 and SHA-512 is certainly slower than MD5 and SHA-1, until now, no collision has been found in SHA-256 and SHA-512 as shown in Table 2.1. SHA-256 and SHA-512 can be used as a replacement for MD5 and SHA-1. NIST (National Institute of Standards and Technology) also recommends using SHA-256 in practical applications. Based on this fact, hash functions still can be used to ensure security in applications.

2.1.2 Symmetric cryptography

Bijection is used as the basis of cryptography, for encryption (Smart, 2010). Bijection is a mathematical function which is one-to-one (injective) and onto (surjective). In particular, if $f : X \to Y$ is a bijection, then for all $y \in Y$, there is a unique $x \in X$ such that $f(x) = y$. This unique x is given by the inverse function $f^{-1} : Y \to X$.

If f is an encryption transformation, then f^{-1} is the corresponding decryption transformation. If a non-injective function were used as an encryption transformation, it would not be possible to decrypt to a unique plain text.

The traditional way of encrypting messages is called symmetric key encryption. Symmetric-key algorithms use a single secret key which must be shared and kept private by both sender and receiver for both encryption and decryption. To use a symmetric encryption scheme, the sender and receiver must securely share a key in advance.

This symmetric encryption scheme assumes that the sender and the recipient share the knowledge of a secret key K and an encryption algorithm A_s to the message M . A message can be encrypted by

$$C = A_s(M, K) \tag{2.2}$$

The secret message C is decrypted by applying the inverse algorithm A_s^{-1} to the secret message C with the key K:

$$M = A_s^{-1}(C, K) \tag{2.3}$$

Symmetric-key algorithms can be divided into stream ciphers and block ciphers. Stream ciphers encrypt the bits of the message one at a time, and block ciphers take a number of bits and encrypt them as a single unit. Table 2.1 summarizes the commonly used symmetric-key algorithms.

Table 2.2 Symmetric-key algorithms

Algorithms	Block size (bits)	Key size (bits)
DES (Kammer, 1999)	64	56
AES (NIST, 2001)	128	128, 192, 256
Triple DES (Barker, 2004)	64	168

The commonly used block ciphers are Data Encryption Standard (DES), Advanced Encryption Standard (AES), and Triple DES as shown in Table 2.2. The DES is a block cipher that was selected by the National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the United States in 1976 (Kammer, 1999). It is based on a symmetric-key algorithm that uses a 56-bit key. DES is now considered to be insecure for many applications, and this is chiefly due to the 56-bit key size being too small. The AES is a symmetric-key encryption standard adopted by the U.S. government (NIST, 2001). The standard comprises three block ciphers, AES-128, AES-192, and AES-256. Each of these ciphers has a 128-bit block size, with key sizes of 128, 192, and 256 bits, respectively. Triple DES (3DES) applies the DES cipher algorithm three times to each data block. Triple DES was designed to provide a relatively simple method of increasing the key size of DES to protect against brute force attacks, without designing a completely new block cipher algorithm (Barker, 2004).

2.1.3 Asymmetric cryptography

The distinguishing technique used in public key cryptography is the use of asymmetric key algorithms, where the key used to encrypt a message is not the same key used to decrypt it. Each user has a pair of cryptographic keys—a public key K_{pub} and a private key K_{priv}. The private key is kept secret, while the public key may be widely distributed. Messages are encrypted with the recipient's public key and can only be decrypted with the corresponding private key. The keys are related mathematically, but the private key cannot feasibly be derived from the public key (Diffie, 1976). Table 2.3 lists the usage of key pairs with different security purposes.

Table 2.3 A pair of cryptographic keys

Security purpose	Kind of key
Send an encrypted message	Use the receiver's public key
Decrypt an encrypted message	Use the receiver's private key
Send a signed message (signature generation)	Use the sender's private key
Verify a signature (and authenticate the sender)	Use the sender's public key

In asymmetric cryptography, each user has a private key K_{priv}, and a public key K_{pub}. A plain-text message M encrypted with the public key K_{pub} can only be decrypted with the private key K_{priv}. The cryptographic algorithm A_e is used for encryption, and A_d is used for decryption. In some public key encryption schemes, e.g. RSA, the same algorithm can be used for both encryption and decryption (i.e. $A_e = A_d$). The encryption and decryption is performed

$$C = A_e(M, K_{pub}) \tag{2.4}$$

$$M = A_d(C, K_{priv}) \tag{2.5}$$

Similarly, a message M' that is encrypted with the private key K_{priv} can only be decrypted with the public key K_{pub}:

$$C' = A_e(M', K_{priv}) \tag{2.6}$$

$$M' = A_d(C', K_{pub}) \tag{2.7}$$

There are two main branches of public key cryptography are public key encryption and digital signature.

- Public key encryption
  A message encrypted with a receiver's public key cannot be decrypted by anyone expect a possessor of the matching private key. This is used for confidentiality. RSA (which stands for Rivest, Shamir and Adleman who first publicly described it) is the first algorithm known to be suitable for encryption as well as signing. RSA is believed to be secure given sufficiently long keys and the use of up-to-date implementations. The RSA algorithm involves three steps: key generation, encryption and decryption (Stallings, 2006).

  Step 1: key generation
  1. Select p, q, where p and q are both prime, and $p \neq q$.
  2. Calculate $n = p \times q$, where n is used as the modulus for both the public and private keys.
  3. Calculate $\varphi(pq) = (p-1)(q-1)$, where φ is Euler's totient function.

4. Select an integer e such that $1 < e < \varphi(qp)$, and $\gcd(\varphi(n), e) = 1$ (this means that e and $\varphi(pq)$ share no divisors other than 1).
5. Calculate $d = e^{-1} \bmod((p-1)\times(q-1))$, where d is kept as the private key exponent.
6. $K_{pub} = \{e, n\}$. The public key consists of the modulus n and the public exponent e. $K_{priv} = \{d, n\}$. The private key consists of the modulus n and the private exponent d which must be kept secret.

Step 2: encryption

Each message $M\,(M < n)$, the ciphertext C corresponding to:

$$C = M^{e} \bmod n \tag{2.8}$$

Step 3: decryption

The original message M can be recovered by using private key exponent d by the formula 2.9 computation:

$$M = C^{d} \bmod n \tag{2.9}$$

- Digital signature

  A digital signature (Pfleeger, 1997) is an emulation of a real, physical signature. A digital signature is a proof that the sender makes the message, and everyone can identify the message belonging to the sender with the sender's public key. Public key encryption algorithms are suited to digital signatures, like RSA. An encryption using a private key of the user serves as a signature that only the owner of the private key can be generated, and everyone with the public key can verify. Another commonly used algorithm for digital signature is Digital Signature Algorithm (DSA). The DSA is based on the difficulty of computing discrete logarithms. The DSA algorithm involves three steps: key generation, signing, and verifying (NIST, 2006).

Step 1: key generation

The key pair is generated for a set of domain parameters p, q, g, where p, q are two large prime numbers such that $q \mid (p-1)$, g is the generator of the cyclic group of order q in Z_p^* (selects an element $h \in Z_p^*$ and computes $g = h^{(p-1)/q} \bmod p$ such that $g \neq 1$). User's private key is a randomly selected integrity $x (0 < x < q)$. User's public key y is calculated by using $y = g^x \bmod p$.

Step 2: signing

Let H be the hashing function, such as SHA1, and M the message to be signed.

1. Generate a random per-message value $k (0 < k < q)$.
2. Calculate $r = (g^k \bmod p) \bmod q$, and $s = (k^{-1}(H(M) + xr)) \bmod q$.
3. The signature is (r, s)

Step3: verifying

1. Reject the signature if either $0 < r < q$ or $0 < s < q$ is not satisfied.
2. Compute $w = (s)^{-1} \bmod q$.
3. Compute $u_1 = (H(M) \times w) \bmod q$, and $u_2 = (r \times w) \bmod q$
4. Compute $v = ((g^{u_1} \times y^{u_2}) \bmod p) \bmod q$.
5. The signature is valid if $v = r$.

2.1.4 Public key infrastructure

The definition of the Public Key Infrastructure (PKI) is "*the set of hardware, software, people, policies and procedures needed to create, manage, store, distribute, and revoke public key certificates based on public key cryptography*" in the IETF PKIX Roadmap (Arsenault and Turner, 1999; Goyal, 2004b). PKIX (Public Key Infrastructure (X.509)) is an Internet Engineering Task Force (IETF) effort to standardize such a PKI.

2.1.5 X.509 certificate

In the terminology of PKIX, a public key certificate is defined as "*a data structure containing the public key of an End-Entity and some other information, which is digitally signed with the private key of the certificate authority* (*CA) which issued it*" (Arsenault and Turner, 1999).

A public key certificate is applied to provide evidence of a legitimate key, and it is a document containing serial number, public key information, and identity – such as the name of a person or email address, and these information is signed by a trusted authority, e.g. a CA (Schneier, 1995; Georgiadis et al., 2002).

One of the most popular standards for public key certificates is contained in the ITU (International Telecommunication Union) X.509 standard. The X.509 standard (ITU, 1997; Ford and Baum, 1997) provides an authentication framework with public key certificate distribution to the X.509 directory standards series (ITU, 1997). The X.509 standard specifies how identity authentication information is generated, illustrates how identity authentication information can be retrieved from a server, and also defines approaches in which applications may utilize the identity authentication information to perform authentication verification process (Georgiadis et al., 2002).

ITU-T X.509 was firstly published in 1988 as part of the X.500 Directory recommendations, and it defines a standard certificate format. The certificate format in the 1988 standard is called the version 1 (v1) format. X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends RSA. The digital signature scheme is assumed to require the use of a hash function. However, the standard does not dictate a specific hash algorithm. The 1988 recommendation included the description of a recommended hash function; this algorithm has since been shown to be insecure and was dropped from the 1993 recommendation (Housley et al., 2002).

When X.500 was revised in 1993, resulting in the version 2 (v2) format. The X.509 version 2 format does not convey all of the information that recent design and implementation experience has shown to be needed. Ford lists the following requirements not satisfied by version 2 (Ford, 1995).

- The Subject field is inadequate to convey the identity of a key owner to a public-key user.
- The Subject field is also inadequate for many applications, which typically recognize entity by an e-mail address, a URL.
- There is a need to indicate security policy information. This makes an application easily to relate a certificate to a given policy.
- It needs to limit the damage, which may result from a faulty or malicious CA, by setting constraints of a particular certificate.
- It is important to be able to identify different keys used by the same owner at different time.

In response to these new requirements, the X.509 certificate version 3 (v3) was developed. The v3 version extends the v2 format by adding provision for additional extension fields. Particular extension field types may be specified in standards or may be defined and registered by any organization (Housley et al., 2002).

In the X.509 structure, a trusted CA assigns a distinguished name (DN) to the user who holds a public key certificate (Schneier, 1995). The CA issues certificates signed under the CA's private key. When a user A wishes to communicate with a user B, A obtains B's certificate from a directory (or by another method) and verifies its authenticity with the CA's public key.

2.2 XPath expressions

In order to retrieve information from encrypted XML data, XML Path Language (XPath) should be deployed within a query. The XPath language is a specification for addressing nodes of an XML data in XPath data model (XDM) proposed by W3C. Using XPath, an XML document as well as atomic values, e.g.

integers, strings, and booleans are represented as a tree structure. It also offers an expressive way to locate nodes within the tree (Berglund et al., 2007).

A path expression contains of a series of path steps, which is separated by "/" or "//", and usually beginning with "/" or "//", where, "/" denotes parent-child operator, and "//" denotes ancestor-descendant operator. Such a path can be either absolute path, which starts from the root of the XML data tree, or relative one starting with known context nodes (Berglund et al., 2007).

A wildcard operator ("*" or "@") is also allowed to be used in an XPath. A wildcard operator can match any element or attribute node of the context node in XML data tree. In addition, a predicate expressed in square brackets ("[ ]") can also be used to refine the selection operation in XPath expression (Jonker and Feng, 2008).

2.3 XML Key Management

Public key provides trustworthy of client's identity, and it can be used to establish secure communication between different clients. Public key information is provided by a digital certificate based on PKI. Deployment of PKI is a complex task because the PKI must reflect the real word trust relationship which is complex and subtle (Hallam-Baker and Ford, 2001). The complexity limited the application of PKI.

In order to support a client to make use of public key management, and further support public key management in XML digital signature and XML encryption, the W3C and Internet Engineering Task Force (IETF) published specification of XML Key Management (Hallam-Baker and Mysore, 2005).

The XML Key Management Specification (XKMS) provides public key management to support XML applications (Hirsch and Just, 2003). XKMS is not a substitute for a PKI. It is expected that a client can make use of key management functionality. It is also expected that the deployments of XMKS allows clients to interoperate with X.509 PKI already deployed (Hallam-Baker and Ford, 2001). In addition, the XKMS is suitable for use in conjunction with the W3C Recommendations for XML Signature and XML Encryption (Hallam-Baker and Mysore, 2005).

The XML Key Management Specification consists of two parts: the XML Key Information Service Specification (X-KISS) and the XML Key Registration Service Specification (X-KRSS).

2.3.1 X-KISS

X-KISS specifies a protocol to resolve public key information contained in element of <ds: KeyInfo> (Hallam-Baker and Ford, 2001). This element is applied to identify a public key in XML signature. X-KISS provides two services as follows:

- Locate service

  The locate service resolves a <ds: KeyInfo> element but does not require the service to make an assertion concerning the validity of the binding between the data in the <ds: KeyInfo> element (Hallam-Baker and Mysore, 2005). When a client submits a locate request, the locate service processing is shown in Figure 2.1 as described by W3C (Hallam-Baker and Mysore, 2005).

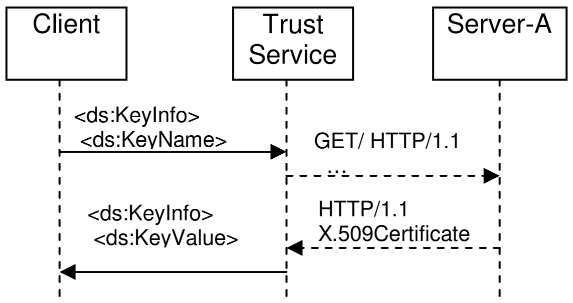


Figure 2.1 Location service provides name resolution
(Hallam-Baker and Mysore, 2005)

A recipient receives a signed XML data from another user which specifies user own certificate but not the key value. The recipient can obtain the key value from the XKMS service by using the locate service. The recipient sends the element of <ds:KeyInfo> to the location service, and the locate service returns the corresponding <KeyValue> to the recipient.

- Validate service

The validate service allows the client to obtain an assertion specifying the status of the binding between the public key and the relative identity information, such as a name or a set of other attributes (Hallam-Baker and Mysore, 2005). Unlike locate service, the validate service makes sure the data returned is valid and bound to the same public key. Figure 2.2 described by W3C shows the validate service.

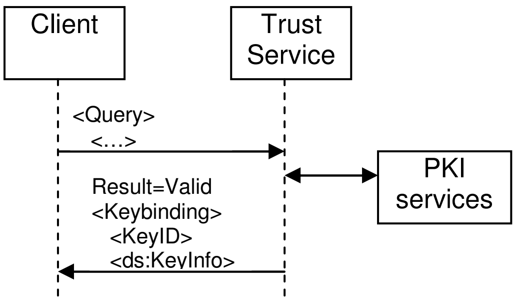


Figure 2.2 Key validation service
(Hallam-Baker and Mysore, 2005)

When a user holds a signed XML data and relative X.509 certificate, it is not known whether the certificate is trustworthy. In order to determine this, the certificate needs to be sent to an XKMS validate service, and the service returns back the validating results. The validate service establishes a certificate trust path, and then validates each certificate in the path against the relevant CRL. If all certificates in the path are valid, the validate service responses a positive result. The client is only informed the validation results by validate service, and shielded from this complex process. Although this approach reduces the complexity for a client, it will increase the burden of the server because of frequently user validation request.

2.3.2 X-KRSS

XML Key Registration Service Specification (X-KRSS) specifies a protocol for a trust service that permits management of information bound to a public key (Hallam-Baker and Mysore, 2005). X-KRSS supports all the following functionalities associated with the public key:

- Registration

The registration service supports binding a public key to a specific identity, such as a name, email address. The key pair can be generated by either client or registration service. If the key pair is generated by client, it also needs additional information to prove possession of private key (Hallam-Baker and Mysore, 2005). The registration request should be authenticated by the client, and this can be done by a digital signature.

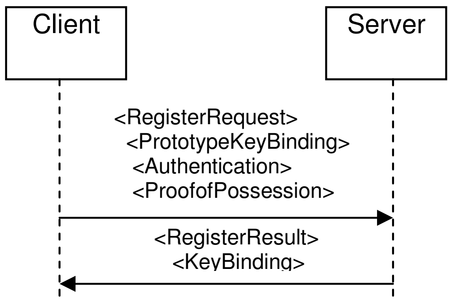


Figure 2.3 Registration of key binding (Hallam-Baker and Mysore, 2005)

A client generates a key pair and registers the public key. The identifier is the email address. The request message should contain the elements of <ProofofPossession> and <Keyauthentication> as shown in Figure 2.3 by W3C (Hallam-Baker and Mysore, 2005).

- Reissue

The registration service permits clients to reissue key bindings previously issued (Hallam-Baker and Mysore, 2005). The reissue process is similar to the initial registration process. Clients only need to submit a reissue request. The registration service accepts the request and returns the response.

- Revocation

A registration service permits clients to revoke a key binding previously issued. An authorized client may request that the trust service revokes a key binding. This is necessary because the key has been compromised or because information contained in the key binding is incorrect (Hallam-Baker and Ford, 2001). Sufficient information must be included in the request to identify the key binding to be revoked such as key binding ID for evidence, and then the registration service responds that the key binding has been revoked.

- Recovery

When the key pair is created by the registration service, private key recovery is essential because clients may lose their private key and require accessing to their encrypted data (Hallam-Baker and Ford, 2001). The registration service provides functionality of recovery a private key to a client under this situation.

2.4 Canonical XML

The specification “Canonical XML 1.1” provides an approach for creating a unique physical representation of an XML data which accounts for permissible changes (Boyer and Marcy, 2007). This specification is used to guarantee that logically-identical XML documents give identical XML signatures. XML Canonicalization (Canonicalization is often simply called “c14n”) discards irrelevant details from an XML data and supplies a non-ambiguous octet representation. If two XML data have the same canonical results, then the two XML data are logically equivalent in a given context.

Canonical XML is used by XML signature to create a unique representation of an XML data or a subset. This unique representation is necessary to compute a cryptographic hash value which to be signed, because the hash function is sensitive to any character changing. XML 1.0 is so flexible in document formats that equivalent contents can be expressed in multiple formats. An example is given below.

(1) <document></document>
(2) <document/>

Both code fragments in (1) and (2) above represent an empty element. They are different in byte representation, but are equivalent as XML data. The XML 1.0 specification allows equivalent XML data to be expressed in multiple formats in terms of attribute occurrence sequence, naming space definitions, and blank character handling, among others.

The digital signature is generated based on hash value of byte representations for XML data. Because of the flexibility of the XML 1.0 specification described above, signing the logically equivalent contents may lead to failed signature verification. Against this background, the canonical XML specification, which provides for canonical forms that are equivalent to XML data formats, was established ahead of XML signature specifications. Based on the canonical XML specification, an XML data is need to be converted to a canonical form before XML data is signed and verified (Weerasinghe et al., 2006).

The changes for canonical XML have been summarized into two different categories, the first is relative to content changes and the other is the structure change for document subset.

2.4.1 Content changes for canonical XML

- Character encoding: c14n always uses the UTF-8 as character encoding scheme.
- Line breaks: all line endings are normalized to #xA.
- Attribute values: attribute values are normalized to the XML 1.0 specification. All attribute values are delimited by double quotes.

- The replacement of references for character and parsed entity.
- CDATA sections are converted into text content.
- XML declaration and DTD removed: both XML declaration and document type declaration are omitted from canonical XML.
- Empty elements: use start-end tag pairs replace empty elements.
- White space: all white space in character content is retained. White space within start and end tags are reduced to a single space.
- Special characters: use character references to replace special characters in character contents.
- Namespace declarations: each element's superfluous namespace declarations bas been removed.
- Default attributes: default attributes for particular elements must be added to respective elements.
- Lexicographic: the namespace declarations and attributes of elements are arranged as lexicographical order.

2.4.2 Structure changes for document subsets

Some applications require a physical representation for an XML document subset. Figure 2.4 illustrates the process of canonical document subset. Figure 2.4 (a) shows the XML tree with selected nodes which will be included in document subsets. Figure 2.4 (b) shows the canonicalized document subsets.

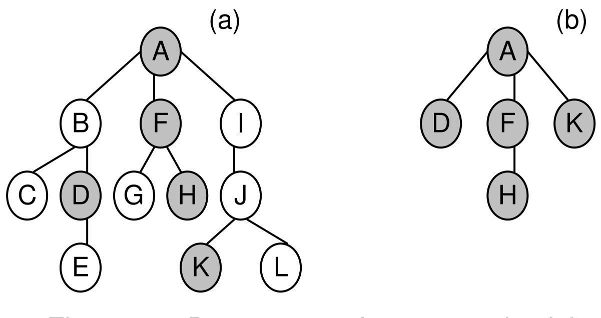


Figure 2.4 Document subset canonicalizing

The selected nodes are A (/A), D (/A/B/D), F (/A/F), H (/A/F/H), K (/A/I/J/K), where the bracketed content is the XPath string of each selected node in Figure 2.4 (a).

The changes are that nodes become direct children of their visible ancestor when their parent node has not been selected. As shown in Figure 2.4, the node K becomes a child of node A, and node D becomes a child of node A.

Exclusive XML Canonicalization is one of the XML canonicalization specifications. It has been established considering special situations. In consideration that signed XML data *A* will be inserted as a child element of XML data *B*. Because of canonicalization, the name space of XML data *A* will be changed when XML data *B* is converted according to the canonical XML specification. This will result an invalid XML signature verification for XML data *A*. The exclusive XML canonicalization specification, which is based on canonical XML specification, was established to avoid this problem (Weerasinghe et al., 2006). This specification is particular important for Web Services Security, which specifies XML-signed SOAP messages.

2.5 XML signature

XML signature is a digital signature technology that is optimized for XML data. The practical benefits of this technology include partial signature, which allows an electronic signature to be written on specific tags contained in XML data, and multi-signature, which enables user to generate more than one signature within the same XML data. The use of XML signature can solve security problems, including falsification, spoofing, and repudiation.

XML signature was established as a formal version of W3C recommendations in Feb. 2002 (Bartel et al., 2008). W3C has also established related specifications that need to be fulfilled when XML signature is actually deployed. The specifications relative to XML signature are listed:

- Canonical XML Version 1.0: W3C Recommendation 03/15/2001 (Boyer, 2001).

- Exclusive XML Canonicalization Version 1.0: W3C Recommendation 07/18/2002 (Boyer et al., 2002a).

- XML-Signature XPath Filter 2.0: W3C Recommendation 11/08/2002 (Boyer et al., 2002b).

Based on specifications above, the W3C published the first edition and the second edition of XML digital signature specification in 2002 and 2008 respectively.

- XML-Signature Syntax and Processing: W3C Recommendation 02/12/2002 (Bartel et al., 2002).

- XML Signature Syntax and Processing (Second Edition): W3C Recommendation 10/06/2008 (Bartel et al., 2008).

2.5.1 Structure of XML signature

XML-Signature Syntax and Processing specification provides the rules for XML signature. It defines signature in XML format, the approach for signature generation, and method for signature verification. Figure 2.5 shows the structure of the element <SignedInfo>.

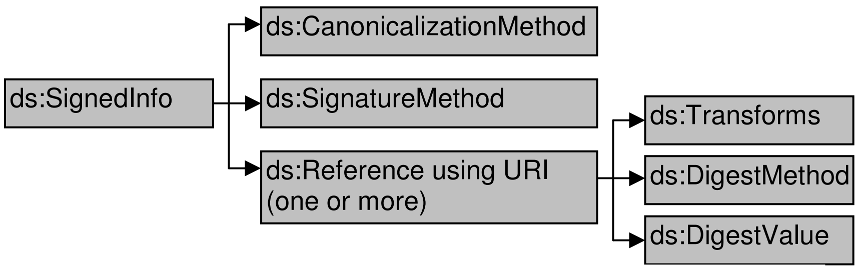


Figure 2.5 Structure of SignedInfo in XML signature

A structure of XML signature is that the <Signature> element lies at the top of the document. The element <Signature> contains the element of <SignedInfo>, which includes references to the algorithms applied to XML signature generation and the target in XML data (Weerasinghe et al., 2006). It also holds hash value and other information. An element of <SignatureValue> includes the signature result, and public key information is contained in <KeyInfo> element, which to be used when the XML signature is verified. When considering the characteristics of XML signature, the <Reference> element is particularly important. Multiple <Reference> elements may be contained in the <SignedInfo> element. This is used to identify XML data segments at any location in XML data to be signed. With this advantage, multi-signature is also supported through simply repeating XML signature. However, this kind of multi-signature will increase the size of signature results. The signing process only can be executed with an independent way, and the signing process for users' dependent relationship cannot be supported.

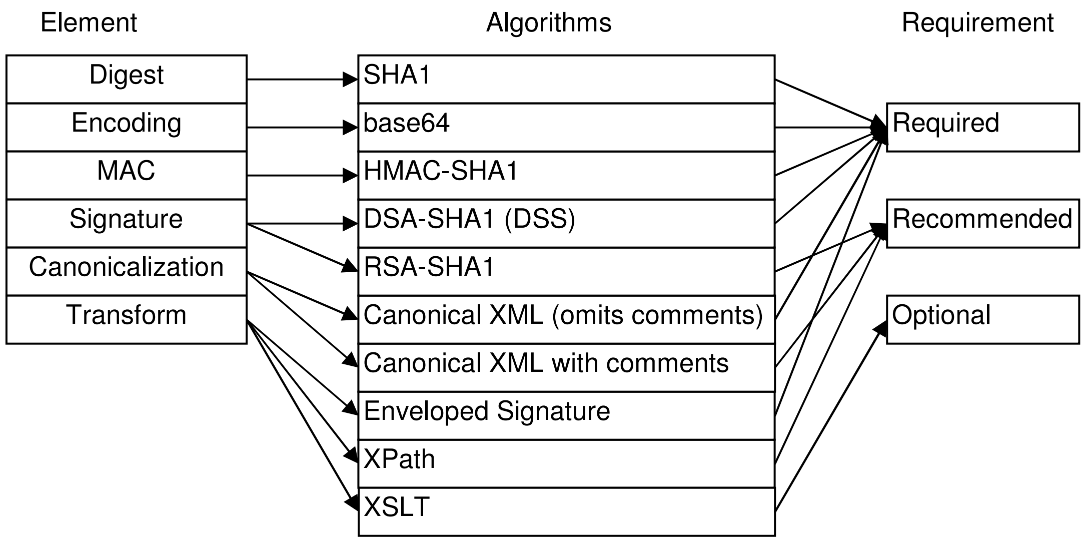


Figure 2.6 Algorithms for XML digital signatures correspondence with different elements according to W3C specification

Figure 2.6 shows algorithms deployed in XML digital signature. The signature algorithms deployed in XML digital signature are RSA and DSA. XML signature permits one to deploy one-way hash functions to get a hash value using SHA-1, and recommends using HMAC-SHA1 to get a MAC. Although the integrity method has been introduced, XML signature scheme does not provide how to organize this information of portions of XML tree. Each signature must have exactly one <SignedInfo> element to indicate what is signed by the signature. The signature is an intermediary list of hash values. Generation of the <SignedInfo> does not require the usage of a private key, as only hash values are generated. The <SignedInfo> is the final object which is being signed by the cryptographic signature.

2.5.2 Enveloping, enveloped and detached signatures

XML signature supports three kinds of signature representation forms: enveloping, enveloped, and detached. These terms for XML signature refers to the relationship between signed contents and signature. The properties and limitations of the three kinds of forms are as follows.

- Enveloping signature

An enveloping signature is an ancestor of the signed contents in the XML tree as shown in Figure 2.7.

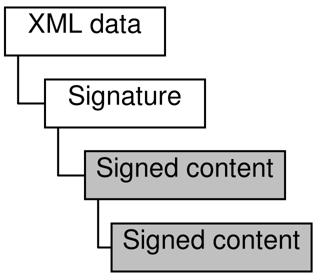


Figure 2.7 Enveloping signature

The major feature of an enveloping signature is that only one data object is in signed contents. The signed contents and signature form a single object. The application must strip away the signature-envelope before handling signed contents within enveloping signature. The advantage of this kind of signature approach is that the signature and signed content form a single entity which can be handled easily during transport. There is no problem to miss the signature or contents since it is always together.

- Detached signature

A detached signature means that the signature is separated from the signed contents. The signed contents are outside of the signature element. A detached signature has no parent/child relationship to the signed contents. There are two situations as shown in Figure 2.8: the signature and the signed contents reside in separate files, or the signature and the signed contents reside in the same XML document but have no parent/child relationship, usually both are siblings.

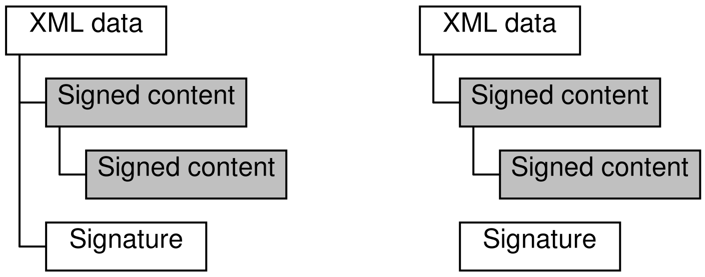


Figure 2.8 Detached signature

The major feature of detached signature is that signature is not merged into signed contents. In XML signature specification, the signed content is identified using URIs. This provides a binding between signature and signed contents, and it makes the selection of signed object more flexible, e.g. the object on a web server or in any directory can be accessed by a URIs. Different from enveloping signature, if the signer sends only the signature, the verifier still can access the signed contents via URIs mechanism. Without protecting elements' context-relationship, this flexibility can lead to a result that a signature can be copied to any XML data still keeping valid signature verification.

- Enveloped signature

An enveloped signature is a descendant relative to the whole or parts of signed contents in the XML tree as shown in Figure 2.9. Enveloped signature introduced by XML signature is that the signature is placed inside the signed contents. Because signature becomes a part of XML data, enveloped signatures can only sign XML data. This kind of approach changed the structure of the original XML data. This is the reason that XML signature provides a transform mechanism to select portions of signed XML data.

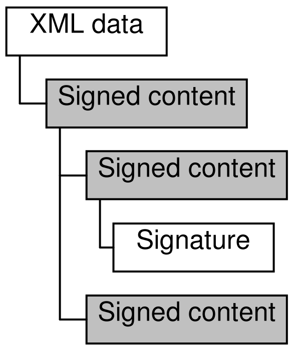


Figure 2.9 Enveloped signature

2.6 XML encryption

XML encryption specification was established by the W3C as a formal version of W3C recommendations in December 2002 (Imamura et al., 2002). The W3C also established related specifications that solve problems raised when XML encryption and XML signature are used in combination. The specifications relative to XML encryption are listed:

- XML Encryption Syntax and Processing: W3C Recommendation 12/10/2002

- Decryption Transform for XML Signature: W3C Recommendation 12/10/2002

XML encryption is an encryption technology that is optimized for XML data. This specification provides format for using XML and processing rules regarding to encryption and decryption. Its practical benefits include partial encryption, which encrypts specific tags contained in XML data, multiple encryption, which means that data can be encrypted multiple times, and even more complex encryption, such as the designation of recipients who were permitted to decrypt respective portions of data. The use of XML encryption also facilitates to solve security problems, including XML data eavesdropping.

XML encryption offers various benefits. An XML element containing XML encryption information can act as a container for encrypted data or as a container for encrypting key or both. XML encryption is capable to encrypt the whole XML data or portions of it within an XML document (Geuer-Pollmann, 2002). XML encryption allows direct inclusion of the encrypted contents into the container or to reference the encrypted contents via the transform mechanism. XML encryption offers key management facilities for symmetric wrapping of private keys, private key transportation, and key agreement using Diffie-Hellman.

```
<EncryptedData Id? Type? Encoding?>
  <EncryptionMethod/>
   <ds:KeyInfo>
     <EncryptedKey/>
     <AgreementMethod/>
     <ds:KeyName/>
     <ds:RetrievalMethod/>
   </ds:KeyInfo>
   <CipherData>
     <CipherValue/>
     <CipherReference URI/>
   </CipherData>
   <EncryptionProperties/>
</EncryptedData>
```

Figure 2.10 Structure for XML encryption
(Imamura et al., 2002)

The structure for XML-encrypted data is shown in Figure 2.10 as described by W3C. The <EncryptedData> element lies at the top of encrypted results. <EncryptionMethod> element is the child element of <EncryptedData>. The element of <EncryptionMethod> contains algorithms information for encryption result generation. The decryption key information is contained in element <KeyInfo>, which is used to decrypt encrypted-data. The <CipherData> element is the final encrypted value. If hybrid encryption is used, the structure can also include the <EncryptedKey> element, which contains the key-encryption key. URIs can be used to specify what has been encrypted. This indicates that XML encryption provides a flexible method to identify the encrypted objects. The detailed element in XML encryption and related algorithms is shown in Figure 2.11.

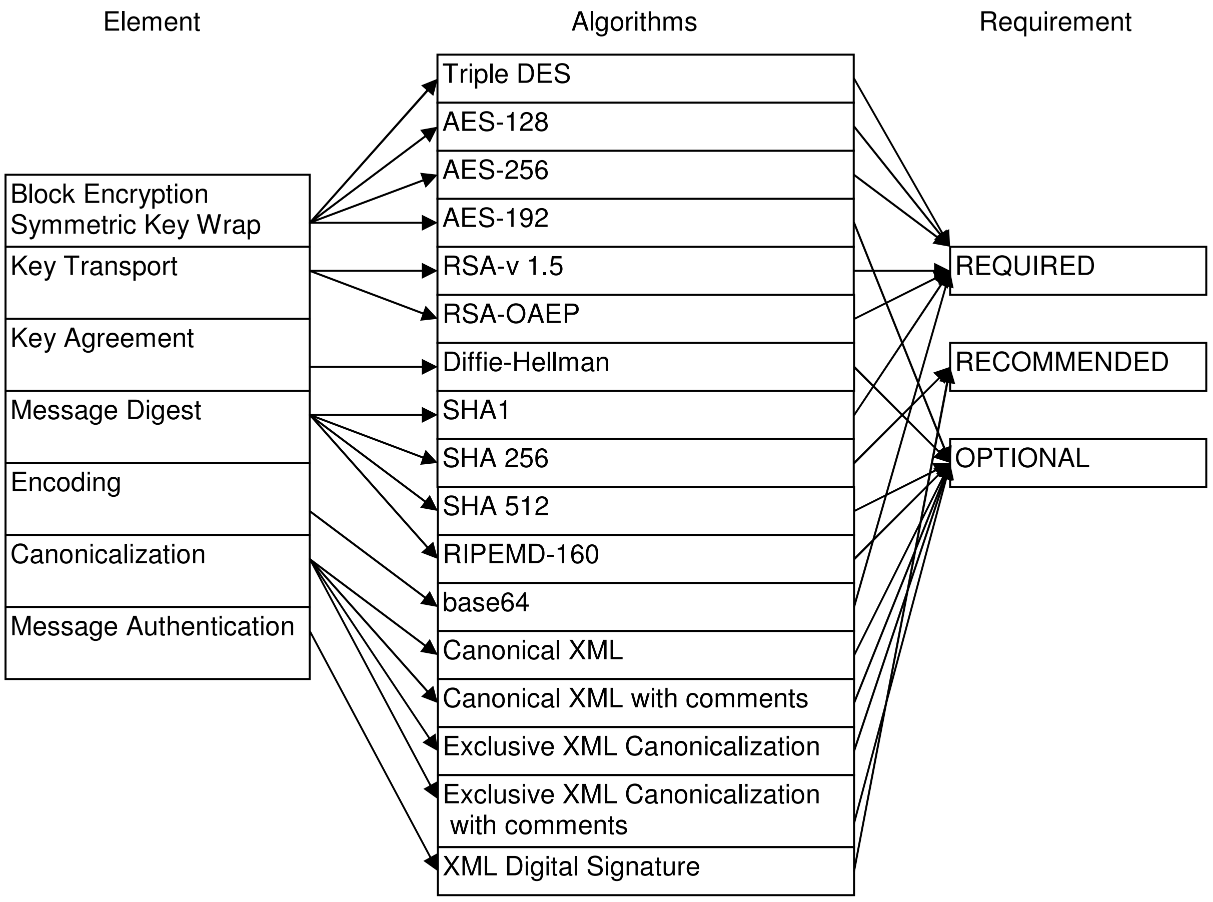


Figure 2.11 Algorithms for XML encryption correspondence with different elements according to W3C specification

The value of element <EncryptionMethod> is the identifiers of block encryption algorithms. The major block encryption algorithms deployed are 3DES, AES-128, AES-256, AES-192. Key Transport algorithms are used to specify the encrypting and decrypting keys. Key transport algorithm includes RSA-v1.5 and RSA-OAEP (Imamura et al., 2002).

In order to judge the sequence of XML signature and XML encryption for the same portions of XML data, the specification of “Decryption Transform for XML Signature” was established (Hughes et al., 2002). Generally, signing an encrypted XML data is meaningless in practice. When user encrypts the whole or portions of signed XML data, it needs to identify the encrypted object for decrypting. This specification provides an approach to solve the problem. This has been established by the W3C’s XML encryption working group as an additional specification with regard to the conversion processing that is performed on XML signatures.

2.7 Summary

Traditional security technologies are the basis of XML security. This chapter mainly introduces the traditional security techniques that are utilized in XML security specifications. The XML security relative specifications published by W3C and OASIS are the core of XML security technology, such as XML key management satisfies key requirements for signature or encryption, XML signature provides XML data authentication, and XML data confidentiality is ensured using XML encryption technology.

XML key management specification provides public key management to support XML security applications. The validation for digital certificate is a bottleneck, and it increases the burden of the server. Without considering elements’ context relationship which can be ensured by XML data integrity, an XML signature can be copied to another XML data still keeping successful signature verification. In addition, simply repeating XML signature to generate multi-signature will increase the size of signature results, and this kind of multi-signature cannot support a dependent signing process. Although XML encryption specification offers some benefits, how to locate the information contained in cipher text has not been addressed. After several rounds encryption, only the plaintext can be queried while the information residing in cipher block cannot be identified. The issues mentioned above will be investigated in detail in next chapter.

Chapter 3 Literature Review

This chapter analyzes the revocation information validation approach for X.509 digital certificate. XML data integrity and relative solutions are investigated. The XML multi-signature schemes are analyzed. The approaches for encrypted XML data query processing are also investigated.

3.1 Revocation information validation for X.509 certificate

Certificate revocation is the action of declaring a certificate invalid before its validity expired. There are two major approaches to check validity of a certificate status, Certificate Revocation List (CRL), and Online Certificate Status Protocol (OCSP).

CRL is a list issued and digital signed by a certificate authority (CA), and it contains the serial number of certificates that they should not be used if they have been revoked before their expiration date. This list is dated and also has an expiration date. User must download a new CRL after it's expired. However, CRLs are too bandwidth and cannot support a good degree of timeliness (Myers et. al, 1999; Micali, 1997, Goyal, 2004a; Arnes, 2000; Benjumea et al., 2007; Goyal, 2007; Wazan et al., 2008). Several CRL relative approaches have been proposed to improve the efficiency of digital certificate validation. Certificate Revocation System (CRS) enables system to answer the user query with a high efficiency (Micali, 1997; Micali, 2002; Goyal, 2007). The basic idea of CRS is as follows. For certificate creation, the CA selects two random numbers Y_0 and X_0, and computes $Y = H(Y_0)$, where H is a hash algorithm such as SHA1. Let $X_1 = H(X_0), X_2 = H(X_1),..., X_{365} = H(X_{364})$, where H is a hash algorithm, the number 365 denotes the number of days in the year. Y and X_{365} are included in the certificate and signed along with the other usual information. $Y_0, X_0,..., X_{364}$ keep secret by CA. When the CA receives a validation request on the i th day, CA makes two choices with checking CRL. If the certificate is revoked, the CA releases Y_0, which can be verified by hashing and comparing with Y specified in the certificate. If the certificate is still valid, the CA releases X_{365-i} which can be verified by hashing i times and comparing with X_{365} specified in the certificate. However, CRS is difficult to be deployed in distributed querying systems (Goyal, 2007). The communication between CA and directory is too frequent, which shoots up the overall bandwidth cost of the system (Naor and Nissim, 1998; Aiello et al., 1998; Goyal, 2007).

Certificate Revocation Tree (CRT) is another well-known approach for certificate revocation solution (Kocher, 1998; Goyal, 2007). A CRT is based on the Merkle hash function (Kocher, 1998). The tree leaves contain the serial number of the revoked certificate which is included in a relevant CRL. The root of the tree is signed by the CA. The certificate status proof for a certificate with serial number consists of the path node siblings from the root to the appropriate leaf, in addition to the signature on the root of the tree. Although the communication is low, the data volume to be downloaded is still large. The overall cost is still relatively high.

The OCSP is another certificate revocation solution designed by IETF (Myers et al., 1999). The protocol requires the security client to send a request to an OCSP responder which is the server returning status information about a specific certificate when asked. OCSP is an online service, and it has a high degree of timeliness. Because the CA is required to create a signature for each query, OCSP increases the communication burden between server and client (Goyal, 2007).

3.2 XML data integrity

General applications of data integrity could exist in many domains, including e-government, e-commerce, e-financial services, e-business, e-banking, e-learning, e-healthcare, mobile communications, heterogeneous networks, digital factories, multi-agent systems, and grid computing (Wu et al., 2002; Chen and Lu, 2004; Rushinek, 2002; Boritz and No, 2005; Jones et al., 2000; O'Neill, 2007; Yee et al., 2006; Blobel, 2004; Dankers et al., 2002; Ekelhart et al., 2008; Karnouskos, 2005; Woerner and Woern, 2005; Oliveria et al., 2006; Cody et al., 2008). Wu and Chen described the need for data integrity when official documents are being transmitted between government agencies for e-government in Taiwan (Wu et al., 2002; Chen and Lu, 2004). O'Neill pointed out the importance of data integrity through an assessment of a bank's web service (O'Neill, 2007). IBM gives an example of data integrity as follows: assume the data is a funds transfer and the hacker alters a random piece of the data that happens to be the account number. When the bank decrypts the data, the account number is not a valid account; therefore, the data tampering is detected and the transaction is not completed. However, assume instead that the data altered by the hacker is the amount of money and, changed it from 1000 units to 9000 units (IBM, 2008). In this case, the transaction would be completed using the incorrect amount. Research into this area would be of great benefit.

There are two approaches to ensure integrity for XML data. The first tries to add additional elements in XML data to record the integrity information. Hussain maintained the integrity of XML signatures using the manifest element (Ekelhart et al., 2008; Hussain and Soh, 2004). McIntosh presented an element position attack, and solved this problem by adding additional objects in XML data (McIntosh and Austel, 2005). Another approach is based on hash function mechanism.

McIntosh summarized the context dependent semantics for XML data integrity with examples. The context dependent semantics for XML data integrity has been summarized into three situations:

- Simple ancestry context

  It means that an element has a specific position in an XML document. From the element's name, value, attributes, and its ancestors or children's name, the semantic meaning of this element can be completely derived (McIntosh and Austel, 2005).

- Sibling value context

  This situation means that the element has sibling elements with the same name but with different semantic meanings (McIntosh and Austel, 2005).

- Sibling order context

  The element's semantics are relative to their order in sibling elements. If the order of sibling has been changed, it also affects the semantics of the element.

In order to prevent authorizing the access requests with a mistake, McIntosh suggested that properly specified and enforced security policy should be deployed. For an optional element context, an absolute XPath expression references should be considered to adding specification of security policy. Although McIntosh presented sibling value context and sibling order value context, he has not proposed proper approach to handle it. Without cryptography, this kind of method is easily attacked by a hacker.

The second approach is based on a cryptography mechanism, and adopts a hash function to ensure integrity. DOM-HASH is the first algorithm proposed by Maruyama to calculate a hash value for XML data (Maruyama et al., 1999). In this algorithm, MD5 and SHA1 were adopted to generate hash values with four different node types related to XML data. The four node types include element, attribute, processing instruction (PIs), and text. The detailed algorithm is as follows.

$dos(v) = h(v.elem \parallel v.text \parallel v.pi \parallel v.attr)$

where, v is the element set of XML data, h is a collision-resistant one-way hash function.

This approach only satisfies the contents integrity of the XML data. It does not provide integrity for subset of DTD (Brown, 1999).

Similar to DOM-HASH, the XHASH algorithm has been proposed by Brown. The XHASH makes use of two parameters: the first is the hash function such as SHA1; the second (optional) can be used to determine how non-significant space characters will be handled by default (Brown, 1999). The values for this attribute are set as ‘default’ and ‘preserved’, and it is difficult to specify the non-significant space characters which should be discarded (Brown, 1999).

Devanbu adopted the DOM-HASH and the Merkle hash function to maintain the integrity of XML data queries (Devanbu et al., 2001). The aim of Devanbu’s scheme is to assure that the client can obtain complete and correct answers corresponding to their queries. The hash value of the XML document is generated by using Merkle hash function. When client obtains a queried result, the correctness can be verified by checking the related hash value.

Bertino also adopted the Merkle hash function to handle integrity of XML documents publishing (Bertino et al., 2004). These two approaches provide a solution to generate hash values of XML data based on the Merkle hash function.

The XML data and the Merkle hash function defined by Bertino:
Let $d = (V, r, E, \varnothing E)$ be an XML data, where V is a set of nodes in XML data d, r is the root node of XML data d, E is the set of edges, and ϕE is the edge labelling function. h is a collision-resistant hash function (Bertino et al., 2004). Let HS be the co-domain of h. The Merkle hash function associated with d denoting as MhX is a function: $V \rightarrow HS$ such that, for each $v \in V$:

$$MhX(v) = \begin{cases} h(h(v.val) \parallel h(v.name)) & if \quad v \in V^a \\ h(h(v.content) \parallel h(v.tagname) \parallel MhX(child(1, v)) \parallel \ldots \parallel MhX(child(N_{c_v}, v))) & if \quad v \in V^e \end{cases}$$

where, V^a is the leaf node in XML data d, V^e is the non-leaf node in XML data d ‘||’ denotes the concatenation operator, $child(N_c, v)$ is used to obtain the children of an element, and $N_c \in (1 \ldots n)$ is the child of node v (Bertino et al., 2004).

The integrity approach proposed by Bertino can ensure both the schema and the contents of an XML data. On the one hand, a subject can verify that contents of an XML data have not been altered, e.g. that no modification occurs at the value of an element's content, or the value of a relative attributes. On the other hand, a subject is able to verify that the XML data schema has not been modified. Attacker altering the name of an attribute or an element tag can be revealed. Based on cryptography, this kind of approach has a higher security level than the first approach. However, the element's attribute integrity has been ignored in this approach (Carminati et al., 2005). Because of using Merkle hash function, the virtual nodes will be increased when generating a hash value from bottom-up for XML data, and this will lead to a low efficiency.

W3C published XML signature specifications in 2000 (Second Edition in 2008) (Bartel et al., 2008; Reagle, 1999). This specification provides the format for data integrity expressions in XML signatures, and gives the optional algorithm to generate hash values, such as SHA-1, SHA-256. However, signed resources can be copied to another document but still keeping signature valid, and this can be utilized by an attacker to generate an authorized XML data.

```
<Books>
 <Title>XML Security</Title>
 …
 <Amount>20</Amount>
 …
 <Payment>£160</Payment>
 …
 <Signature>
 …
 </Signature>
</Books>
```

(a)

```
<Books>
 <Title>XML Technology</Title>
 …
 <Amount>5</Amount>
 …
 <Payment>£70</Payment>
 …
</Books>
```

(b)

```
<Books>
 <Title>XML Technology</Title>
 …
 <Amount>5</Amount>
 …
 <Payment>£160</Payment>
 …
 <Signature>
 …
 </Signature>
</Books>
```

(c)

Figure 3.1 An example of forging a signature

Figure 3.1(a) and (b) are two different invoices for the book order. The authorized entity signed the payment £160 in Figure 3.1(a). Figure 3.1 (b) also contains an element <Payment> with value £70. An attacker may change the payment from

£70 to £160 in Figure 3.1(b), and copy the signature from Figure 3.1 (a) to Figure 3.1 (b). The forged document is shown in Figure 3.1 (c). In the forged document, the signature still keeps valid.

3.3 XML multi-signature schemes

3.3.1 Multi-signature schemes for non-XML data

Multi-signature schemes for non-XML data include extended DSA, RSA, or ElGamal schemes, signing sequence, broadcast signing architecture, distinguished signing authorities, and order specify. Table 3.1 lists the advantages and disadvantages of these schemes.

Table 3.1 Multi-signature schemes for non-XML data

Approach	Advantages	Disadvantages
Extended DSA, RSA, or ElGamal	Easy to be implemented	The size of multi-signature results grows with the increasing numbers of signers.
Improved extended DSA, RSA, or ElGamal (Itakura and Kiesler, 1990; Harn and Kiesler, 1989; Kiesler and Harn, 1990; Ohta and Okamoto, 1991; Boyd, 1991)	The size of multi-signature results has nothing to do with the numbers of signers.	Predefined signing sequence. Verifying the signature with the knowledge of signing sequence.
Undistinguished signing authorities (Harn, 1994a; Harn, 1994b; Hardjono and Zheng 1992; Michels and Horster, 1996)	Signing and verifying process is independent to the sequence of signing process.	All signers sign the same message.
Distinguished signing authorities (Harn, 1999; Wu et al., 2001; Mitomi and Miyaji, 2000; Wu and Hsu, 2002; Huang and Chang, 2005; Yamamoto and Ogata, 2007)	Sign the message which who is responsible for.	The signing order is not a mixed sequential and broadcasting way.
Signing order specified (Doi et al., 2000; Tada, 2002; Burmester et al., 2004; Wang et al., 2006; Yang et al., 2006)	The signing is a mixed sequential and broadcasting way.	Inflexibility in adding or deleting signers.

One of approach to construct a multi-signature for a message is to repeat the scheme of DSA, RSA, or ElGamal. The major drawback of this approach is that the size of a multi-signature result grows with the increasing of the number of signers (Wu et al., 2001).

In order to overcome the drawbacks mentioned above, Italura and Nakamura presented a multi-signature scheme based on the RSA scheme (Itakura and Kiesler, 1990). In this scheme, the size of a multi-signature result has nothing to do with the numbers of signers. However, the signers have to follow the predefined signing sequence to sign the document, and verify the signature with the knowledge of signing sequence. Similar schemes also have been proposed by (Harn and Kiesler, 1989; Kiesler and Harn, 1990; Ohta and Okamoto, 1991; Boyd, 1991), which are based on extended RSA, DSA, or ElGamal schemes with sequential multi-signature.

Harn proposed a multi-signature scheme based on a modified ElGamal digital signature. In this scheme, the signature-generation and verification process is independent of the sequence of signing process (Harn, 1994a; Harn, 1994b). This scheme is known as multi-signature scheme which is based on broadcast architecture. The similar schemes can be found in (Hardjono and Zheng 1992; Michels and Horster, 1996). In these schemes, all signers sign the same message, and it was defined as “undistinguished signing authorities” by Harn (Harn, 1994b). It was defined as “distinguished signing authorities” if signers can sign different portions of a document. “Undistinguished signing authorities” indicates that all signers have the same responsibility for the signed document. “Distinguished signing authorities” indicates that signers have different responsibility for different portions of the signed document. However, multi-signatures with distinguished signing authorities are needed in applications, e.g. a company publishes a document that may involve the financial department and engineering department to sign different sections of the document (Huang and Chang, 2005).

A multi-signature scheme which has distinguished signing authorities proposed by Harn in 1999 (Harn, 1999). In this scheme, signers can only sign the message which he is responsible for. However, Li discovered an efficient insider attack on

Harn multi-signature scheme in 2000 (Li et al., 2000). Wu proposed a “delegated multi-signature scheme with document decomposition” in 2001 (Wu et al., 2001). Wu’s scheme is more efficient in multi-signature-generation and verification. However, Lu and Chen pointed out that Wu’s balanced strategy to delegate subdocuments to qualified signers is problematic, because each signer should sign the portions of the document that they are responsible for rather than the portions of the documents based on some balanced strategy (Lu and Chen, 2004). Mitomi proposed a general model for multi-signature with message flexibility in 2000 (Mitomi and Miyaji, 2000). Yamamoto improved Mitomi’s scheme in 2007 (Yamamoto and Ogata, 2007). Wu proposed an ID-based multi-signature scheme with “distinguished signing authorities for sequential and broadcasting architectures” in 2002 (Wu and Hsu, 2002). Huang presented “multi-signatures with distinguished signing authorities for sequential and broadcasting architectures” in 2005 (Huang and Chang, 2005). Although these models considered message flexibility, they have not considered the signing order in a mixed sequential and broadcasting way.

To date, signing order specified multi-signature schemes are Doi’s model in 2000, Tada’s model in 2002, Burmester’s model in 2004, Wang’s model in 2005, and Yang’s model in 2006 (Doi et al., 2000; Tada, 2002; Burmester et al., 2004; Wang et al., 2006; Yang et al., 2006). There are two different major approaches to deal with this directed series-parallel signing graph. Tada and Yang adopt a series-parallel group, which are based on directed graphs (Tada, 2002; Yang et al., 2006). Another approach presented by Burmester, who also represented the group of signers by a graph, and then decomposed the graph to a tree (Burmester et al., 2004). There are two obvious disadvantages in these schemes. First, the scheme makes the signer order as a signature parameter, increasing the complexity of multi-signature algorithm. Second, each signer needs to verify the signing order before signing, and update the signing graph or decomposition tree after signing. These disadvantages will lead to inflexibility in adding or deleting signer group members.

3.3.2 XML multi-signature schemes

As for the XML multi-signature, two schemes have been presented. The first is based on a repeated DSA or RSA scheme. The second approach is proposed by Lu based on delegated multi-signature scheme proposed by Wu.

- Repeat of DSA or RSA

This approach is deployed by W3C in XML signature specification (Bartel, 2008). The process of repeated DSA or RSA can be described through following three steps:

Step 1: Assigning XML data which need to be signed to each signer using XPath expression, $X_d = \{M_1, M_2, \ldots, M_n\}$, where, $M_{i(i=1,\ldots,n)}$ is the data to be signed.

Step 2: Each signer generate signature separately as: $S_1 = A_e(M_1, K^1_{priv})$, $S_2 = A_e(M_2, K^2_{priv})$,..., $S_n = A_e(M_n, K^n_{priv})$, where, S_i is the signed result, $A_e()$ is the encryption function based on RSA or DSA to generate signature results, and K^i_{priv} is the private key of the signers.

Step 3: Assembling the signed results to a single XML data, and letting $S = S_1 \cup S_2 \cup \ldots \cup S_n$, where, S is the final signature. This means that the final multi-signature result is a set of individual single signatures.

The major advantage of this approach is easy to be implemented. However, this approach increases the XML data size when signers group is big, and cannot support multi-signature generation under dependant situation.

- Delegated multi-signature scheme

Wu has presented the “delegated multi-signature scheme with document decomposition” (Wu et al., 2001). In this scheme, a document is decomposed into a set of subdocuments and then assigned to signers using a dispatch algorithm. The scheme consists of four components (Wu et al., 2001): a group of signers, a system authority (SA), which provides system initialization such as system parameters, private key and public key generation. A document dispatcher (DD) is used to decompose document and delegate subdocument. A signature collector (SC) collects and verifies the individual signatures created by each signer.

The scheme consists of the private key and public key generation, the multi-signature generation, and the multi-signature verification (Wu et al., 2001).

Stage 1: Private key and public key generation

SA chooses a large prime $p \geq 2^{512}$, a large divisor $q \geq 2^{140}$ of $p-1$, a generator α of order q in $GF(p)$, and a hash function $h(x) \in GF(p)$ for any x. After publishing p, q, α, h, SA can accept the registration requested by any signing group. Let $G = \{u_1, u_2, \ldots, u_n\}$ be the registered signer group, and u_j is the individual signer. After finished registration, SA generates a distinct private key and public key pair (x_j, y_j) for each $u_j \in G$, where $x_j \in Z_q$ and

$$y_j = \alpha^{-x_j} \bmod p$$

The private key and public key pair (X, Y) for G are generated by using:

$$X = \sum_{u_j \in G} x_j \bmod q \text{ , } Y = \prod_{u_j \in G} y_j \bmod p$$

Stage 2: The multi-signature generation stage

DD decomposes M, which need to be signed, into set of subdocuments, and it is denoted as: $\Gamma = \{w_1, w_2, \ldots, w_m\}$. Let M_j be the subset of Γ delegated to u_j. DD assigns M_j to u_j. The multi-signature generation consists of seven steps.

Step 1: DD sends $\{h(M), M_j\}$ and $\{h(M)\}$ to u_j and SC, respectively.

Step 2: All $u_j \in G$ extract w_j from their delegated M_j and cooperatively check the integrity of M by verifying that $h(M) = h(w_1 \| w_2 \| \cdots w_m)$, where "||" is the concatenation operator.

Step 3: Each $u_j \in G$ randomly selects an integer $z_j = Z_q$ and computes

$$r_j = \alpha^{z_j} \bmod p,$$

$$R_j = r_j^{h(M_j \| r_j)} \bmod p$$

and sends $\{R_j\}$ to other participant signers and SC.

Step 4: Each $u_j \in G$ computes

$$R = \prod_{u_k \in G} R_k \bmod p,$$

$$s_j = (z_j h(M_j \| r_j) R + x_j h(h(M \| R)) \bmod q$$

and sends $\{M_j, r_j, s_j\}$ to SC. Here, (r_j, s_j) is the personal signature of M for u_j

Step 5: SC obtains w_j from received M_j and check the integrity of M by verifying that $h(M) = h(w_1 \| w_2 \| \cdots w_m)$.

Step 6: SC computes R and verifies (r_j, s_j) by checking whether following equation holds.

$$r_j^{h(M_j \| r_j)R} = (\alpha^{s_j})(y_j^{h(h(M) \| R)})(\bmod p)$$

Step 7: If the personal signatures generated above are successfully verified, then SC computes $S = \sum_{u_j \in G} s_j \bmod q$ and publishes (R, S) as the multi-signature of M for G.

Stage 3: The multi-signature verification stage

Any verifier can check the signature by using the following equation.

$R^R = (\alpha^S)(Y^{h(h(M) \| R)})(\bmod p)$, if this equation holds, then (R, S) is successfully verified.

This scheme is more efficient in multi-signature-generation and verification. However, its balanced strategy to delegate subdocuments to qualified signers is problematic. In addition, this scheme only supports parallel signature-generation scenarios, and cannot handle a multi-signature generation under a dependant situation.

- Lu's XML multi-signature scheme

Based on Wu's delegated multi-signature scheme, Lu presented XML multi-signature in 2004 (Lu and Chen, 2004). In this scheme, he first proposed signing XPath expression instead of XML data itself.

In Lu's scheme, there are four components: a group of signer G, a system authority (SA), document decomposition (DD), and a signature collector (SC) (Lu and Chen, 2004). DD decompose a document M into a set of subdocuments $\Gamma = \{w_1, w_2, \ldots, w_m\}$ using a set of rules $T = \{t_1, t_2, \ldots, t_m\}$, where t_i is the XPath expression. Via XPath expression, one can easily obtain a subdocument M_i. The procedure for generating a multi-signature of M for G is as follows.

Step 1: DD sends $\{h(M), M_j, T_j\}$ and $\{h(T), h(M)\}$ to u_j and SC, respectively.

Step 2: All $u_j \in G$ extracts w_i from M_j delegated to them and then cooperatively checks the integrity of M by verifying $h(M) = h(w_1 \,||\, w_2 \,||\ldots||\, w_m)$ where "||" is the concatenation symbol.

Step3: Every $u_j \in G$ extracts t_i from T_j, computes hash value $w_i = C_{t_i}(M)$, and verifies whether or not every newly computed w_i is identical to the received w_i. If all w_i are successfully verified, each u_j randomly selects an integer $z_j \in Z_q$, computes both

$r_j = \alpha^{z_j} \bmod p$, and

$R_j = r_j^{h(T_j \| r_j)} \bmod p$, and sends R_j to other participant signers and SC.

Step 4: Each $u_j \in G$ computes both

$$R = \prod_{u_k \in G} R_k \bmod p \text{ ,and} \qquad (3.1)$$

$$s_j = (z_j h(T_j \parallel r_j)R + x_j h(h(M \parallel R))) \bmod q$$

and sends $\{T_j, r_j, s_j\}$ to SC. (r_j, s_j) is the personal signature of M for u_j.

Step 5: SC checks the integrity of T by extracting t_i from the received T_j and verifying whether or not $h(T) = h(t_1 \parallel t_2 \parallel \dots t_m\}$ holds.

Step 6: To verify (r_j, s_j) for every u_j, SC computes R by Eq.3.1 and checks whether or not the following equation holds.

$$r_j^{h(T_j \parallel r_j)R} = (\alpha^{s_j})(y_j^{h(h(M)\parallel R)})(\bmod p) .$$

Step 7: If all personal signatures generated in the previous steps are successfully verified, then SC computes

$$S = \sum_{u_j = G} s_j \bmod q$$

and publishes (R, S) as the multi-signature of M for G.

In Lu's scheme, the XPath expression is used to transform an XML document into subdocument (Lu and Chen, 2004). Let M be the XML data to be cooperatively signed by the signers. XML data M can be divided into set of subdocuments $\{w_1, w_2, \dots, w_m\}$ using XPath expression, and then signers only need to sign the XPath expression instead of XML data itself. This scheme decreases the communication overhead, although it has three major disadvantages.

First, by division, $M = \{w_1, w_2, \dots, w_m\}$, the integrity checking for each subdocument depends on the formula $h(M) = h(w_1 \parallel w_2 \parallel \dots \parallel w_m)$. This indicates that the document must be delegated entirely; otherwise the integrity checking will be invalid. Supposing a document consists of five parts, and the signers only need to sign three of them. The other two parts have not been delegated, and then the integrity checking will be failed. Second, the subdocument integrity check needs the signers to check cooperatively online. When the group of signers is small, this is possible, but it is impractical when the group of signers is very large. Third, the scheme only provides broadcast (parallel) signature-generation scenarios. It cannot satisfy the signing process under a dependent multi-signature situation. For example, the company policy is set up in a way that the sequence of approval is important and has to be respected: before launching a project, the financial department has to approve the project. Lu’s scheme cannot deal with this application scenario.

3.4 Encrypted XML data querying

With the widely applications of XML, it is necessary to handle sensitive information in XML data, and XML data confidentiality becomes an important issue (Yang et al., 2006). The sensitive parts of the XML data have to be protected in case unauthorized access. There are two approaches to protect the sensitive information in XML data, one is using access control mechanism, and the other is using encryption technology, especially XML encryption technology. Most cases, the access control mechanism can be bypassed and encryption technology is a must (Yang et al., 2006). When XML data is transmitting through an untrusted channel, it needs encryption technology to protect sensitive information (Fan et al., 2004; Agrawal et al., 2004). However, how to query encrypted XML data has not been addressed in XML encryption specification.

Querying encrypted XML data schemes or survey can be found in (Brinkman et al., 2004; Feng and Jonker, 2003; Wang and Lakshmanan, 2006; Lee and Whang, 2006; Gao et al., 2008; Ünay and Gündem, 2008; Jammalamadaka and Mehrotra, 2006; Yang et al., 2006). The basic idea for encrypted XML data query is to build index information for encrypted XML data. Two types of index information are deployed for encrypted XML data. The first one is the structural index information and the other is the value index information (Ünay and Gündem, 2008). Structural index is used to determine the XPath matches any paths in a submitted query. The value index is used to support the range query. These indexes are deployed in either at the server side or client side (Ünay and

Gündem, 2008). Maintaining index at the server side can be found in (Feng and Jonker, 2003; Wang and Lakshmanan, 2006; Lee and Whang, 2006; Jammalamadaka and Mehrotra, 2006) and maintaining index at the client side can be found in (Gao et al., 2008; Yang et al., 2006).

3.4.1 Hash function based index building

Feng and Jonker built the index information using hash function. The basic idea is to augment encrypted XML data with encodings which characterize the topology and contents of each XML data, and then filter out candidate data for decryption and query execution by examining query conditions against these encodings. The searching encrypted XML data is comprised of three phases (Feng and Jonker, 2003): query preparation, identify candidate, and query execution.

- Query preparation

In this phase, XML data and DTD are encoded before encryption using hash function $HashFunc(p)$. Each node in path p is calculated with *Base26Value* and calculated the module of the hash table size, which is assigned by the user. The encoding result is a pair of element and relative value (c_{name}, c_{val}). An example is shown in Table 3.2.

Table 3.2 An example of pairs of element and value with their hash values

Element/Attribute c_{name}, Value c_{val}	$HashFunc(c_{name})$	$HashFunc(c_{val})$
c_1 = (Name, “Baolong Liu”)	0	1
c_2 = (number, 3209446589721205)	1	10
c_3 = (Issuer, “HSBC”)	3	0
c_4 = (Expiration, “04/12”)	2	25
…	…	…

- Identify candidate using hashing paths

  Given a query, it can be matched to a path p, and compute the hash value for p using the same hash function $HashFunc(p)$, then consult with the table generated in phase 1 to obtain possibly items containing path p.

- Query execution

  The identified results from phase 2 are decrypted into plaintext, on which the query can be executed.

The major contribution of the method is using hash function to generate XML data structure encodings. When DTD for XML document has been changed, it needs to re-compute all related hash values. In consequence, it is inefficient when XML document is updated frequently. In addition, the hash function adopted may generate hashing collision, and this needs to be resolved.

3.4.2 Discontinuous structural index (DSI)

A discontinuous structural index (DSI) for encrypted XML data has been proposed by Wang and Lakshmanan (Wang and Lakshmanan, 2006). The DSI is built based on interval-based labeling scheme. In DSI, the root node has been assigned the interval [0,1], the children nodes are assigned an interval which within the range of their parent's interval. Two index tables are used for the structural index. One of it is the encrypted XML data block as shown in Table 3.3 (a), and another is the DSI table as shown in Table 3.3 (b).

Table 3.3 Structural index tables

(a)

ID	Represented Interval
1	[0.27, 0.32]
2	[0.65, 0.659]

(b)

Tags	DSI index
PaymentList	[0, 1]
PaymentInfo	[0.14, 0.46]
Name	[0.16, 0.2]
CreditCardInfo	[0.27, 0.32]
…	…

In the query processing, the query processor translates the query into encrypted form against the structural index table. The processor replaces each element name in XPath with corresponding encrypted tags in the structural table (Ünay and Gündem, 2008). The encrypted block id can be found by joining query the two tables.

The major disadvantage of Wang's scheme is that it increases data size by scaling encrypted XML data, and this will increase the time cost in query processing. Another disadvantage is that the scheme cannot satisfy the security against inference attack. An inference attack is a data mining technique performed to analyzing data in order to illegitimately gain knowledge about a subject (Krumm, 2007). In scheme DSI, attackers may infer nodes relationship or infer whether a node resides in encrypted block by using DSI index. In addition, this scheme is not efficient in XML data insertions when the data updating frequently (Ünay and Gündem, 2008).

3.4.3 Query-Aware decryption

Lee and Whang proposed Query-Aware decryption for encrypted XML data (Lee and Whang, 2006). Based on Query-Aware scheme, Xia designed architecture for XML encrypted data querying (Xia et al., 2009). In these schemes, the index information is kept at the server side. The index information consists of three columns. The first column is the key name. The second column is element name, and the third is the occurrences, which is expressed as the Dewey number of elements in the second column (Ünay and Gündem, 2008). All three columns are encrypted using the keys in the first column. Table 3.4 shows an index for payment information.

Table 3.4 Index for payment information

Key name	Element name	Occurrences
Null	PaymentList	1
Null	PaymentInfo	1.1
Null	Name	1.1.1
K1	CreditCardInfo	1.1.2
Null	Address	1.1.3
K1	Number	1.1.2.1
K1	Issuer	1.1.2.2
…	…	…

The following steps illustrate the querying process. Assuming a client holds the key k1, and then submits a query "//PaymentInfo//Issuer". The query processor decrypts the field of key name using k1. The column element name is decrypted using k1. The field occurrences of the row associated with element type "Issuer", which is requested in the query, is decrypted by the processor (Lee and Whang, 2006; Ünay and Gündem, 2008). The position of element type "Issuer" is at the node with number 1.1.2.2, and encrypted data element is included in the node with Dewey number 1.1.2 (Lee and Whang, 2006; Ünay and Gündem, 2008). The node with Dewey number 1.1.2 is returned and decrypted.

Although the scheme proposed by Lee and Whang is efficient to match the XPath in querying, it has an important security issue. When a query is being processed, the key applied to decrypt index table is disclosed to server, and this may lead to potential security problems. Another disadvantage of the work is that it cannot support range query without decrypting all encrypted blocks. It will also lead to other nodes re-labelled when inserting the new XML data.

3.4.4 Scheme based on random number

Encrypted XML data querying is processed by both maintaining index information at the server side and the client side proposed by Schrefl (Schrefl et al., 2005; Ünay and Gündem, 2008). In the presented scheme, each possible path is stored with unique identifier as shown in Table 3.5.

Table 3.5 Each possible path stored at the client side

Path ID	Path Schema
PS1	PaymentList/PaymentInfo*/Name
PS2	PaymentList/PaymentInfo*/CreditCardInfo
PS3	PaymentList/PaymentInfo*/Address
PS4	PaymentList/PaymentInfo*/Amount
PS5	PaymentList/PaymentInfo*/ CreditCardInfo/Number
…	…

There are two kinds of hash results maintained at the server side. First table uses path instances as key and the second table uses path values as key as shown in Table 3.6 (a) and Table 3.6 (b) (Schrefl et al., 2005; Ünay and Gündem, 2008).

Table 3.6 Index information at the server side

(a)

Cryptographic Hash (PI)	E(value, k, nonce)	Nonce
H(PS1-1)	E(Baolong Liu, k, 10)	10
H(PS1-2)	E(Jack Xia, k, 11)	11
H(PS5-1)	E(3209446589721205, k, 12)	12
H(PS5-2)	E(446534762218 5421, k, 13)	13
…	…	…

(b)

Cryptographic Hash (PS-V)	E(PI*, k, nonce)	Nonce
H(PS1-Baolong Liu)	E({1}, k, 14)	14
H(PS1-Jack Xia)	E({2}, k, 15)	15
H(PS5-3209446589721205)	E({1}, k, 16)	16
H(PS5-446534762218 5421)	E({2}, k, 17)	17
…	…	…

Based on index information above, the querying process is described as follows, assuming that the client submits a query is:

/PaymentList/PaymentInfo/[Number=”3209446589721205”]/Name.

The client retrieves the path id of /PaymentList/PaymentInfo*/CreditCardInfo/Number is PS5 in Table 3.5. The client then computes H(PS5-3209446589721205), the value returned is E({1}, k, 16). Firstly, the client decrypts the reply using key k together with nonce and finds out that the answer is as first instance {1} of “PaymentInfo” in the path. Secondly, the client filters the card number path and adds the “/name” path to the query. The client knows that /PaymentList/PaymentInfo/Name path is PS1. Now the query becomes /PaymentList/PaymentInfo[1]/Name which is PS1-1. The client executes function H(PS1-1), finally the server returns the encrypted value with its nonce E(Baolong Liu, k, 10). Through decryption results returned, the client can get the final query results expected.

The approach adopts random number to prevent frequency based attacks, because the same plaintext can get different encryption results with different random number. One of the disadvantages is the multiple rounds of communication between the server and the client when a query is processed, and it has a high requirement on bandwidth (Ünay and Gündem, 2008). Another disadvantage is that it cannot support range query. In addition, the computing hash function is a time-consuming task, when XML data is changed, it is need to re-compute the hash results for XML data, and it will increase the system burden.

3.5 Summary

This chapter has investigated the current situations of XML security. Two main approaches for revocation information validation for X.509 digital certificate were investigated. CRLs are too bandwidth and cannot support a good degree of timeliness. The improved CRLs still has a high data volume download. The CA is required to create a signature for each query in OCSP, so the communication burden is increased between the server and clients.

Existing integrity models only generate a hash value for XML data content. Using Merkle hash function to generate hash values has a low efficiency because the process will increase the numbers of virtual nodes, and the hash times will also be increased because of increased virtual nodes. Without considering XML data features, these solutions cannot protect the structure integrity and context-referential meaning. This results that a signed XML data can be copied to another document but still keeping signature valid.

The main drawback of repeated DSA or RSA of XML multi-signature is that the size of a multi-signature result grows with the increasing of the number of signers, and the time for verifying the multi-signature is equal to the total time for verifying all personal signatures individually. Lu first presented signing XPath expression instead of the message itself. In this scheme, the document must be delegated entirely; otherwise the integrity checking will be invalid. The subdocument integrity checking needs the signers to check cooperatively online. When the group of signers is small, this is possible, but it is impractical when the group of signers is very large. The scheme only provides broadcast (parallel) signature-generation scenarios. It cannot satisfy the signing process under a dependent multi-signature situation.

In a scheme for encrypted XML data query based on index information mechanism, two major points should be considered. The first is that avoids unnecessary encrypted blocks being decrypted, and most of existing scheme achieved this objective. Considering frequently changing of XML data, the efficiency of index information updating should be considered. The second point has not been taken into account by researchers. Most cases in existing literatures, the XML document update will lead to a global index information updating. There should have a scheme with considering updating efficiency for index information. In addition, the sensitive nodes in internal structure of XML data are confidential, so simply substituting values by crypto-index may infer the structural information to the third party.

Chapter 4 XML-based X.509 digital certificate

A novel revocation information validation approach for X.509 digital certificate is proposed based on XML digital signature technology. Two-party identity authentication process for presented approach is described. The evaluation is also made to verify the efficiency of improved X.509 certificate.

4.1 Introduction

X.509 digital certificate plays an important role in identity authentication. Although XKMS makes PKI deployment easily, it still needs to check the validation of a digital certificate. One of the main concerns associated with digital certificate is that mechanism for revocation validation of certificate is required (Nielsen and Hamilton, 2005; Rivest, 1998; Housley et. al, 2002; Liu et. al, 2008; Noor, 2008).

Investigation on revocation information validation for X.509 digital certificate has demonstrated that existing techniques for certificate revocation validation are the bottleneck of a PKI or XKMS system. To alleviate the problem, this chapter proposes a novel idea to check certificate revocation information validation by using XML signature technology. XML signature technology enables a user to sign arbitrary portions of the message. After the XKMS issued a new certificate, the certificate owners can add additional information for the latest status of the certificate with their signature. Certificate owner’s signature is only used to provide evidence for revocation information of the certificate. It does not need to query XKMS or CA for revocation information of such certificate because the certificate already contains the status information.

4.2 Structure of X.509 certificate

The structure of an X.509 v3 digital certificate in XML format is shown in Figure 4.1 (ITU-T, 1997).

```
<Certificate>
  <Version/>
  <SerialNumber/>
  <AlgorithmID/>
  <Issuer/>
  <Validity>
    <NotBefore/>
    <NotAfter/>
  </Validity>
  <Subject/>
  <PublicKeyInfo>
  <PublicKeyAlgorithm/>
  <SubjectPublicKey/>
  </PublicKeyInfo>
  <SignatureAlgorithm/>
  <CerificateSignature/>
</Certificate>
```

Figure 4.1 Structure for an X.509 v3 certificate

Based on the structure of X.509 v3 shown in Figure 4.1, the mathematical symbol expression of X.509 v3 digital certificate is

$$Cert_A = (ID_A \parallel K_{pub,A} \parallel S((ID_A \parallel K_{pub,A}), K_{priv,CA})) \tag{4.1}$$

where, $Cert_A$ is an original X.509 certificate for entity A issued by a CA. ID_A denotes identity of entity A, such as subject name or email address. The ID_A corresponds to the element <Subject> in Figure 4.1. $K_{pub,A}$ is entity A's authenticated public key from the current date D_i to the future date D_e, where $K_{pub,A}$ corresponds to element <PublicKeyInfo>, D_i corresponds to element <NotBefore>, and D_e corresponds to element <NotAfter>. "||" denotes the concatenation operator. $S(M, K_{priv})$ represents signature algorithm in Figure 4.1. M is the message to be signed, and K_{priv} is the private key. The certificate provides a binding of identity ID_A to public key $K_{pub,A}$ with CA's signature.

4.3 XML-based X.509 certificate (X-certificate)

Based on advantages of XML signature technology, the thesis makes an improvement on X.509 certificate to improve the efficiency of digital certificate revocation validity. The improved x.509 certificate is named as X-certificate. The X-certificate contains two parts, the first is the X.509 digital certificate, and the second part is the status information added by certificate owner.

4.3.1 Definition for X-certificate

The basic idea is that after received $Cert_A$ from CA, the certificate owner can attach revocation information at the end of $Cert_A$ to generate new certificate C_{xml}. The X-certificate is defined in formula (4.2).

$$\begin{aligned} C_{xml} &= (Cert_A \parallel S((SN \parallel T^{'}), K_{priv,A})) \\ &= (Cert_A \parallel S((sn_1, sn_2, \ldots, sn_i \parallel T^{'}), K_{priv,A})) \\ &= ((T \parallel ID_A \parallel K_{pub,A} \parallel S(h(ID_A \parallel K_{pub,A}), K_{priv,CA})) \\ &\quad \parallel (S(h(sn_1 \parallel sn_2 \parallel \cdots \parallel sn_i \parallel T^{'}), K_{priv,A}))) \end{aligned} \tag{4.2}$$

- where, $Cert_A$ is the certificate issued by CA. This is used to provide the binding of entity A to relative public key $K_{pub,A}$.
- SN denotes revoked certificates, and $SN = \{sn_1, sn_2, \ldots, sn_i\}$. where, sn_i denotes the reference number of revoked certificates. This information can be obtained from CA and then checked by certificate owner.
- T is the timestamp of CA's signature, and $T^{'}$ is the timestamp of certificate owner's signature.
- h is a one-way hash function which is used to generate hash values.
- $S(M, K_{priv})$ denotes signing function of certificate authority, M is the message to be signed, and K_{priv} is the private key. The formula (4.2) should include two signatures generated by CA and certificate owner respectively, they only sign information which they are responsible for.

When a client holds an entity's certificate, it is an invalid certificate if the series number belongs to SN. Otherwise, the client can confirm present status by verifying two signatures without querying the CA. If one of the two signatures is invalid, the certificate is invalid. When CA's signature is invalid, it means that the identity is not identical to public key, and when certificate owner's signature is invalid, it indicates that the certificate has been revoked. Compared to using a certificate server, the X-certificate's status checking is off-line, and this approach reduces the complexity of XKMS or PKI systems because it eliminates the requirement for additional revocation checking from certificate server. Correspondingly, the communication burden between server and client is alleviated.

4.3.2 Two-party authentication process based on X-certificate

A digital certificate records the information necessary for encryption or verifying digital signature. The protocol for authentication and confidentiality of X-certificate can be described as follows.

- Authentication process for purpose of verifying signature

Step 1: $EntityA(K_{priv,A}) \xrightarrow{C_{xml}, S(M, K_{priv,A})} EntityB$

Where, $K_{priv,A}$ is the private key of entity A. C_{xml} is the certificate signed by CA and certificate owner A. $S(M, K_{priv})$ is the signed information to be transferred to entity B. Step 1 can be described as: entity A sends signed information with X-certificate to entity B.

Step 2: $EntityB \xrightarrow{C_{xml}} \{valid, invalid\}$

With the certificate C_{xml}, entity B can obtain public key K_{pub} of entity A. Entity A's signature can ensure the status of K_{pub}. The signature of CA ensures the identity of entity A binding to relative public key K_{pub}. Step 2 is that entity B verifies the validity of received certificate. This step includes two sub-steps: verify the identity and relative public key with CA signature, and check certificate status by verifying signature of entity A. If certificate C_{xml} received is valid, it can be used to verify the signed information.

- Authentication process for purpose of encryption

Step 1: $EntityB \xrightarrow{requestC_{xml}} EntityA$

This process shows that entity B wants to obtain the public key of entity A, and send a request to entity A for certificate C_{xml}.

Step 2: $EntityA(K_{priv,A}) \xrightarrow{C_{xml}} EntityB$

This round describes the entity A sending the certificate to entity B the same as step 1 for verifying signature purpose, but without additional signed information.

Step 3: $EntityB \xrightarrow{C_{xml}} \{valid, invalid\}$

With the certificate C_{xml}, entity B can obtain public key K_{pub} of entity A. Entity A's signature can ensure current status of delivered certificate.

4.4 Evaluation

4.4.1 Evaluation methods

The evaluation of X-certificate is divided into two parts. Firstly, the size of the required data structure is calculated. Secondly, the transferred data volume in revocation is evaluated.

4.4.2 Size evaluation

Figure 4.2 illustrates the size of an X-certificate. The size of each field is calculated by using software BERViewer v2.1.1 (Available at: http://www.freedownloadscenter.com/Utilities/Misc__Utilities/BERViewer_Download.html, accessed on October 2010). BERViewer is the software that allows user to view encoded files, such as X.509 certificate. It can also analyze each field with length and values.

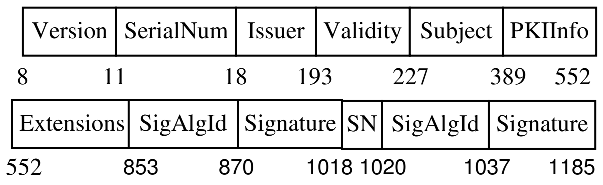


Figure 4.2 Size of X-certificate (bytes)

Table 4.1 lists the size of different kinds of validation mechanisms, size of X.509, and X-certificate. Except for X-certificate, other parameters are the same as from the ones proposed by Arnes and Hormann (Arnes, 2000; Hormann et al., 2006). The X.509 CRL is downloaded from W3C Server CA (http://ca.csail.mit.eud/drl/w3c-server.crl, accessed on October 2010). The downloaded “empty” (before any revocation) CRL is about 4 KB. The real size of X.509 CRL in Table 4.1 is also calculated by using program BERViewer v2.1.1.

Table 4.1 Size of different mechanism

Parameters	Size Description	Value (bytes)
S_{Cert}	X.509 v3 certificate	1018
S_{CRL}	X.509 CRL	39, 400
$S_{OCSP\,\mathrm{Re}\,q}$	OCSP Request	449
$S_{OCSP\,\mathrm{Re}\,sp}$	OCSP Response	459
S_{X-Cert}	X-certificate	1185

4.4.3 Efficiency evaluation

In application, a CA is assumed to manage users between $N = 1{,}000$ and $N = 100{,}000$ (Hormann et al., 2006). A typical validity period of the issued certificate is one year. For the probability of certificate revocation, this thesis takes 10%, i.e. the probability that a certificate will be revoked before its expiration. Table 4.2 lists the parameters to be measured.

Table 4.2 Parameters for evaluation

Parameters	Description	Value
N	Certificate users	1,000; 100,000
P	Percentage of certificate revoked	10
Q	Status requests per day per user	1;10;20
U	Percentage of user requesting content	1;5;10
F	Percentage of user providing content	10;50

For CRL model, the data volume transferred during one day depends on the parameters U and N. The transferred data volume during one day is determined by Eq (4.3) (Hormann et al., 2006).

$$v_{CRL} = (PN39 + 400)UN \tag{4.3}$$

In the OCSP scenario, Eq (4.4) is used to determine the data volume created during one day (Hormann et al., 2006).

$$v_{OCSP} = (S_{OCSP\,\mathrm{Re}q} + S_{OCSP\,\mathrm{Re}sp})QUN \tag{4.4}$$

The X-certificate contains the status information, and user does not need to request status information from the server, therefore, the data volume of X-certificate generated is relative to parameter N, P and U. Eq (4.5) describes the data volume created using X-certificate during one day.

$$v_{X-cert} = 1185NPU \tag{4.5}$$

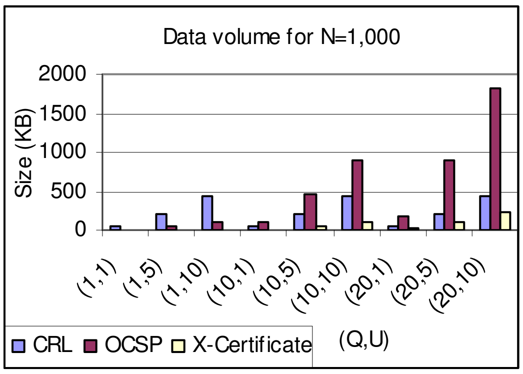


Figure 4.3 Data volume for N=1,000

Figure 4.3 shows the total data volume transferred during one day for users of 1,000. In a situation of $N = 1{,}000$, the OCSP scheme is performed worst in most of the cases. This means that the usage of OCSP creates the biggest data volume. The CRL approach performs better than the OCSP scheme except for $(Q,U) = (1,5)$ and $(1,10)$. X-certificate always shows the best efficiency in data volume transferring, compared to other two approaches.

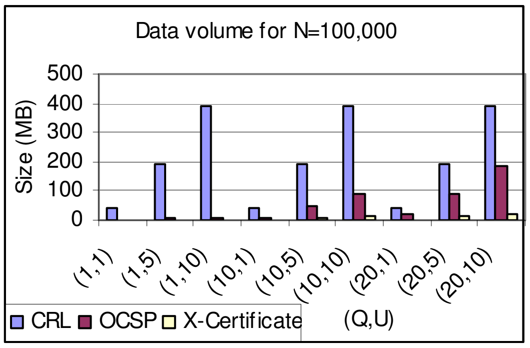


Figure 4.4 Data volume for N=100,000

For users of $N = 100{,}000$, the CRL model is obviously the most storage cost as shown in Figure 4.4. The OCSP performs better, and X-certificate shows again the best efficiency because of its simple status information validation approach. Because the X-certificate user does not need to query status information from the CA, it decreases the times of communication between and clients, and further decreases the data volume transferred. For CRL approach, the data transferred contains the user request and the size of CRL, and it increased the total data volume. In the OCSP solution, the validation includes request and response process, so it increases the data volume transferred.

4.5 Discussion and analysis

The discussion criteria listed in this section are based on a list of general criteria (Hormann et al., 2006; Arnes, 2000; Adams et al., 2001; Zhang, 2003).

- **Timeliness**

  The timeliness of CRL depends on the length of the period between the updating. The direct way to improve timeliness for CRL is to short the update period. However, the certificate server's burden will be increased significantly if the CRL is updated frequently. The timeliness of OCSP heavily depends on what approach the OCSP responder is used to gather the revocation information. Even though OCSP provides real-time replies, the revocation information carried may not be fresh if the OCSP responder acquires its information through the use of CRL. When a certificate has been invalided, the certificate owner is the first entity knowing certificate status. Based on this fact, X-certificate can update status information in time. When an entity obtains an X-certificate from certificate owner, the X-certificate has the latest status information.

- **Scalability**

Currently, two types of CRLs exist: base CRLs and delta CRLs. Base CRLs maintain a complete list of revoked certificates while delta CRLs maintain only those certificates that have been revoked since the last publication of a base CRL. The major drawback of CRLs is their potentially large size, which limits the scalability of the CRL approach (Komar et al., 2010). The large size adds significant bandwidth and storage burdens to the CA and relying party, and therefore limits the ability of the system to distribute the CRL. Bandwidth, storage space, and CA processing capacity can also be negatively affected if the publishing frequency gets too high. OCSP solved the problem of scalability experienced by CRL, because it requests certificate status on demand and only for specific certificate. The periodic downloading of large files is no longer necessary. As to X-certificate, the status information has been contained in the certificate contents, it does not need to download revocation information, and then the X-certificate has a good scalability.

- **Security**

When revocation information generated, it means that the information is from an authenticated entity, and non-repudiation. As the CRL is a 2-party scheme, only the CA has to be trusted. OCSP is a 3-party scheme, since both CA and the OCSP server have to be trusted. The OCSP server has to be trusted to gather authenticated revocation information and produce correct and digital signed responses to each request. X-certificate also is a 2-party scheme. It provides integrity, authentication, and non-repudiation through two digital signatures. The signature generated by CA ensures the public key and relative identity, and certificate owner's signature provides revocation information. X-certificate can not only provide the binding of public key to relative identity securely, but also can ensure revocation information not being forged, e.g. prevent man-in-the-middle attack.

- **Simplicity**

Simplicity means that the revocation scheme is easy to be deployed in practice. The CRL scheme is easily managed by adding a new entry to current revocation list for each update period and distributed this CRL to its repository. OCSP specifies the behaviour of the OCSP server and the OCSP end-entity. However, the management of the OCSP server is a time-consuming task, since the number of server can be quite high. X-certificate only needs to register at CA after revocation information has been changed by owner, and it will not increase additional service compared to CRL and OCSP.

- **Compatible with XKMS**

The X-certificate proposed in this chapter is an improvement of X.509 certificate. It still holds the original architecture of X.509 certificate. It can be used as an <x.509data> in XKMS without changing. With the X-certificate, the X-certificate owner has the same operation as X.509, e.g. it needs to be registration, reissue, and so on. It only offers benefits to the certificate users. With X-certificate, a client can validate the certificate easily.

4.6 Summary

A novel revocation information validation approach for X.509 digital certificate was proposed in this chapter. This approach reduces the complexity of XKMS or PKI systems because it eliminates the requirement for additional revocation checking from XKMS or CA, and in consequence, the communication burden between server and client is alleviated. The authentication processes of X-certificate show that the presented approach can satisfy identity authentication for signature and encryption purpose. Through evaluation, the approach has a higher efficiency than existing revocation checking solutions, such as CRL, OCSP. Analysis indicates that the presented approach is secure. The approach is an off-line certificate validation service, and it is easy to be deployed.

Chapter 5 XML data integrity based on concatenated hash function

This chapter presents the XML data integrity requirements. Based on the presented XML data integrity requirements, the following section builds the integrity model for XML data. The specifications for proposed integrity approach are described. The testing and evaluation are also executed.

5.1 Introduction

Existing integrity models only generate a hash value for XML data content without considering XML data features. For non-XML data formats, a user can directly generate hash value of the data content to ensure integrity, but protecting data content integrity alone is not enough for XML data. Besides data content integrity, XML data integrity should also protect element location information and element context meaning under a fine-grained security situation. Location information of an XML element refers to the position of this element in the XML data (McIntosh and Austel, 2005). An element has an entire meaning related to its position in XML data, and will lose original meaning if the position has been changed. XML data integrity should also protect location information of an XML element in XML data. Another factor which affects the meaning of XML elements is the context relationship. The element will no longer have its original meaning without context relationship in an XML data, and the thesis defines this as context-referential integrity. In other words, an XML element has an entire meaning only related to other elements in the same XML data.

This chapter aims to present XML data integrity requirements combined with XML data features. Based on the XML data integrity requirements proposed, it proposes an integrity model for XML data, and improves the efficiency of hash value-generation for XML data.

This chapter proposes an XML data integrity model named as CSR. The model consists of three parts, and CSR is an acronym for these parts: ‘C’ for content integrity, ‘S’ for structure integrity, and ‘R’ for context-referential integrity. The three parts are combined with the concatenated hash function. Content integrity is protected using the concatenated hash function. Structure integrity is used to protect the location information of an element in XML data by hashing an absolute path string from the root node. Finally, context- referential integrity protects the integrity of context-related elements. This chapter also describes the combination of the model with XML specification, and integrates the model into the XML signature.

5.2 Theory guidance for XML data integrity

In order to ensure integrity, there are means to ensure the information integrity, such as hashes or check-sum mechanisms (Geuer-Pollman, 2004). Both approaches can be used to find changes occurring in original message. But hashes are focused on malicious attack while check-sums are deployed to find coincidental changes (Brandt and Bonte, 2000).

In this thesis, data integrity is ensured by a hash function mechanism. The reasons of adopting a hash function as an integrity method is (Geuer-Pollman, 2004): checksums are usually applied in detecting accidental data changing. Checksums provide low security level against a malicious attack because their mathematical structure makes them easy to be broken. An example is CRC series. A hash function has one-way and collision-resistant features with a complex mathematical model, and it provides a higher level security than the checksum.

5.3 XML data integrity model CSR based on concatenated hash function

The integrity model to be presented is referred to the model DOM-HASH and the model proposed by Bertino although the construction process is different. The integrity model proposed by Bertino is based on Merkle hash function (Bertino et al., 2004). The integrity model CSR is constructed based on the concatenated hash function. Just like the Merkle hash function, the concatenated hash function also is designed to handle tree structure hash process. The reasons of adopting a concatenated hash function to construct the integrity model for XML data is: concatenated hash functions can handle arbitrary tree structure, but the Merkle hash function mainly deals with binary tree structure (Merkle, 1989). A concatenated hash function is more suitable to handle XML data. Concatenated hash functions can decrease the numbers of hash processes, so it has higher efficiency in hash value-generation for XML data than the Merkle hash function.

The basic idea of integrity model CSR is that content integrity, structure integrity, and context-referential integrity are combined with the concatenated hash function. This section first presents the requirements of XML data integrity, and then describes the model definition.

5.3.1 XML data integrity requirements

In order to illustrate the requirement of XML data integrity, an example is given in Figure 5.1, and it is a real application document derived from a website. Note that some details have been omitted.

```
<Certificate>
  <Title>Certificate of calibration</Title>
  <RefNumber>TDFRG</RefNumber>
  <CertificateDate>12/10/2008</CertificateDate>
  <Description>A single-mode Fibre Attention...</Description>
  <Measurements>
    <Description>The measurement of the spectral...</Description>
    <Table>Designed figure used in measurement</Table>
  </Measurements>
  <Results>
    <Description>The total attenuation...</Description>
    <Graph>Chart related to measurement results</ Graph >
    <Table>Figure of measurement results</Table>
  <Results>
        ⋮
</Certificate>
```

Figure 5.1 A certificate of calibration

- Content integrity (CI)

The XML data contents refer to element name, attribute, and values of an element or sub XML data. Content integrity means that XML data content will not be changed or destroyed in transmitting or storage. This is ensured by generating a hash value of XML data. As shown in Figure 5.1, content integrity for element ‘Title’ should include tag name ‘Title’ and its value ‘Certificate of calibration’.

- Data structure integrity (STI)

An XML data structure integrity protects the location information of an element in XML data (McIntosh and Austel, 2005). It means that if the location of an element in the XML data is changed, it will lead to an invalid verification. Location information of an XML element refers to the position of this element in the XML data. Element location information consists of three parts: parent, level, and order in sibling. This position helps users to understand the meaning of the element. An element may have different meanings when it is located in different positions in XML data. As shown in Figure 5.1, there are three ‘Description’ elements in line 04, 07, 11. The ‘Description’ element has a completely different meaning related to its location: line 04 is a description for certificate information; line 07 is a description for measurement; line 11 is the description for measured results. Location information for an XML element is an important aspect and needs to be protected.

- Context referential integrity (CRI)

When adopting XML data format, without considering element context relationship, only one element will also lose its original meaning. As shown in Figure 5.1, the measurement result has a completely meaning related to measurement method or technique deployed in the certificate. The element ‘Measurements’ and the element ‘Results’ in Figure 5.1 are generated by different responsibilities. It cannot be signed by only one user, or signed together, because each user is only responsible for own role. Under this situation, element ‘Certificate/Results’ has a completely meaning that is only related to element ’Certificate/Measurements’. It means that this kind of testing results occurrence corresponds to a specific given measurement. In other words, an XML element has an entire meaning only when related to other elements in the same XML data, and these elements are defined as context-related elements in this thesis. Another example is shown in Figure 5.2.

```
<Books>
  <Title>XML Security</Title>
  …
  <Amount>20</Amount>
  …
  <Payment>£160</Payment>
  …
  <Signature>
  …
  </Signature>
</Books>
```

Figure 5.2 An example of CRI

As shown in Figure 5.2, the signature is generated on element ‘Payment’. However, element ‘Payment’ has a complete meaning that is only relating to element ‘Amount’. The context-related element of element ‘Payment’ is the element ‘Amount’.

Context-referential integrity is used to protect context-related elements of an element in XML data. It will provide a binding between an element and context-related elements. This means if context-related elements of an element are altered, it will also lead to an invalid verification.

The basic requirement for XML data integrity is that XML data has not been changed or destroyed. Considering XML data integrity features analyzed above, the detailed integrity requirements for XML data include XML data content, which includes element name, value, and attribute, has not been changed, destroyed, or lost. Element location information, which includes element's parent, level, and order in sibling, should be protected. In order to ensure a complete meaning of an element within an XML data, context-related elements should also be protected together with this element.

5.3.2 Definition of integrity model CSR

In order to develop a model for XML data integrity, this section introduces a definition for XML data proposed by Bertino as in definition 5.1.

Definition 5.1 An XML data is tuple $X_D = (V, V_r, E_d, \phi E_d)$ (Bertino et al., 2004), where:

- $V = V^e \cup V^a$ is a set of nodes, where V^e represents elements, and V^a represents attributes. Each $v \in V^a$ has an associated attribute value; each $v \in V^e$ may have associated data content.
- V_r is a node representing the document element as called XML data root node.
- $E_d \subseteq V \times V$ is the set of edges.
- ϕE_d is the edge labelling function.

Definition 5.2 Content integrity $CI(v)$

XML content integrity should protect name, attributes, value of an element or sub XML data. Let X_D be an element or sub XML data, and h be a collision-resistant one way hash function. The $CI(v)$ associated with X_D is a function, and for each $v \in V$

$$CI(v) = \begin{cases} h(((v.content) \parallel (v.attribute)) \parallel (CI(v.child^1) \parallel \cdots \parallel CI(v.child^n))) & if\ v\ \ is\ \ a\ \ vertice \\ h((v.content) \parallel (v.attribute)) \quad if\ v\ \ is\ \ a\ \ leafnode \end{cases} \quad (5.1)$$

Formula (5.1) only provides the hash value for an element or portions of XML data, where, $v.content \in V^e$, and $v.attribute \in V^a$. h is a collision-resistant one-way hash function such as SHA256. $v.child^i (i = 1...n)$ denotes v's the i th child. "||" denotes the concatenation operator. The definition is also based on a concatenated hash function, meaning that all children of an element are concatenated together before, generating a hash value.

Definition 5.3 Label for an XML node $L(v)$

$$L(v) = C_1 C_2 \tag{5.2}$$

where, $C_1 \in$ Integer is the level of corresponding node v. $C_2 = sibling(v)$ is the order of sibling nodes, and $sibling(v)$ is the function to get sibling order of node v.

The label for element "Certificate\Results\Description" in Figure 5.1 can be expressed: $L(\backslash Certificate \backslash \mathrm{Re}\,sults \backslash Description) = 31$

Definition 5.4 Structure integrity $ST(v)$

For each $v \in V$, $ST(v) = h(path(r, v))$ (5.3)

The result is the hash value of path string related to $v \in V$, where r is the root of XML data. $p = path(r, v)$: $p \in string$, denotes a path from root r to current element v . p is an ordered sequence of one or more nodes $p \in r_{L(r)} / v^1_{L(v^1)} / \cdots / v^m_{L(v^m)} / v_{L(v)}$, and r is the root node of XML data, v^1 is the child of node r, v^m is the child of v^{m-1}, and v is the current element. $L(v)$ is the label for an internal node.

The location of an element can be expressed as a path string from root node to current node. This path records the level, sibling order, and parent of an element. Through hashing this path string, element location information would be protected.

Definition 5.5 Context referential integrity $CRI(v)$

Suppose w is the context-related element of an XML data v, $v \to w$, then,

$$CRI(v) = h(CI(w) \,\|\, ST(w)) \tag{5.4}$$

where $w \in V$. This definition includes integrity of context-related element content and its location information. Context-related elements can be selected by a signer before signing an XML data with considering context relationship.

The problem is that the context-related elements only can be selected by user instead of generating automatically. Context-related elements defined in this section concentrate on the business rules in XML data, such as dependencies, relationship attributes, so it is difficult to give common rules to select the context-related elements in practice, especially when integrity constraints for XML are still at infant stage. There are not unified types of integrity constraints for XML data, so it is impossible to integrate the integrity constraints for XML data into context-related elements selection. Under this situation, selection of context-related elements depends on constraints which are defined on the DTD by user. These constraints can be captured automatically by the system. With the development of integrity constraints for XML data, it is possible to define common rules to capture the elements which have the context-related relationship, and this point will be discussed in the section of future works.

Definition 5.6 Definition of integrity model CSR

$$CSR(v) = h(CI(v) \parallel ST(v) \parallel CRI(v)) \qquad (5.5)$$

The result of formula (5.5) is a hash value for the XML data. This value consists of three parts: $CI(v)$, $ST(v)$, and $CRI(v)$, and the three parts are combined by a concatenated hash function, where, $v \in V$ is the node set of the XML data. $CI(v)$ is a hash value of an element or sub XML data, which is used to protect the XML data content. $ST(v)$ is a hash value of element location information, which is used to protect the position of an element or sub XML data in the XML data. $CRI(v)$ is a hash value of context-related elements, and which is used to protect context relationship of an element. h is a collision-resistant one-way hash function. The combination of these three parts is by string concatenation, i.e., by hashing the concatenated string $x_1 \parallel \ldots \parallel x_l$.

In case an element copied from an XML data to another document which has the same structure as original one, the original XML data creation timestamp is used to distinguish them as defined in definition 5.7. This definition is a combination of timestamp with integrity model CSR.

Definition 5.7 Let $S = h(T \parallel CSR(v))$ be the hash value that is finally signed. Where, T is an attribute of the creation timestamp related to root node V_r for XML data X_D. It records the creation time of XML data X_D. This value is derived from function $T = Ctimestamp(X_D)$, and it obtains the timestamp of XML data creation.

5.3.3 Integrity analysis

The integrity proofs are expressed by three theorems. Theorem 5.1 provides the evidence of structure integrity, theorem 5.2 proves context-referential integrity, and theorem 5.3 proves that a signed XML data cannot be copied into another document.

Theorem 5.1 If an element $v \in V$ in XML data X_D and $X_D^{'}$, and $X_D \neq X_D^{'}$, without considering context-related elements, then $CSR(v) \neq CSR^{'}(v)$.

This theorem is used to judge the data integrity when an element copied from one XML data to another which has different structure. Because the two XML data have different structures, the element location will be changed. From the defined integrity model, they will have different hash values and lead to an invalid verification.

Proof: In the theorem, because v is the same in XML data X_D and $X_D^{'}$, and without considering context-related elements, there is the same $CI(v)$, $CRI(v)$ in X_D and $X_D^{'}$. If $CSR(v) \neq CSR^{'}(v)$, there must be different $ST(v)$ in X_D and $X_D^{'}$. In other words, v has different location in X_D and $X_D^{'}$. Location information consists of three parts: parent, level, and order of sibling.

Assuming the path from root node to current element v in XML data X_D is:

$p_1 = v_{11} / v_{2j} / \ldots / v_{ij}, i, j \in \text{int}$

Assuming the path from root node to current element v in XML data $X_D^{'}$ is:

$p_2 = r_{11} / r_{2n} / \ldots / r_{mn}, m, n \in \text{int}$

The value of $ST(v)$ in XML data X_D:

$ST(v) = h(path(p_1)) = h(v_{11} / v_{2j} / \ldots / v_{ij})$

The value of $ST(v)$ in XML data $X_D^{'}$:

$ST(v) = h(path(p_2)) = h(r_{11} / r_{2n} / \ldots / r_{mn})$

Because $X_D \neq X_D^{'}$, there are two kinds of situations:

- Different level

If v has different level in XML data X_D and $X_D^{'}$, then $i \neq m$, and $h(v_{11} / v_{2j} / \ldots / v_{ij}) \neq h(r_{11} / r_{2n} / \ldots / r_{mn})$.

Then, $CSR(v) \neq CSR^{'}(v)$. It also means element v has different ancestors.

- Different sibling order

If v has different sibling order in XML data X_D and $X_D^{'}$, then $j \neq n$, and $h(v_{11} / v_{2j} / \ldots / v_{ij}) \neq h(r_{11} / r_{2n} / \ldots / r_{mn})$

Then, $CSR(v) \neq CSR^{'}(v)$.

Theorem 5.2 An element $v \in V$ in XML data X_D and $X_D^{'}$, if the context-related element is T_1 in XML data X_D, $T_1^{'}$ in XML data $X_D^{'}$, and $T_1 \neq T_1^{'}$, then $CSR(v) \neq CSR^{'}(v)$.

The theorem 5.2 is used to check for changes in context-related elements. If the same element has different context-related elements, regardless of whether or not the two XML data have the same structure, it will lead to an invalid verification.

Proof: If $X_D \neq X_D^{'}$, from theorem 5.1, then $CSR(v) \neq CSR^{'}(v)$

If $X_D = X_D^{'}$ and $T_1 \neq T_1^{'}$, then the value of $CSR(v)$ in XML data X_D is expressed as follows:

$CSR(v) = ST(v) \| SE(v) \| CRI(v) = ST(v) \| (CI(T_1) \| ST(T_1)) \| CRI(v)$

The value of $CSR^{'}(v)$ in XML data $X_D^{'}$:

$CSR^{'}(v) = ST(v) \| SE(v) \| CRI(v) = ST(v) \| (CI(T_1^{'}) \| ST(T_1^{'})) \| CRI(v)$

$T_1 \neq T_1^{'}$ means $T_1, T_1^{'}$ have different content, or different structure.

If $T_1, T_1^{'}$ have different content, then $CI(T_1) \neq CI(T_1^{'})$ Thus, $CSR(v) \neq CSR^{'}(v)$

If $T_1, T_1^{'}$ have different structure, then $CI(T_1) \neq CI(T_1^{'})$ and $ST(T_1) \neq ST(T_1^{'})$ Thus, $CSR(v) \neq CSR^{'}(v)$

Theorem 5.3 An element $v \in V$ in XML data X_D, if X_D is signed and copied to another XML data $X_D^{'}$, it will lead to an invalid verification.

Proof: If X_D and $X_D^{'}$ have not same structure and content, then from theorem 5.1, there has $CSR(v) \neq CSR^{'}(v)$. It will lead to an invalid verification.

If two XML data have same structure and content, they should be the same XML data. An element copied from one XML data to another will not affect the validation result. However, XML data has its own creating time, which can be used to judge the validation of an element in an XML data. Therefore, the integrity model combined with timestamp, to prevent an element is being copied maliciously from one XML data to another.

Assuming $S(v)$ is the signature related to element v, so the value of $S(v)$ in XML data X_D:

$S(v) = h(h(t_1) \parallel CSR(v)))$

The value of $S(v)$ in $X_D^{'}$: $S(v) = h(h(t_2) \parallel CSR(v)))$

If X_D and $X_D^{'}$ have a different creation time, $h(t_1) \neq h(t_2)$, and it will lead to an invalid verification. If X_D, $X_D^{'}$ have a same creation time, and X_D has the same structure and content as $X_D^{'}$, this means that X_D is the same XML data as $X_D^{'}$.

5.3.4 Efficiency analysis

The following two factors affect the efficiency of model CSR: the node size and the depth size. In a $k-ary$ tree with a depth of m , and worst situation, the numbers of nodes that would be hashed is $N = \sum_{x=1}^{m} k^{x-1} = \dfrac{k^m - 1}{k-1}$, and the numbers of hash required $W = \sum_{x=1}^{m} xk^{x-1} = \dfrac{mk^{k+1} - (m+1)k^m + 1}{(k-1)^2}$.

The time complexity of an iterative hash function h can be described as a function of its input size l by the function, $T(l) = c_1(\left\lfloor \dfrac{l}{D} \right\rfloor + 1) + c_2$, where D is constant (Tamassia and Triandopoulos, 2003). If v is a vertex of XML data X_D , $in\deg(v)$ denotes the depth of vertex v, that is the numbers of predecessors of v in X_D . Let S be a sub-tree of X_D . The two components of the integrity cost for S are defined as follows. The node size S_n of S is the number of its vertices. The depth size S_d of S is the sum of the depth of its vertices, that is $S_d = \sum_{v \in S} in\deg(v)$. The rehashing overhead is given by a linear combination of the node size and the depth size of S , that is $c \mid v \mid + c^{'} \sum_{v \in S} in\deg(v) = cS_n + c^{'} S_d$, where both c and $c^{'}$ are constants. The verification time is a quantity of the form $c \mid v \mid + c^{'} \sum_{v \in S} in\deg(v)$.

5.4 Combination with XML specification

XML security has two sides: how traditional security technologies can be applied to solve security problems existing in XML data and how security technologies can be expressed in XML format. Based on the approaches proposed for XML data integrity, this section describes how the proposed model is expressed in XML format. The XML data content integrity has been described in the XML signature specification by W3C, therefore, this section only gives the description for structure integrity, and context-referential integrity.

5.4.1 Specification for structure integrity

The structure integrity is ensured by three elements as follows.

- The 'STIGenerate Algorithm' is an element, which describes the algorithm applied to generate hash values of the location information of an element in the original XML data.

- The content of the 'DigestMethod' element is the definition of hash algorithm adopted in this specification, and the default algorithm is SHA-1.

- The value of the 'DigestValue' element is the generated hash value in base64 encoding.

An example of structure integrity is

```
<STI name="structure integrity" xmlns="http://www.example.org">
  <STIGenerate Algorithm="http://www.example.org/xmldsig-csr/#STI" />
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
  <DigestValue>49-2A-ED-1A-5A-E1-BD-9C-59-04-19-58-8F-B7-08-5C-19-14-
                15-11</DigestValue>
</STI>
```

Figure 5.3 An example of structure integrity

Syntax: Schema for STI

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
   elementFormDefault = "qualified">
   <xsd:element name = "STI" type = "STIType"/>
   <xsd:complexType name = "STIType" mixed = "true">
      <xsd:sequence>
          <xsd:element ref = "STIGenerate"/>
          <xsd:element ref = "DigestMethod"/>
          <xsd:element ref = "DigestValue"/>
     </xsd:sequence>
     </xsd:complexType>
     <xsd:element name = "STIGenerate">
          <xsd:complexType>
             <xsd:attribute name = "Algorithm" use = "optional" type = "xsd:anyURI"/>
          </xsd:complexType>
     </xsd:element>
          <xsd:element name = "DigestMethod">
             <xsd:complexType>
               <xsd:attribute name = "Algorithm" use = "optional" type = "xsd:anyURI"/>
             </xsd:complexType>
          </xsd:element>
     <xsd:element name = "DigestValue" type = "xsd:string"/>
</xsd:schema>
```

Figure 5.4 Schema for STI

5.4.2 Specification for context-referential integrity

Context-referential integrity includes four elements:

- The ‘CRIGenerate Algorithm’ is an element, which describes the algorithm applied to generate the hash values of context-related elements.
- The content of the ‘RelatedNode’ is an element, which is used to record the context-related elements.
- The content of the ‘DigestMethod’ element is the definition of hash algorithm adopted in this specification, and the default algorithm is SHA-1.
- The value of the ‘DigestValue’ element is the generated hash value in base64 encoding.

An example of CRI is as follows.

```
<CRI name="context-referential integrity" xmlns="http://www.example.org">
  <CRIGenerate Algorithm="http://www.example.org/xmldsig-cri/#CRI" />
  <RelatedNode>#myData</RelatedNode>
  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
 <DigestValue>36-C3-C5-A4-02-41-A9-0F-38-B7-C1-7C-7A-A0-A5-DE-
                7D-3A-75-9</DigestValue>
</CRI>
```

Figure 5.5 An example of CRI description

Syntax: Schema for CRI

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
   elementFormDefault = "qualified">
   <xsd:element name = "CRI" type = "CRIType"/>
     <xsd:complexType name = "CRIType" mixed = "true">
         <xsd:sequence>
           <xsd:element ref = "CRIGenerate"/>
           <xsd:element ref = "RelatedNode"/>
           <xsd:element ref = "DigestMethod"/>
           <xsd:element ref = "DigestValue"/>
         </xsd:sequence>
     </xsd:complexType>
     <xsd:element name = "CRIGenerate">
         <xsd:complexType>
            <xsd:attribute name = "Algorithm" use = "optional" type = "xsd:anyURI"/>
         </xsd:complexType>
         </xsd:element>
         <xsd:element name = "RelatedNode" type = "xsd:string"/>
         <xsd:element name = "DigestMethod">
           <xsd:complexType>
             <xsd:attribute name = "Algorithm" use = "optional" type = "xsd:anyURI"/>
           </xsd:complexType>
         </xsd:element>
      <xsd:element name = "DigestValue" type = "xsd:string"/>
</xsd:schema>
```

Figure 5.6 Schema for CRI

5.5 Testing and evaluation

5.5.1 Evaluation environment

- Hardware environment

All the testing and evaluation are performed on a PC with a 2.39 GHz Pentium (R) 4 processor, 0.99GB of RAM, and the MS Windows XP operating system.

- Software environment

The software deployed in the evaluation is listed in Table 5.1. The programming language is the C#.net.

Table 5.1 Software deployed

Related models	Software
Integrity model CSR	Developed
Integrity model proposed by E. Bertino	Developed
DOM-HASH	Built-in package by Microsoft
XMark	The XML Benchmark Project

5.5.2 Evaluation methods

- Efficiency evaluation

Efficiency evaluation aims at comparing the time-consuming of hash value-generation between different models. In order to evaluate the efficiency of proposed model, XMark is used to generate XML data (Schmidt et al, 2001). For the XMark dataset, various scaling factors (0-1, incremental step is 0.1) are selected to create from 26.5KB to 113MB of documents. The DTD of XMark can be found in Appendix B. The compared integrity models are DOM-HASH, integrity model proposed by Bertino, and CSR.

- Functionality testing

In order to test functionality of proposed model based on described XML data integrity requirements, testing criterion is listed in Table 5.2.

Table 5.2 Testing criterion description

Functionality	Aims
XML data content integrity	The XML data content, including element name, value, and attribute, must not be modified in transit
Structure integrity	Element location information, including element parent, level, and order of sibling, has not been changed
Context-referential integrity	The model can protect context-related elements of an element. The protection of context-related elements include content integrity, and structure integrity

The testing cases are categorized by element numbers. There are three kind of category: element number is 1, element numbers are equal or bigger than 2, and the whole XML data. The reasons of choosing this kind of category are: CSR based XML signature will be the same as XML signature when XML data has only one element. The algorithms of STI and CRI both are based on iterative, so there will no effect on CSR model no matter how many or how deep the elements will be hashed. When signing the whole XML data, XML data content integrity can ensure the structure integrity and context-referential integrity. Because the signed XML data contains all information related to structure and context relationship.

Table 5.3 lists 13 testing cases under different situation.

Table 5.3 Testing cases

Element Numbers	Case No	CI	STI	CRI	Description	Expected Result
1	1.1	√	×	×	Only change content of signed information	Both XML signature and CSR based XML signature has the same verification result of invalid.
	1.2	√	√	√	Check all integrity properties with one element.	
	1.3	√	×	√	Check content integrity and context referential integrity with one element	
	1.4	√	√	×	Check content integrity and structure integrity with one element	
>=2	2.1	√	×	×	Only change content of signed information with more than one element.	
	2.2	√	√	√	Check all integrity properties with more than one element.	
	2.3	√	×	√	Both change content and context-related element of signed information	
	2.4	√	√	×	Both change content and location of signed information	
	2.5	×	×	√	Only change context-related element of signed elements	XML signature: valid CSR based signature: invalid
	2.6	×	√	×	Only change location of signed information	
	2.7	×	√	√	Both change context-related element and location of signed information	
Whole XML data	3	√	N/A	N/A	Signed the whole XML data	Both XML signature and CSR based XML signature has the same verification result of invalid.
N/A	4	N/A	N/A	N/A	Signed portions of XML data with timestamp	XML signature: valid CSR based signature: invalid

√: Denote checking this property ×: Denote without checking this property

Based on testing cases and algorithms, the parameters need to be provided when executing the test process is listed in Table 5.4.

Table 5.4 Parameters

Parameter name	Constraints	Description
Doc	Null, an XML data	The whole XML data need to be handled
DataSign	Null, the whole Doc, or portions of Doc	Elements need to be signed
Sdata	Null, element of Doc, or set of elements	Context-related elements of a signed XML data
Identifier	Null, user private key	User private key is used to sign XML data

5.5.3 Evaluation results

The integrity model proposed by Bertino is based on the Merkle hash function. The model CSR in this thesis is based on a concatenated hash function. DOM-HASH is also based on an iterative algorithm. When all of them have the same node size, the efficiency depends on the depth of XML data. There are five elements on each level in this testing. Let $H_i, i \in N$ be the depth of XML data, and the time requirement is expressed as $T(H_i), i \in N$. The comparison is made based on two different hash algorithms, SHA-1 and SHA256 as shown in Figure 5.7 and Figure 5.8

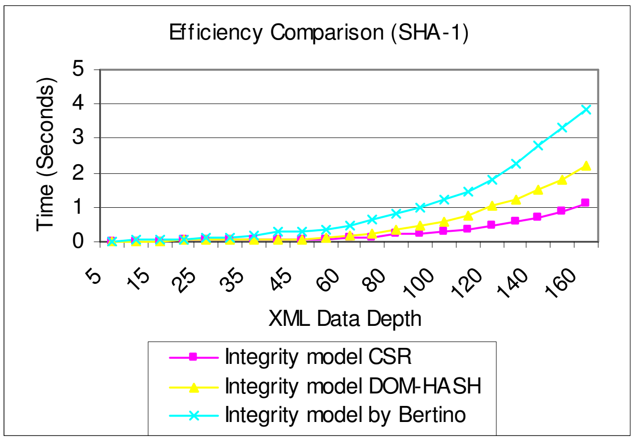


Figure 5.7 Efficiency comparison based on SHA-1 for XML data depth

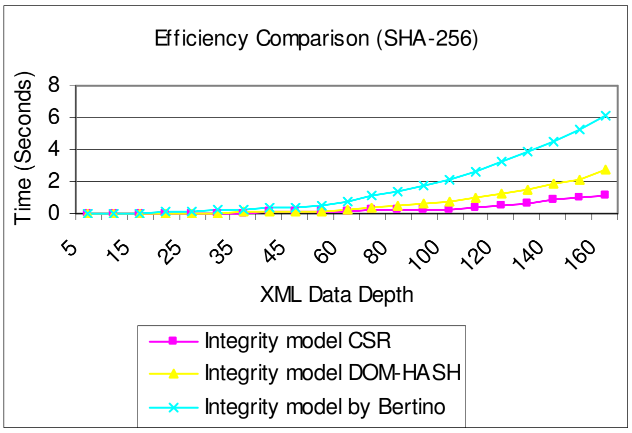


Figure 5.8 Efficiency comparison based on SHA-256 for XML data depth

Figure 5.7 shows that, these models have almost the same efficiency when XML data depth is less than 30. When the XML data depth is increased, the concatenated hash function-based integrity model CSR has the highest efficiency compared to integrity model DOM-HASH and integrity model proposed by Bertino. The integrity model DOM-HASH has a higher efficiency when compared to integrity model proposed by Bertino, and this is obvious when XML data has a higher depth. It can be calculated that the model CSR has 49.03% higher efficiency than DOM-HASH, and 74.72% higher efficiency than the integrity model proposed by Bertino. Figure 5.8 has the same development trend as Figure 5.7, but because the algorithm SHA256 is slower than SHA-1, the total time overhead is increased as shown in Figure 5.8. This indicates that although different hash algorithms have an impact on efficiency, the integrity model CSR is still the most efficient under different hash algorithms, and this is determined by integrity model mechanism, having nothing to do with adopted hash algorithms.

Without changing node size and numbers, when these nodes are at the same level, and it is defined as XML data width, the model CSR also is the most efficiency than others model as shown in Figures 5.9 and 5.10. Compared to Figure 5.7 and Figure 5.8, XML data depth has a significant impact on XML data integrity generation process.

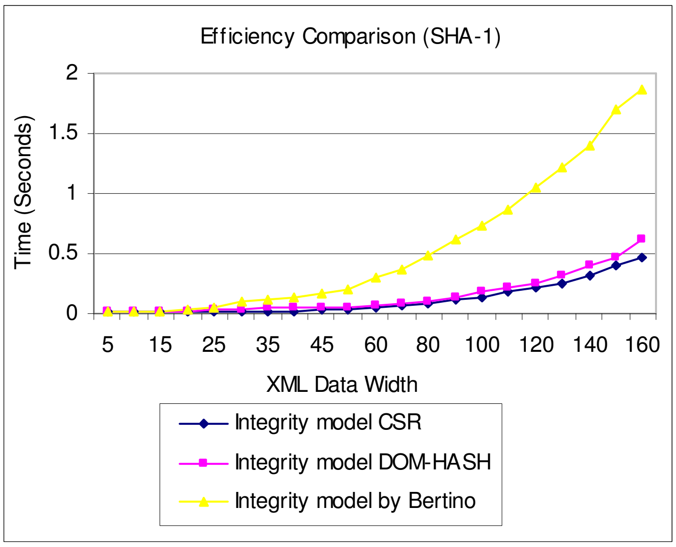


Figure 5.9 Efficiency comparison based on SHA-1 for XML data width

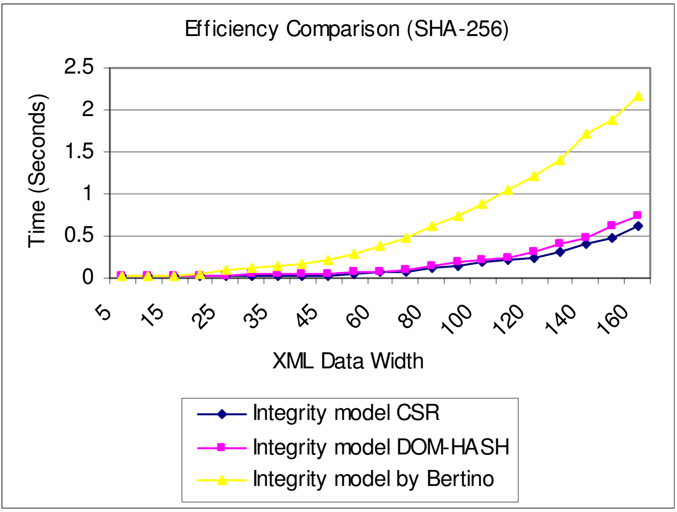


Figure 5.10 Efficiency comparison based on SHA-256 for XML data width

The reason of this result is the different numbers of hash computations in the three models. Figure 5.11 shows the total hash times of the three integrity models used in the testing cases. Bertino's model hashes the leaf node with $h(h(v.val) \| h(v.name))$, and there are 3 hash processes for each element. DOM-HASH hashes the leaf node with $h(v.elem \| v.text \| v.pi \| v.attr)$, and there is only 1 hash process for each element. In model CSR, the leaf node returned directly with 1 hash process, and the non-leaf node will have 2 hash processes.

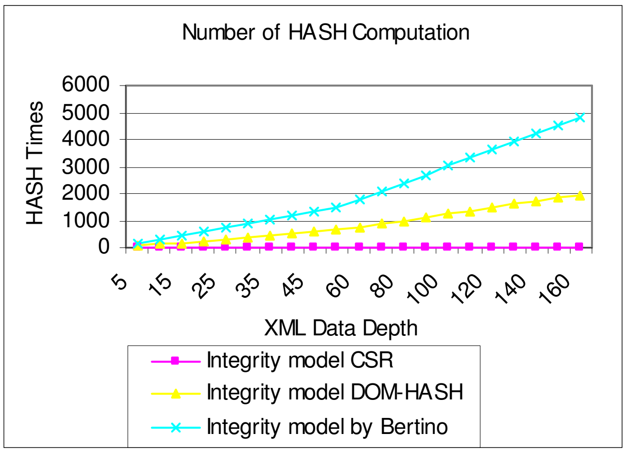


Figure 5.11 Comparison for numbers of hash computations

Based on the Merkle hash function, hashing the leaf node will increase virtual nodes, and then increase the node numbers which need to be hashed, which can lead to a low efficiency. Based on concatenated hash function, this thesis concatenates the child node firstly, and then generates a hash value. It has been proved that increasing hash numbers will not improve the security of hash function (Joux, 2004). Therefore, the model presented has the same security level as DOM-HASH and Bertino's integrity model, but because of decreased hash times, the presented hash process has a higher efficiency.

5.5.4 Testing results

Table 5.5 shows the testing results based on testing cases in Table 5.3.

Table 5.5 Testing results based on testing cases

Sample No	XML signature		CSR based XML signature	
	Signing	Verification	Signing	Verification
1.1	Successful	Invalid	Successful	Invalid
1.2	Successful	Invalid	Successful	Invalid
1.3	Successful	Invalid	Successful	Invalid
1.4	Successful	Invalid	Successful	Invalid
2.1	Successful	Invalid	Successful	Invalid
2.2	Successful	Invalid	Successful	Invalid
2.3	Successful	Invalid	Successful	Invalid
2.4	Successful	Invalid	Successful	Invalid
2.5	Successful	Valid	Successful	Invalid
2.6	Successful	Valid	Successful	Invalid
2.7	Successful	Valid	Successful	Invalid
3	Successful	Invalid	Successful	Invalid
4	Successful	Valid	Successful	Invalid

As shown in Table 5.5, all the testing results correspond to expected result as described in Table 5.3.

- XML data contains only one element

When XML data consists of only one element, it includes all the information of structure and context relationship, so CSR based XML signature is the same as XML signature as shown in cases 1.1, 1.2, 1.3, and 1.4, they have the same testing results.

- Signing the whole XML data

When signing the whole XML data, the hash value of XML data content should have contained XML data structure integrity and context-referential integrity, and hashed result is included in the signed information. CSR based XML signature is the same as XML signature, they also have the same testing result, and this can be verified in case 3.

- Signing portions of XML data when XML data elements are more than one element

  When signing portions of XML data, user can make a choice to ensure CI, STI, CRI, or all. When changed the contents of signed information, both XML signature and CSR based XML signature can detect this change and lead to an invalid verification result as shown in cases 2.1, 2.2, 2.3, and 2.4. When changing location information, context-related element of signed elements, or both of them, XML signature still keeps a valid verification result as shown in cases 2.5, 2.6 and 2.7. CSR based XML signature can detect this change and lead to an invalid verification result as shown in cases 2.5, 2.6, and 2.7.

- Signed element copied to another XML data

  When portions of XML data signed, attacker can copy signed XML data to another XML data still remaining a valid verification. XML signature cannot prevent this situation happening as shown in case 4. CSR based XML signature can find this kind of attack and achieve an invalid verification.

Above testing results corresponds to expected result in Table 5.3. Therefore, proposed XML data integrity model CSR satisfies the integrity requirements for XML data presented previous, and can protect content integrity, element location information, and context-related elements for XML data.

5.6 Analysis and discussion

In order to summarize the advantages of the XML data integrity model CSR proposed, this section makes a comparison of integrity solutions as shown in Table 5.6.

Table 5.6 XML Data integrity model comparison

Model Name	Description	Hash times	Integrity objects
DOM-HASH by Maruyama (Maruyama et al., 1999)	$dos(v) = h(v.elem \parallel v.text \parallel v.pi \parallel v.attr)$ Where, v is the element set of XML data, h is a collision-resistant one-way hash function.	1	Element name, attribute, value
XHASH by Brown (Brown, 2000)	$dos(v, s) = h(v.elem \parallel v.text \parallel v.pi \parallel v.attr)$ Where, v is the element set of XML data, h is a collision-resistant one-way hash function. s is default processing of non-significant SPACE characters.	1	Element name, attribute, value
XML Data integrity by Devanbu (Devanbu et al., 2001)	$f(v) = \{ \begin{matrix} h(v) \\ h(v, f(v_1), f(v_2), \cdots, f(v_k)) \end{matrix}$ Where v is a sink node, $v_1 \cdots v_k$ are the successor of v. h is a collision-resistant one-way hash function.	3	Element name, attribute, value
XML Data integrity by Bertino (Bertino et al., 2004)	$MhXd(v) = \{ \begin{matrix} h(h(v.val) \parallel h(v.name)) \quad if \quad v \in V_d^a \\ h(h(v.content) \parallel h(v.tagname) \parallel MhXd(child(1, v)) \parallel \ldots \parallel MhXd(child(n, v))) \quad if \quad v \in V_d^e \end{matrix}$ Where, v is the element set of XML data, h is a collision-resistant one-way hash function.	3	Element name, value
XML Data integrity by Hussain (Hussain and Soh, 2004)	<Manifest> contains the data whose location is going to change and apply an XSLT transform to omit the URI attributes	N/A	Element position
XML Data integrity by Qiao (Qiao, 2007)	$\{ \begin{matrix} Info(BCD \ldots M \ldots) = h(Info(A - B)), \cdots, H(Info(A - M)), \cdots \\ U - digest : h(Info(BCD \ldots M \ldots)) \end{matrix}$ Where, Info(A-B), …,Info(A-M), … is the sub XML data, Info(BCD…M…) is the united hashed result, and h is a collision-resistant one-way hash function.	N/A	Element name, attribute, value
XML Data integrity model CSR	$CSR(v) = h(CI(v) \parallel ST(v) \parallel CRI(v))$, where, v is the element set of XML data, h is a collision-resistant one-way hash function. $CI(v)$ is the content integrity of signed elements, $ST(v)$ is the structure integrity, and $CRI(v)$ is the context referential integrity.	1	Element name, attribute, value, position, context-related elements

The similarities of the integrity model CSR compared to existing models mainly focus on two aspects. The integrity model CSR adopts a bottom-up iterative hash process as with DOM-HASH, Devanbu's, and Bertino's integrity model. The integrity model DOM-HASH, XHASH, Devanbu's model, Qiao's model, and model CSR ensure element name, attribute, and value, except Bertino's model ignored the attribute integrity of an element.

As shown in Table 5.6, only the model CSR for XML data provides overall integrity protection, including data content, element location information, and element context meaning. Based on this comparison, the major differences of the model proposed compared to others are:

- Only integrity model CSR is considering XML data features

DOM-HASH and XHASH just consider the hash objectives, and the model proposed by Devanbu and Bertino focus on the hash value-generation process. The model CSR combined the XML data features, such as the element location and context-related elements for example.

- Integrity model CSR not only ensure the integrity of data content, but also provides a method for hash value-generation process

The integrity model DOM-HASH and XHASH just provide the integrity objects which include element name, attribute, and value, without describing the process of hash value-generation process. The integrity model CSR not only ensures the integrity of data content, but also describes the hash value-generation process. Two kinds of element have been involved, the leaf node and vertices. It will directly return the hash values of content and attribute if the node is the leaf node, otherwise it will iteratively call the function.

- Bertino's model ignored the integrity attribute

The content integrity in Bertino's model is only from $h(h(v.val) \| h(v.name))$. This does not consider the integrity attribute. In integrity model CSR, the integrity content includes $v.content \| v.attribute$, and $v.content = v.name \cup v.value$.

- Different hash numbers in the models

DOM-HASH and XHASH hash the leaf node from $h(v.elem \| v.text \| v.pi)$, and there are 1 hash processes in total. Devanbu's and Bertino's model hashes the leaf node from $h(h(v.val) \| h(v.name))$, and there are 3 hash processes in total. In integrity model CSR, the non-leaf node returned directly using $(v.content) \| (v.attribute)$ without hash process, and there is only 1 hash process for leaf node.

5.7 Summary

This chapter presents overall XML data integrity requirements combining XML data features. An integrity model is also presented based on the concatenated hash function to protect the requirements presented. The testing results show that the integrity model proposed not only ensures XML data content integrity, but also protects the structure integrity and elements' context relationship within an XML data. With this approach integrated into XML signature technology, the signature cannot be copied to another document still keeping valid. This indicates that the presented model overcome the limitations existing in XML signature specification. Integrity model CSR not only provides a model for XML data integrity, but also provides a method for the hash value-generation process. The integrity model has been verified a higher efficiency on hash value-generation than the Merkle hash function-based integrity model for XML data.

Chapter 6 A Series-parallel XML Multi-signature Scheme for XML Data Authentication

This chapter firstly describes series-parallel signing group and then XML data integrity-checking pool is presented. Based on series-parallel signing graph and off-line XML data integrity-checking approach, a series-parallel XML multi-signature scheme for XML data is proposed. The testing and evaluation are also executed in this chapter.

6.1 Introduction

XML data authentication is important research area related to XML security (Bertino, 2001). General applications of data authentication could exist in many domains. For example, a user contacting a mirror site would need to cryptographically validate the information as genuine, that is, as being the same information as if the response had come directly from the source (Polivy and Tamassia, 2002; Damiani et al., 2002).

A document is delivered through a hierarchical network of responsibilities with different roles and access rights. An example has been given by Leung and Hui to describe this situation. The computing department of a university would like to renovate its staff room so as to meet the contemporary hardware requirements (Leung and Hui, 2001). The requirement has to be approved by the Financial Office. The subsequent approval from the Estate Office will depend on the signature of the Financial Office (Leung and Hui, 2001). The approval of the Estate Office is based on the approval of the Financial Office. Traditional digital signature approach focuses on signing the entire document, and the XML signature specification is infeasible to make complex workflows secure on an XML data with multiple signatures (Leung and Hui, 2001). Under this situation, it is necessary to build an XML multi-signature scheme which is compatible with a dependant signing process.

This chapter proposes a series-parallel XML multi-signature scheme based on Lu's model (Lu and Chen, 2004). The series-parallel XML multi-signature scheme presented is a mixed-signing order including both dependent and independent signing process. In proposed scheme, signers are divided into series or parallel subgroups and the members in the signer group can be flexibly managed. The signing order is generated before the signing process without a relationship to multi-signature scheme. This scheme uses XPath expression to transform XML data, and generates an XML data integrity-checking pool to provide integrity-checking for decomposed XML data. With an integrity-checking pool, a signer can check integrity without cooperation from other signers. XML data does not need to be delegated entirely, and signers can complete integrity verification off-line. If there is a single signer, the scheme is compatible with single XML signature. When each subgroup has a single signer, the scheme is compatible with a sequential multi-signature scheme. When all signers are in the same subgroup, the scheme is compatible with a broadcast multi-signature scheme.

6.2 Theory guidance for data authentication

There are two mechanisms to ensure data authentication:

- Message authentication code (MAC)

  MAC, a cryptographic check value, is used to provide data origin authentication and data integrity (ISO/IEC, 1997). Both data integrity and data origin authentication can only be provided for the receiving entity. A third party cannot verify these properties, as both sender and receiver are capable to create the MAC (or HMAC).

- Digital signature

  Digital signature allows a recipient of the message to prove the source and integrity of the message and protect against forgery (ISO 7498-2, 1989; Georgiadis et al., 2002). More specifically, the using of asymmetric encryption provides a means to ensure the authentication, also known as non-repudiation (Brandt and Bonte, 2000).

In this chapter, data authentication is ensured using digital signature. The reasons of adopting digital signature as the data authentication method is: digital signature can be used to support requirements for non-repudiation. This is because access to the private key is usually restricted to the owner of the key, which makes it easier to verify proof of ownership. W3C has developed the technology of XML signature for XML data authentication. The new scheme should be compatible with XML signature specification.

6.3 A series-parallel XML multi-signature scheme

6.3.1 Series-parallel signing group

- Signing order graph

In order to represent signing orders, among n signers, series-parallel graph is deployed, which is a directed acyclic graph as shown in Figure 6.1.

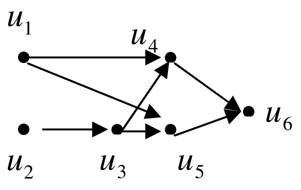


Figure 6.1 Signing order graph

A directed acyclic graph $\varphi = (V, E)$ consists of set V of nodes and a set of E edges connecting pairs of distinct nodes. For an edge e between two nodes, the initial vertex of the edge is represented by $I_e (I_e \in V)$, and the terminal vertex is represented by $T_e (T_e \in V)$. The signers correspond to the vertices in the graph φ.

- The rules for series-parallel signing group

Given signers group $SG = \{u_1, u_2, \ldots, u_n\}$, it can be divided into several ordered subgroups according to the following rules.

1. Given signer group SG, it can be defined as $SG = G_1 \cup G_2 \cup \ldots \cup G_n$, and $G_1 \cap G_2 \cap \ldots \cap G_n = \phi$, where G_i is the sub set of SG, ϕ denotes an empty set. The signing order is $G_1, G_2, \ldots, G_n$, and this means that $G_1, G_2, \ldots, G_n$ is signing in sequential.
2. For $\exists(u_i) \in G_k$, $\exists(u_j) \in G_m$, if $k = m$, then $u_i, u_j \in G_k (= G_m)$, and u_i, u_j can sign parallel. In other words, the signers who are in the same subgroup can sign in parallel.
3. For $\exists(u_i) \in G_k$, $\exists(u_j) \in G_m$, if $k < m$, then u_i, u_j should sign sequentially, and u_i should sign before u_j.
4. For $\exists(u_i) \in G_k$, $\exists(u_j) \in G_m$, if $k > m$, then u_i, u_j should sign sequentially, and u_j should sign before u_i.
5. Only the groups obtained by the rules (1), (2), (3), and (4) are series-parallel signing groups.

- Signing order graph conversion to series-parallel signing group

The following steps illustrate how to convert a signing order graph to a series-parallel signing group.

Assume $G = G_1 \cup G_2 \cup \ldots \cup G_n$ and let $G_{k(k=1,\ldots,n)} = \phi$.

Step 1: With a labelled edge $e \in \varphi$, where the initial vertex is I_e, and the terminal vertex is T_e.

Step 2: If $I_e \notin G$, then let $I_e \in G_1$. If $T_e \notin G$, then let $T_e \in G_2$.

Step 3: If $I_e \in G_{k(k=1\ldots n)}$, and $T_e \notin G$, then let $T_e \in G_{k+1}$. Otherwise, assume $T_e \in G_{m(m=1\ldots n)}$. If $m \le k$, then move T_e from G_m to G_{m+1}, until $m > k$.

Step 4: Go to step 1 until each edge in φ has been handled.

According to above algorithm, the signing order graph can be converted to the following series-parallel signing group.

$G_1 = \{u_1, u_2\}, G_2 = \{u_3\}, G_3 = \{u_4, u_5\}, G_4 = \{u_6\}$

This means signers can generate a parallel signature in each subgroup, where every subgroup signing is sequential. The converted signing order group of Figure 6.1 is shown in Figure 6.2.

$$\begin{tikzcd}\boxed{\{u_1, u_2\}} \arrow[r] & \boxed{\{u_3\}} \arrow[r] & \boxed{\{u_4, u_5\}} \arrow[r] & \boxed{\{u_6\}}\end{tikzcd}$$

Figure 6.2 Converted series-parallel signing order

6.3.2 XML data decomposition (XDD)

XML data is based on the tree structure. DOM is used to define how XML data can be accessed, and it is naturally a tree structure representation (Devanbu et al., 2001). For integrity verification purpose, the important properties of DOM-HASH are as follows.

If a signer u_i knows the hash value of a root for an XML data X_D, it is possible to prove that any sub-tree st_i of the XML data occurs under X_D without revealing all of X_D and online verification. A u_i can generate the hash value of st_i by DOM-HASH the sub-tree st_i. By given the hash value of the sibling of st_i and the sibling of all its parents, the signer u_i can compute the hash value of the root node. Based on the feature of one-way hash function and comparison of hash value, the signer u_i can judge whether the sub-tree st_i is included in the XML data X_D. This process also can be used to prove that a sub-tree st_i is contained in another sub-tree st_j without revealing other sub-tree in st_j.

Giving an XML data X_D, a DTD relative to the XML data and a pool τ with a limited number of XPath in DTD, the integrity-checking pool can be defined:

Definition 6.1 XML data integrity-checking pool τ, τ is a tuple as $(p, h(p), c(p), h(c(p)))$, where

- p is the possible XPath in the DTD.

- $h(p)$ is the hash value of each p , and h is a secure one-way hash function.
- $c(p)$ denotes the content accessed by XPath p ,
- $h(c(p))$ is the hash value of $c(p)$.

The generation process for XML data integrity-checking pool τ is:

1. Generate each possible XPath $\forall p_i \in p, i \in \{1,...,n\}$ in the DTD, and relative hash value $h(p_i)$. Insert p_i , content $c(p_i)$, and $h(p_i)$ into pool τ.
2. Build DOM-HASH associates a secure hash value $h(c(p_i))$ with each p_i, and let $m_i = h(c(p_i))$.
3. There could be many sub-trees $st_i, i \in \{1,...,n\}$ relative to XPath p_i, and these sub-trees can be hashed together using the concatenation hash function $m_i = h(st_1 \parallel st_2 \parallel \ldots \parallel st_n)$ to get a hash value each entry p_i.

For integrity verification, there is the pool τ with XPath entries, and an integrity verification request from a signer with the XPath q.

1. Match q against each entry in τ.
2. If the XPath q matches an entry p_i in τ, retrieve the hash value m_i relative to the entry p_i. If there is no corresponding entry matched to q , reject, otherwise, go to step 3.
3. Build hash value $m_i^{'}$ with step 3, check that $m_i^{'} \overset{?}{=} m_i$. If $m^{'} \neq m$, then reject, otherwise, accept. If signer does not believe in this result after accepting, the verification process can be extended to parent verification as shown in step 4.
4. Assume $q^{'}$ is the XPath of q parent, and let $q = q^{'}$, then go to step 1. Finally, signer can generate the hash value of the whole XML data X_D , check that $m_i^{'} \overset{?}{=} m_i$. If it is not equal, reject, otherwise accept. This is a convincing result, because the integrity of whole XML data has been checked.

6.3.3 XML multi-signature scheme

The system has the following roles which are similar to Wu and Lu's scheme: a group of signers, a system authority, an XDD, and a signature collector (Wu et al., 2001). SA supports to initialize system parameters, and to generate the secret keys and public keys for the group and the individual signer (Wu et al., 2001). XDD is used to decompose the XML data to a set of sub-data. Individual signatures generated by the signers are collected and verified by SC. SC also constructs a multi-signature for XML data based on verified individual signature (Wu et al., 2001). It is supposed that SA and SC can be trusted by all signers. The proposed approach consists of three stages as Wu's scheme: the stage of private key and public key generation, the stage of multi-signature generation, and the stage of multi-signature verification (Wu et al., 2001).

1. Common parameters

   The common parameters are similar to those defined in (NIST, 2006) for DSA standard to which the group dimension has been added. Assuming a group of n signers, the parameters are defined:

   - p, q: Two large prime numbers such that $q \mid (p-1)$ as defined in digital signature algorithm (NIST, 2006).
   - g : Generator of the cyclic group of order q in Z_p^* (selects an element $h \in Z_p^*$ and computes $g = h^{(p-1)/q} \bmod p$ such that $g \neq 1$).
   - $x_1, x_2, \ldots, x_n$:Group members' private keys.
   - $y_1, y_2, \ldots, y_n$: Group members' public keys such that $y_i = g^{x_i} \bmod p$ is computed.
   - (X_i, Y_i) is the key pair for each subgroup G_k, where

$$X_i = \sum_{u_{ji} \in G_k} x_j \bmod q \tag{6.2}$$

$$Y_i = \prod_{u_j \in G_k} y_j \bmod p \tag{6.3}$$

- $h(.)$: A cryptographic hash function (one-way function) such as SHA-1, SHA-256.

2. Signature generation and verification

The procedure for generating a multi-signature of X_D for G is as follows.

Step 1: XDD sends $\{\tau, X_D, T_j\}$ to u_j, and $T_j = \{p_1, p_2, \dots, p_j\}$

Step 2: Each $u_j \in G$ extracts X_D^j from X_D using T_j, and then checks the integrity of X_D^j using τ and the integrity verification process.

Step 3: If integrity of X_D^j is successfully verified, each $u_j \in G_k, j,k \in [1,N]$ randomly selects an integer $z_j \in Z_q$, computes

$$r_i = g^{z_j} \bmod p, \tag{6.4}$$

and sends $\{T_j, r_j\}$ to other participant signers in the same subgroup and SC.

Step 4: After receiving $\{T_j, r_j\}$, $u_i (i \neq j)$ and SC can compute

$$R_j = r_j^{h(T_j \| r_j)} \bmod p \tag{6.5}$$

Step 5: Each $u_j \in G_k, j,k \in [1,N]$ computes both

$$R_k = \prod_{u_j \in G_k} R_j \bmod p \tag{6.6}$$

$$s_j = (z_i h(T_j \| r_i) R_k + x_j h(h(\tau) \| R_k)) \bmod q \tag{6.7}$$

and sends $\{s_j\}$ to SC. (r_j, s_j) is the personal signature of X_D by signer u_j.

Step 6: In order to verify (r_j, s_j) for every $u_j \in G_k, j,k \in [1,N]$, SC computes R_k by Eq. (6.6) and checks whether or not the following equation holds.

$$r_j^{h(T_j \| r_j) R_k} = (g^{s_j})(y_j^{h(h(\tau) \| R_k)}) \bmod p \tag{6.8}$$

Step 7: If all personal signatures generated in the previous steps are successfully verified, then SC computes

$$S_k = \sum_{u_j \in G_k} s_j \bmod q \tag{6.9}$$

and publishes (R_k, S_k) as the multi-signature of X_D by subgroup G_k.

The verifier checks the equality to verify the subgroup multi-signature (R_k, S_k):

$$R_k^{R_k} = (g^{S_k})(Y^{h(h(\tau) \| R_k)})(\bmod p) \tag{6.10}$$

If Eq. (6.10) holds, then subgroup multi-signature (R_k, S_k) is successfully verified.

The signature of the whole group (this signature is used to ensure sequential signing order):

Step 1: SC verifies each subgroup multi-signature (R_k, S_k), if any of them are invalid, then reject, otherwise, go to step 2.

Step 2: SC computes $S_G = h(S_1 \| S_2 \| \ldots \| S_k)$, here $S_i, i \in [1..k]$ is each subgroup signature.

Step 3: The signature for subgroup G_1:

$$\sigma_1 = g^{k_1} \bmod p \tag{6.11}$$

$$\rho_1 = S_G X_1 - \sigma_1 k_1 \bmod q \tag{6.12}$$

and sends (σ_1, ρ_1) to next subgroup.

Step 4: For subgroup G_i, first verifies the signature by G_{i-1} through

$$g^{\rho_{i-1}} \prod_{j=1}^{i-1} \sigma_j^{\sigma_j} = \prod_{j=1}^{i-1} Y_i^{SG} \bmod p \tag{6.13}$$

If this generates a failed verification, then reject the signature from G_{i-1}, otherwise, compute

$$\sigma_i = g^{k_i} \bmod p\ , \tag{6.14}$$

$$\rho_i = \rho_{i-1} + S_G X_i - \sigma_i k_i \bmod q \tag{6.15}$$

Then (σ_i, ρ_i) is the final multi-signature for group SG.

Step 5: Verification for final multi-signature:

$$g^{\rho_i}\prod_{j=1}^{k}\sigma_j^{\sigma_j} = \prod_{j=1}^{k} Y_i^{S_G} \bmod p \qquad (6.16)$$

6.3.4 Correctness proofs

Since proposed scheme for subgroup signature is based on Lu's scheme, correctness of the single signature and subgroup signature is as their scheme. This section just provides the proofs of correctness of sequential signature for subgroup.

Theorem 6.1 If equation (6.13) is true, then the subgroup signature (σ_i, ρ_i) is valid

Proofs: From Eq. (6.15), for each i,

$$\sum_{j=1}^{i}\sigma_j k_j + \rho_i = \sum_{j=1}^{i}\sigma_j k_j + \rho_{i-1} + S_g X_i - r_i k_i \bmod q$$

$$= \sum_{j=1}^{i-1}\sigma_j k_j + \rho_{i-2} + S_G(X_i + X_{i-1}) - \sigma_{i-1}k_{i-1} \bmod q$$

$$= \sum_{j=1}^{i} S_G X_i \bmod q$$

Then, $g^{\sum_{j=1}^{i}\sigma_j k_j + \rho_j \bmod q} = g^{\sum_{j=1}^{i} S_G Y_j \bmod q} \bmod p$

$$= \prod_{j=1}^{i}(g^{X_i})^{S_G} \bmod p$$

$$= \prod_{j=1}^{i}(Y_i)^{S_G} \bmod p$$

The Eq. (6.13) is correct.

Theorem 6.2 If Eq. (6.16) is true, then the final signature for group is valid.

Proofs: Because Eq. (6.16) is a special expression from Eq. (6.13), for $i = k$, then Eq. (6.13) is equal to Eq. (6.16), the Eq. (6.16) is correct, and the sequential signature for group is valid.

6.3.5 Security analysis

The security of the proposed scheme is as secure as Wu's scheme because both of them are based on discrete logarithm and one-way hash function. Note that there are two particular issues that need to be addressed. The security issues related to proposed scheme:

- **Issue 1**: Forging an integrity verification table τ

  Assuming (given an XML data and a conforming DTD) that the decomposition process is executed correctly, the signer can accept a correct answer and reject an incorrect one, unless a collision in the hash function applied in decomposition process is found.

  Analysis of issue 1:

  Suppose that the signers use the DTD to compute the set of table entries which matches their XML data to be signed. Based on repeating the computation done by the decomposition process and results in the same hash value, the signers can accept correct XML data delegated to them. Now we discuss that the signers will reject any incorrect XML data to be signed. If a signer received an incorrect XML data delegated to him from an adversary, the process of computing the hash value for that entry will be different from that used to generate provided hash value. There are two ways to get a same hash value for a different XML data. First way is that a hash collision has to be found in the process of computing the hash value of delegated XML data. Alternatively, a second pre-image is found in the process of computing the hash value of delegated XML data. For these two cases, a collision should be found in the hash functions to generate the same hash value for different information. However, for a secure one-way hash function h, given $y = h(x)$, it is computationally unfeasible to find $x_1 \neq x_2$, such that $h(x_1) = h(x_2)$. The signer can reject an incorrect XML data to be signed.

- **Issue 2**: Forging a multi-signature

The signature generated by the last subgroup is the multi-signature (S_G, σ_i, ρ_i), the verification equation is Eq. (6.16). The security of Eq. (6.16) is expressed by theorem 6.3.

Theorem 6.3 It is a DL (Discrete Logarithm) problem to calculate ρ_i through (S_G, σ_i), or to calculate σ_i through (S_G, ρ_i) in Eq. (6.16).

Proofs: From Eq. (6.16), it is easy to understand that it is a DL problem to calculate ρ_i through (S_G, σ_i).

Given (S_G, ρ_i), then g^{ρ_i} and $\prod_{j=1}^{i} Y_j^{S_G}$ are constants. Let $C_1 = g^{\rho_i}, C_2 = \prod_{j=1}^{i} Y_j^{S_G}$,

then Eq. (6.16) can be rewritten as: $\sigma^{\sigma} C_1 = C_2^{\sigma} \bmod p$, then has

$$(\sigma C_2^{-1})^{\sigma} = C_1^{-1} \bmod p \tag{6.17}$$

We can get $C_3 = C_1^{-1}$, and $C_4 = C_2^{-1}$ in $GF(p)$. Then Eq. (6.17) can be written as:

$$(\sigma C_4)^{\sigma} = C_3 \bmod p \text{, so, } (\sigma C_4)^{\sigma C_4} = (C_3)^{C_4} \bmod p \tag{6.18}$$

Assume $\sigma C_4 = X$, and $(C_3)^{C_4} = C$, then Eq. (6.18) can be written as:

$$X^X = C \bmod p \tag{6.19}$$

Given (S_G, ρ_i), calculation σ_i is equal to obtain X in Eq. (6.19). It is a DL problem to obtain X in Eq. (6.19).

6.3.6 Efficiency analysis

Let T_m, T_e, and T_h be the time required to perform a modular multiplication, a modular exponential, and the one-way hash function h; respectively. n is the number of signers in G; k is the number of divided subgroup for G; and i is the signer's number in subgroup G_k.

The time-consuming for generating and verifying an individual signature (r_i, s_i) is identical to Lu's scheme. The time complexities of both stages are

$O((n+2)T_m + 2T_e + 3T_h)$ and $O(T_m + 3T_e + 2T_h)$ respectively. The time complexities for generating and verifying a subgroup signature are different from the signers in the subgroup, both stages are $O((i-1)T_m + iT_e + (i+2)T_h)$ and $O(T_m + 3T_e + 2T_h)$; respectively. The worst situation is where all the signers are in the same group, that is $i = n$. The time complexities for constructing multi-signature from subgroup are $O((k+2)T_m + 3T_e + T_h)$ and $O(T_m + 3T_e + T_h)$.

6.3.7 Compatibility with XML Signature Specification

As described in proposed scheme, each signer $u_i \in G$ extracts XPath expression p_i from the set of XPath expressions T delegated to him. "Transforms" element can be used to describe p_i's content to be signed. Other information can also be defined in an XML signature. The method applied to generate hash values can be described in the "DigestMethod" element. The element "SignatureValue" can contain the multi-signature result. The proposed scheme is compatible with the XML signature specification.

6.4 Testing and evaluation

6.4.1 Evaluation environment

All the testing is performed on a PC with a 2.39 GHz Pentium (R) 4 processor, 0.99GB of RAM, and the MS Windows XP operating system. The algorithms are coded in C#.net.

6.4.2 Evaluation methods

Two parameters have been taken into account in the evaluation: the number of signers and the number of bits used to generate the common parameters. The schemes are compared including the two major XML multi-signatures: repeated DSA, and Lu's scheme.

- Input bits

The evaluation is categorized to two situations of 160 bits and 256 bits. Which is corresponding to hash algorithm SHA-1 and SHA-256, and also is the length of the parameter q in bits.

- Testing cases

The testing cases are generated using XMark. The selected scaling factor is 1, and created XML data size is 113MB. The XPath is used to assign XML data to be signed. The XML data assigned to each signer is selected randomly from these XPath.

(1) /site
(2) /site/regions
(3) /site/regions/europe
(4) /site/regions/europe/item
(5) /site/regions/europe/item/description
(6) /site/regions/europe/item/description/parlist/listitem
(7) /site/regions/europe/item/description/parlist/listitem/text/keyword

- Sign order graph

In the testing, the signer group has 20 members, and the relationship of their signature generation is shown in Figure 6.3.

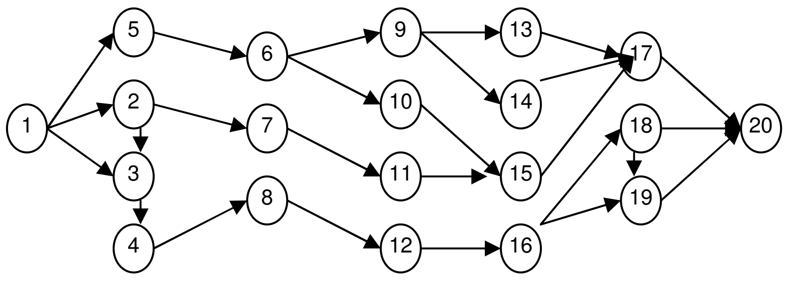


Figure 6.3 Signing order graph

According to approach presented previous, the signing order graph can be converted to a series-parallel signing order as shown in Figure 6.4.

$$\begin{tikzcd}\boxed{1} \arrow[r] & \boxed{2, 5} \arrow[r] & \boxed{3} \arrow[r] & \boxed{4} \arrow[r] & \boxed{6, 7, 8} \arrow[r] & \boxed{9, 10, 11, 12} \arrow[r] & \boxed{13, 14, 15, 16} \arrow[r] & \boxed{17, 18} \arrow[r] & \boxed{19} \arrow[r] & \boxed{20}\end{tikzcd}$$

Figure 6.4 Converted series-parallel signing order

- Output results

The output is the execution time of different multi-signature schemes. The time taken is after XML data assigned to each signer and stopped after the signature generated or validated.

6.4.3 Evaluation results

Figures 6.5 and 6.6 show the execution time overhead corresponding to the signing process, while Figures 6.7 and 6.8 show the execution time overhead corresponding to the verifying process.

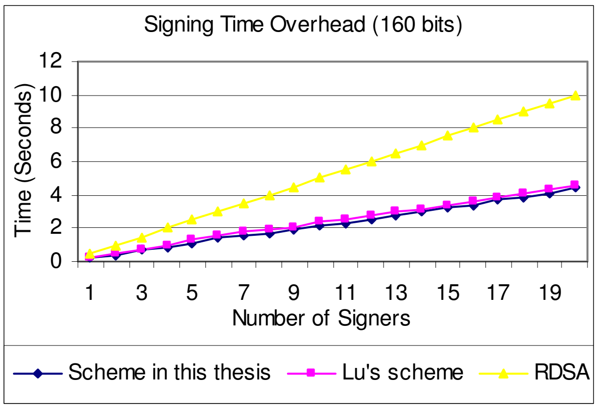


Figure 6.5 Execution time comparison (160 bits signing)

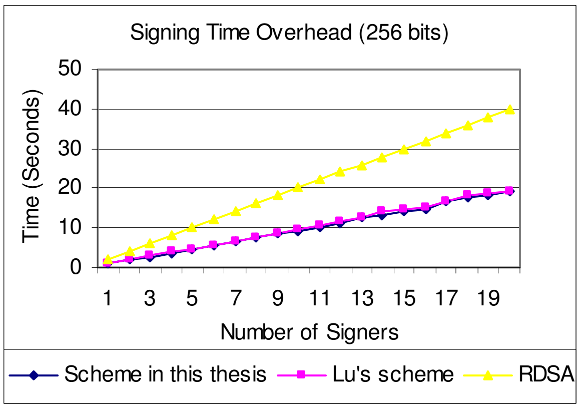


Figure 6.6 Execution time comparison (256 bits signing)

Figure 6.5 and 6.6 show that the superiority of the scheme presented in this thesis and Lu’s scheme over RDSA increasing with the signers size. Although all signers should sign specific XML data, the scheme in this thesis and scheme by Lu have almost 50% higher efficiency. The reason for this result is that these two schemes only sign the XPath expression, not the XML data itself. Compared to sign XML data itself, the XPath expression are significantly smaller. This will decrease the time taken to generate the hash value. Compared to Lu’s scheme, the two have almost the same efficiency; however, the scheme proposed has more functionality and is more practicable in applications.

Figure 6.7 and 6.8 show the superiority of scheme proposed in this thesis and scheme proposed by Lu over RDSA in terms of execution times. The increase of the size group has less impact on schemes both in this thesis and by Lu. When a signature is verified, RDSA should check each signature generated by signers, and this leads to a line of increasing verification time. The schemes presented both in this thesis and by Lu only need to verify the signature generated by SC, so the verification time almost is a constant of about 1.2 seconds.

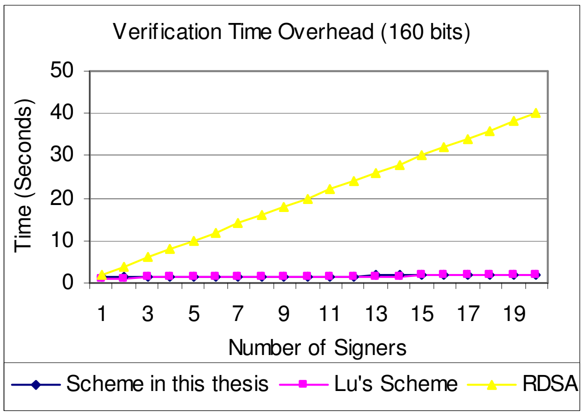


Figure 6.7 Execution time comparison (160 bits verification)

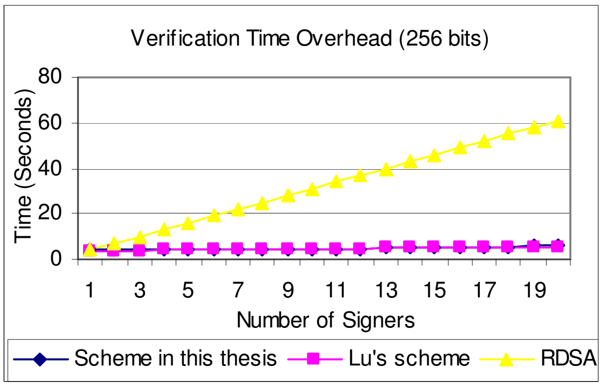


Figure 6.8 Execution time comparison (256 bits verification)

6.5 Discussion and analysis

The three models including RDSA or RRSA, the scheme proposed by Lu, and the scheme proposed in the thesis are compared as listed in Table 6.1.

Table 6.1 Existing schemes comparison

Comparison aspects	RDSA or RRSA	Lu's scheme	Scheme proposed
Integrity validating before signing	The contents which need to be signed	Difficult to validate the contents denoted by XPath	Easy to validate assigned XML data with integrity-checking pool
Signer affection on results	With increasing the numbers of signers, the size of signature results will increase significant.	XML data size increase depends on signing order	Without effect on XML data size when signer numbers increasing
Number of signed objects	An arbitrary number of objects can be signed	An arbitrary number of objects can be signed	An arbitrary number of objects can be signed
Signing order	Broadcast	Broadcast	Series and parallel
Signed contents access	Signed content is directly accessible	Signed content is accessible by XPath	Signed content is accessible by XPath
Signed contents constraints	Arbitrary data can be signed	Arbitrary data can be signed	Arbitrary data can be signed
Binding to signed contents	URI plus transforms	URI plus transforms	URI plus transforms
Numbers of signature value	Depends on signer numbers	1 or more	1

- Integrity validating

Before signing the contents, the signer needs to check the integrity of XML data to be signed. Repeated DSA or RSA only checks the integrity of delegated XML data contents, so the signed results can be copied to another document still with a valid verification results. For Lu's scheme and the scheme presented in this thesis, the integrity checking is not only the XML data itself but also the XPath expression, which denotes the XML data need to be signed. In addition, the presented scheme can ensure that the signed results cannot be copied to another document.

- Signer's number constraint

Although the three schemes have not limited the signer numbers, the scheme of repeated RSA or DSA can increase the size of signature results when signer numbers is increased. It indicates that the repeated RSA or DSA is not suitable to a large signer group.

- The numbers of signed objects

All the three schemes can sign arbitrary numbers of objects.

- Signing order

The scheme of repeated DSA or RSA and the scheme proposed by Lu only support broadcast signature generation. The presented scheme supports a natural signing process, e.g. series and parallel.

- Generated signature value

The numbers of signature value will affect the XML data size. The numbers of signature value depends on the signer numbers in approach of repeated DSA or RSA, because each signer will generate an independent signature value. The number of signature value in approach proposed by Lu depends on signing order graph, and the worst situation is the numbers equal to signer numbers. The best situation is only 1 signature value. In the scheme presented, there only 1 signature value, this value is a mixed multi-signature value.

6.6 Summary

This chapter proposes a series-parallel XML multi-signature scheme. The presented scheme is a mixed order specified XML multi-signature scheme according to a dependent and independent signing process. Using presented XML data integrity-checking pool to provide integrity-checking for decomposed XML data, it makes signing XPath expression practicable, instead of signing XML data itself. The proved evidence shows that the scheme is correct, and the scheme is secure since it is a DL problem. The evaluation results show that the scheme satisfies the functionality of sequential and parallel signing process, and has a higher efficiency than scheme of repeated DSA or RSA. This scheme is also compatible with single XML signatures, sequential or broadcast multi-signature schemes.

Chapter 7 NLBILS based encrypted XML data querying

A number list based interval labeling scheme (NLBILS) for XML data is presented in this chapter. Based on proposed labeling scheme, a structural index for encrypted XML data is illustrated. The testing and evaluation for proposed scheme are also executed.

7.1 Introduction

Using XML encryption technology proposed by W3C, user can encrypt any parts of an XML data. Because of the flexibility of XML encryption, it raises new issue for XML data querying. As shown in Figure 7.1(a), some information of a credit card needs to be encrypted, and the customer names should be viewed by others. Using the encryption methods described by W3C, only information about credit cards can be encrypted, and the result is shown in Figure 7.1(b).

```
<PaymentList>
 <PaymentInfo>
   <Name>Baolong Liu</Name>
   <CreditCardInfo>
      <Number>3209 4465 8972 1205</Number>
      <Issuer>HSBC<Issuer>
      <Expiration>02/11</Expiration>
      <Limit>1000</Limit>
   </CreditCardInfo>
   <Address>Huddersfield</Address>
   <Amount>£120.00</Amount>
 </PaymentInfo>
 <PaymentInfo>
  <Name>Jack Xia</Name>
   <CreditCardInfo>
      <Number>4465 3476 2218 5421</Number>
      <Issuer>Lloyds tsb<Issuer>
      <Expiration>04/12</Expiration>
      <Limit>500</Limit>
   </CreditCardInfo>
   <Address>Manchester</Address>
   <Amount>£210.00</Amount>
 </PaymentInfo>
</PaymentList>
```

(a)

```
<PaymentList>
 <PaymentInfo>
   <Name>Baolong Liu</Name>
   <EncryptedData>
      <CipherData>
    <CipherValue>A23B45C5
    </CipherValue>
      </CipherData>
   </EncryptedData>
   <Address>Huddersfield</Address>
   <Amount>£120.00</Amount>
 </PaymentInfo>
 <PaymentInfo>
   <Name>Jack Xia</Name>
   <EncryptedData>
      <CipherData>
    <CipherValue>C67DR87T
    </CipherValue>
      </CipherData>
   </EncryptedData>
   <Address>Manchester</Address>
   <Amount>£210.00</Amount>
 </PaymentInfo>
</PaymentList>
```

(b)

Figure 7.1 An example for XML encryption

This method of encryption can be used for XML data confidentiality. Consider the following query: //PaymentInfo[//Issuer = "HSBC"]/Name

Only the issuer of credit card can answer the query above, and the credit card information has been encrypted. Without decrypting the contents, this query cannot be executed properly.

This chapter proposes a structural index with considering both efficiency of index information updating and query processing security. A structural index based on number list based interval labeling scheme (NLBILS) is proposed. Proposed structural index provides spare space for node insertion, and makes management of index information more efficiency, so it is easy to update XML data without affecting other nodes. Value index is based on order preserving encryption. With the feature of order preserving, it can support range querying. The index information will be encrypted using different keys. User accesses different parts of index information according to their keys. It will not disclose the structural information and contents to untrusted server. Inspired by XML pool encryption (Geuer-Pollmann, 2004), this thesis proposes a novel approach to protect structural information for encrypted XML data. The encrypted nodes are removed from original XML data, and consist of an encrypted XML data pool. The structural information is protected. When user submits a query Q according to original XML schema, it will be translated to $Q^{'}$ for encrypted XML data with the helping of index information. The server will retrieve the query result and return to user.

7.2 Number list based interval labeling scheme (NLBILS)

7.2.1 Interval-based labeling scheme

The interval-based labeling scheme is described by Li in 2001 (Li and Moon, 2001). In this scheme, each node is assigned two values: start position value and the end position value. The values are positive numbers during the depth first traverse of an XML data as shown in Figure 7.2 (Li and Moon, 2001; Yun and Chung, 2008). The step size of increment is set as 3 in Figure 7.2.

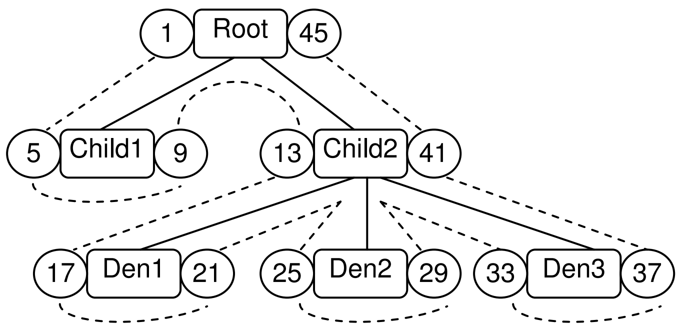


Figure 7.2 Example of interval-based labeling

This technique aims at determining if there exists a relationship ascendance/precedence between two given nodes. A pair $(order(x), size(x))$ is associated to each node x in the document in such a way that, for each child node y of x,

$order(x) < order(y)$ and $(order(y) + size(y)) \leq (order(x) + size(x))$

It has the following property:

$[order(y), order(y) + size(y)] \subset [order(x), order(x) + size(x)]$ if and only if y is the child of x.

When inserting a child to an existing node, it is always possible to find an interval that satisfies that property above. The computation of a new interval for a sibling between two nodes depends on the available remaining space. However, it is difficult to predict the XML data updating, it means that it is difficult to reserve the space which is used to insert XML data. After data updated several times, the space required to contain inserted data will exceed the reserved space, and the re-label of the whole XML data is needed (Yun and Chung, 2008).

7.2.2 NLBILS

This section improves the interval-based labeling scheme focusing on labeling the nodes when there have not enough space for inserting. The basic idea is that if there is not enough space for inserting, the labeling process assigns a number for the sub-tree to be inserted, and then start with a new labeling process for each node in the sub-tree. The labeling result of each node will be consisting of a number list with its parent's nodes label.

Definition 7.1 Number list NL

Let NL be the number list, and $NL = p_1.p_2.\cdots.p_n (n \geq 1)$, where $p_i, i \in (1...n)$ is an integer.

In definition above, if $i = 2$, this means that the situation of not having enough space occurs first time. If $i > 2$, the situation of low inserting space has happened several times, and $p_1.p_2.\cdots.p_n (n \geq 1)$ contains the label of parent node. With the number list, it can overcome the space problem of insertion, and avoid to re-labeling of whole XML data, so improving the efficiency of XML data updating.

Figure 7.3 shows an example of XML data inserting. The sub XML data in rectangle will be inserted into original data tree, and there have not enough inserting space. The sub XML data will be assigned a new number 27, which is in the range of interval-based labeling scheme. Each node in sub XML data will be labelled with a new start number. The labeling result of each node in sub-tree is a number list. For example, the label of node "Enn1" is (27.1, 27.21), "Fcb1" (27.5, 27.9), and "Fcb2" (27.13, 27.17).

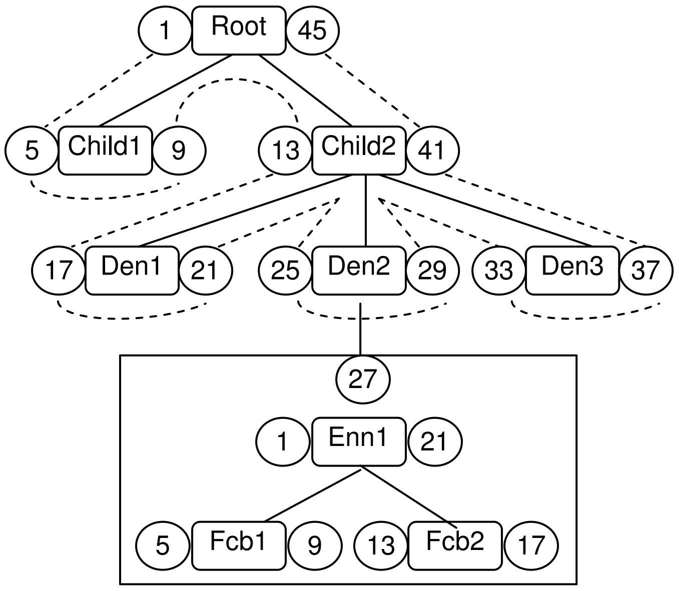


Figure 7.3 Example of XML data inserting

Definition 7.2: Node label The label of each node is denoted as the 3-tuple $(left, right, level)$, where $left$ is the left number list of the node, $right$ is the right number list of the node, and the $level$ is the depth of the node in the XML data.

Lemma 7.1 (Number list relationship). Given two number lists $s = s_1.s_2.\cdots.s_n$, $r = r_1.r_2.\cdots.r_n$, their relationship can be judged by following rules:

- $s > r$, if $s_1 > r_1$, or $(s_i = r_i) and (s_{i+1} > r_{i+1})$, where $i = 2, \ldots, n$
- $s = r$, if $s_i = r_i$, where $i = 1, \ldots, n$
- $s < r$, if $s_1 < r_1$, or $(s_i = r_i) and (s_{i+1} < r_{i+1})$, where $i = 2, \ldots, n$

Lemma 7.2 (Nodes relationships). Given two nodes x, y, let $(x_{left}, x_{right}, x_{level})$ and $(y_{left}, y_{right}, y_{level})$ are the node label respectively;

- $x = y$, if $x_{left} = y_{left}$, $x_{right} = y_{right}$, and $x_{level} = y_{level}$
- x is the parent of y, if $x_{left} < y_{left}$, $x_{right} > y_{right}$, and $x_{level} = y_{level} - 1$
- x is the ancestor of y, if $x_{left} < y_{left}$, and $x_{right} > y_{right}$
- x is the descendant of y, if $x_{left} > y_{left}$, and $x_{right} < y_{right}$
- x is the preceding of y, if $x_{right} < y_{left}$
- x is the following of y, if $x_{left} > y_{right}$

The basic rules for updating are that the lemma 7.2 is still holds. The updating includes the insertion process and deletion process.

1. Insert process

   This process includes two steps: adding a sub-tree into original XML data, and labeling the inserted sub-tree. Two situations should be considered when labeling inserted sub-tree. If the provided space size is bigger than inserting size, the sub-tree to be inserted with integer numbers should be in the range of the space. If the provided space size is smaller than the insert size, it needs to label the data to be inserted as an interval of parent node.

- Space size>Insert size

Under this situation, the sub-tree to be inserted can be labelled in the range of provided space size. Assuming the space size is 100, and n nodes need to be inserted ($n < 100$), and incremental size $S = [100/2n+1]$. Figure 7.4 shows the case. In Figure 7.4 (a), there are 5 nodes to be inserted, and space is 100. Because $2n = 10 < 100$, it means that there has enough space to insert. The incremental size is $S = [100/(2*5+1)] = 9$. The inserted result is shown in Figure 7.4 (b), and it still holds the lemma 7.2.

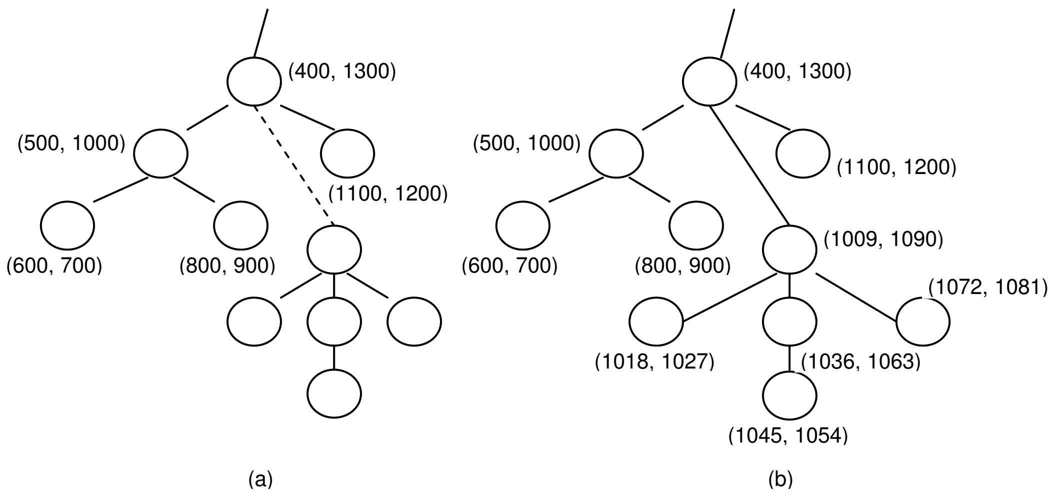


Figure 7.4 Insert processing with enough space

- Space size <=Insert size

The root of the sub-tree to be inserted will be denoted an integer number r, and r is in the range of space. The descendants of r will be labelled with a new data range. Figure 7.5 (a) represents three nodes need to be inserted, and space size is smaller than insert size. The root of sbutree denotes an integer number “1076” in data range. The descendants of root is labelled with a new start as shown in Figure 7.5 (b), and the labelled result of inserted nodes are (1076.100, 1076.600), (1076.200, 1076.300), and (1076.400, 1076.500).

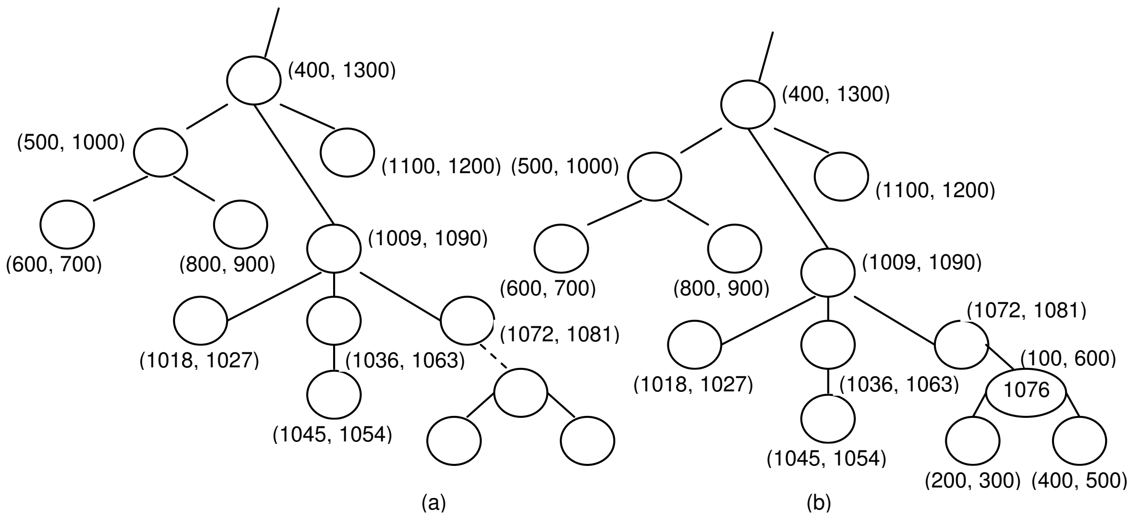


Figure 7.5 Insert processing without enough space

- Space=0

When space size is equal to 0, the sub-tree to be inserted with its parent will be treated as a new sub-tree. Because the parent of the tree to be inserted has obtained position in the XML data tree, the inserting process is similar to situation of space size smaller than insert size. As shown in Figure 7.6 (a), three nodes need to be inserted into data tree, and the insert space is equal to 0. The sub-tree combined with parent node (5,10) as a new sub tree, and it can obtain insert space as shown in Figure 7.6 (b). Although this situation will lead to re-label portions of other nodes, it decreases the affected nodes to the lowest.

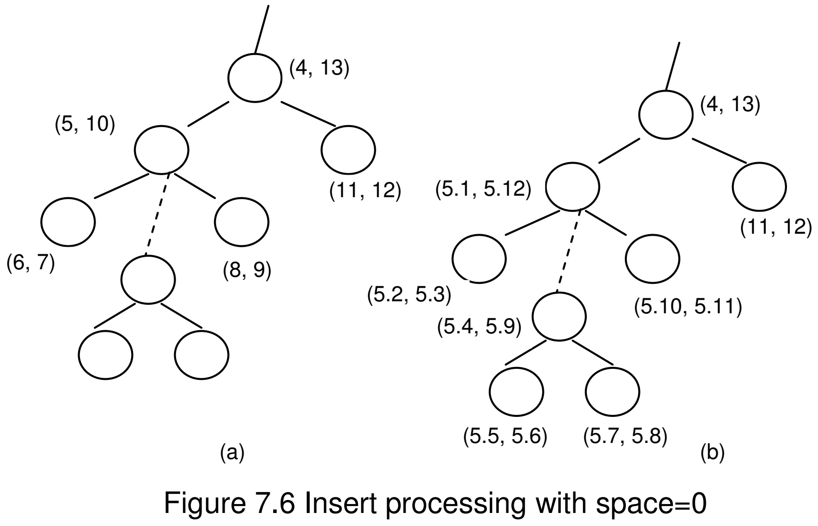


Figure 7.6 Insert processing with space=0

2. Delete process

The XML data deletion can be treated as removing a sub-tree from the original XML data. Because the lemma 7.2 is not broken after deleting a sub-tree, it does not need to do additional performance. As shown in Figure 7.7, a sub-tree and an element will be delete in Figure 7.7 (a), and Figure 7.7 (b) is the deleted result. Figure 7.7 shows that lemma 7.2 will still be hold after deleting process.

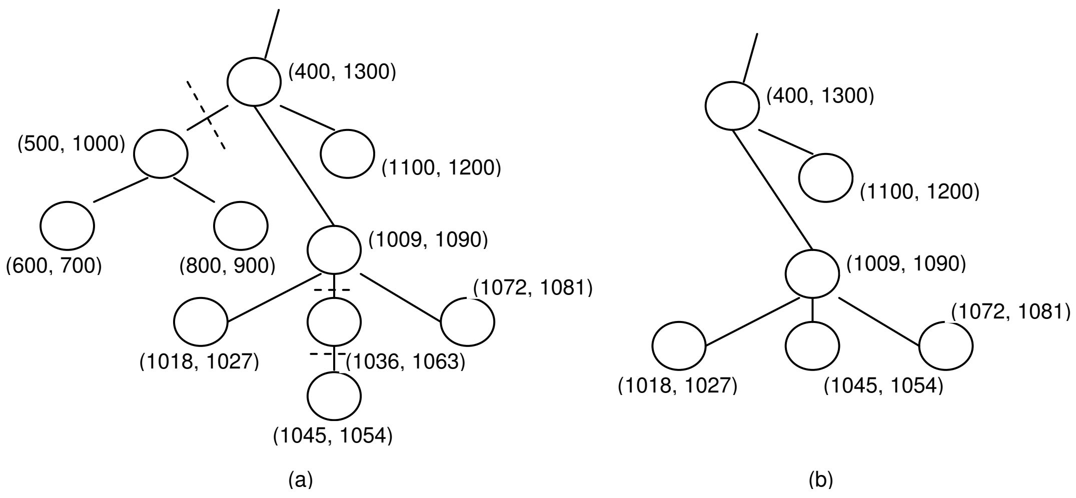


Figure 7.7 Delete processing

7.3 NLBILS based encrypted XML data querying

7.3.1 XML encryption process

Christian Geuer-Pollmann presented the idea of XML encryption pool, which provides a fine-grained XML encryption (Geuer-Pollmann, 2004). The final aim of this method is to encrypt XML data at any granularity. In this research, the encryption process is directly adopted from XML encryption pool. In the encryption process, the selected nodes to be encrypted are encrypted individually under a public key. The encrypted nodes are removed from their original position in the XML data, and collected in a pool of encrypted nodes. Figure 7.8 describes a graphical representation for presented example in Figure 7.1. Figure 7.8 (a) is the original XML data, and Figure 7.8 (b) is the encrypted result, the encrypted data are stored in a pool. This approach can hide the structural information of encrypted nodes. As a result, it can prevent the inference of structural information.

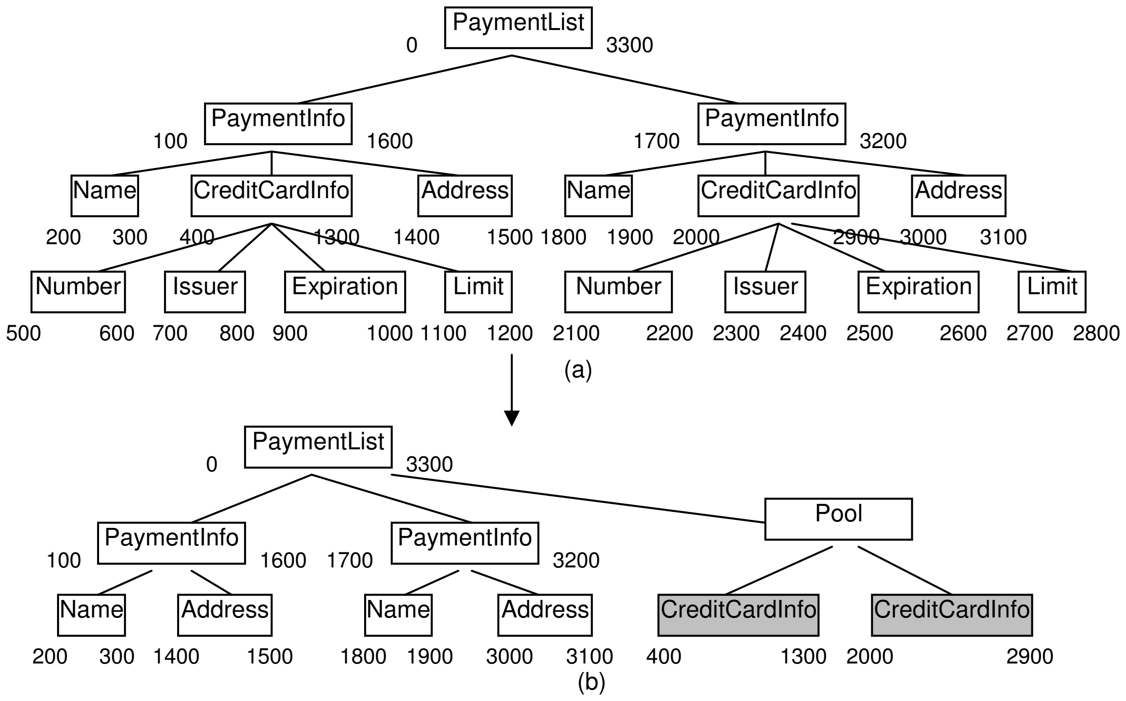


Figure 7.8 A graphical representation for encrypted XML data

7.3.2 Index information

In order to complete the query processing, index information is added on the hosted data at the server side. The index information includes two parts a structural index and a value index.

7.3.2.1 Structural index based on NLBILS

The structural index is set up based on NLBILS that is an effective approach to index tree structured data with considering the efficiency of index information updating. Because the inverted index has been widely used to index XML data (Lee and Whang, 2006), this research adopts inverted index as the structural index for the index information.

Table 7.1 shows the structural index for XML data used in the research. Each entry in the table represents (1) a set of key name, (2) an element name, (3) encoding results, which is the labelled result using NIBILS, and (4) encrypted block id, which indicates the block id of the encrypted XML data.

Table 7.1 Structural index information

Key ID	Element Name	Encoding	Encrypted block ID
k_1	PaymentList	Encoding result of “PaymentList” element	EB1
…	…	…	…

7.3.2.2 Value index

This thesis adopts an order preserving hash function presented by Czech (Czech et al., 1992) to build value index.

$$h(w) = (g(f_1(w)) + g(f_2(w))) \bmod m \tag{7.1}$$

where f_1 and f_2 are functions that map string into integers, and g is a function that maps integers into [0, m-1] within a unique integer (Czech et al., 1992).

So, $h(w)$: integer $\rightarrow$ integer if w is of type integer,

$h(w)$: string $\rightarrow$ integer if w is of type string,

The source code for implementation above hash function can be found at the website http://sourceforge.net/projects/cmph/ (Accessed on October 2010).

7.3.3 Query processing

The architecture for encrypted XML data querying is illustrated in Figure 7.9. A user encrypts an XML data X_D using a public key K_{pub} and encryption function A_e. The encrypted XML data are stored in XML encryption pool. Users can publish the encrypted XML data $A_e(X_D, K_{pub})$ together with the index information. When a query Q needs to be executed on the encrypted XML data $A_e(X_D, K_{pub})$, the user translates Q into an encrypted query Q'. The answer to Q', i.e., $Q'(A_e(X_D, K_{pub}))$, consists of set of encrypted blocks. After received encrypted block, the user decrypts the encrypted block using decryption function d with a private key K_{priv}, and obtain expected results, such that $Q(A_d(Q'(A_e(X_d, K_{pub})), K_{priv})) = Q(X_D)$. This research use XPath, the core of XQuery language for illustrating query processing.

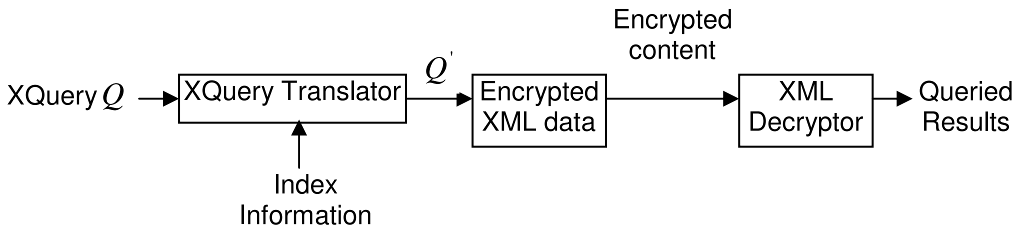


Figure 7.9 The architecture for XQuery on encrypted contents

Based on structural index and value index above, the whole index information is set as in Table 7.2. The greyed portion indicates that contents are encrypted by different keys. Based on this index table, the query processing can be done by five steps.

Table 7.2 Index table for query processing

KeyId	Node Name	Encoding	Value	Encrypted block ID
Null	PaymentList	(0,3300)	null	Null
Null	PaymentInfo	(100,1600)	null	Null
Null	Name	(200,300)	Baolong Liu	Null
k_1, k_2	CreditcardInfo	(400,1300)	null	EB1
	Number	(500,600)	n_1	
	Issuer	(700,800)	i_1	
	Expiration	(900,1000)	e_1	
	Limit	(1100,1200)	l_1	
Null	Address	(1400,1500)	Huddersfield	Null
Null	PaymentInfo	(1700,3200)	null	Null
Null	Name	(1800,1900)	Jack Xia	Null
k_2	CreditcardInfo	(2000,2900)	null	EB2
	Number	(2100,2200)	n_2	
	Issuer	(2300,2400)	i_2	
	Expiration	(2500,2600)	e_2	
	Limit	(2700,2800)	l_2	
Null	Address	(300,3100)	Manchester	

Step 1: User submits a query $< Q, K >$ according to original XML data schema, where, Q denotes an XQuery, and K denotes user private key.

Step 2: System decrypts the corresponding encrypted blocks of index table using user key K. Because the key for encrypted blocks is the same as encrypted content, it can judge the user's accessibility to sensitive information.

Step 3: Structural query translation.

This step can be divided into three sub-steps.

First, client obtains index entries which are associated with each path node in XQuery by checking the index table.

Second, system lists the encoding value associated with path nodes, and prunes away encoding value that do not match structural constraints in the query. This means that the remaining encoding value satisfies the structural constraints of the XQuery.

Third, system replaces each element name with the corresponding encoding value in the structural index table. These encoding values are used to obtain the encrypted block id among encrypted XML data. The results of the structural index processing are the returned encrypted block id.

Step 4: Value-based constraints translation

The value-based constraints can be defined as a triple of $< tag, op, value >$, where $op \in \{<, \leq, >, \geq, =\}$. The $value$ may be plaintext or encrypted contents. If it is a plaintext, which can be found in index table directly, otherwise, generate order preserving hash value by using formula (7.1). The related encoding value is obtained through consulting index table.

Step 5: Final results.

Through previous two steps, the encrypted block which satisfying the XQuery can be determined. In this step, system only needs to return the related encrypted block or plain text obtained from step 3, and step 4.

Example 7.1 Suppose a client holds a key k_2, and submits a query //CreditCardInfo[Issuer=”HSBC”] against the encrypted XML data in Figure 7.8 (b) using the index information in Table 7.2. The query processor first decrypts the index Table 7.2 using the key k_2, and obtains the plaintext of index information, which contains elements “CreditCardInfo”, “Number”, “Issuer”, “Expiration”, and “Limit”. With order preserving hash function, the vale “HSBC” is converted to i_1 . The original query can be translated to //CreditCardInfo[Issuer= i_1]. Through checking the index table, the encrypted block EB1 satisfies this condition, and then obtains the encoding result of element “CreditCardInfo”. The query can be translated to //[400,1300]. The server retrieves the encryption pool and returns the element “CreditCardInfo [400,1300]”. The client decrypts it and obtains the final query results.

7.4 Efficiency analysis for index information updating

Let D is the depth of the XML data, M is the maximal fan-out of the XML data, K is the nodes in each sub-tree, and T is the total nodes in the XML data. The average numbers of re-labelled nodes is N .

$$N = \frac{K}{T} = \frac{\sum_{i=0}^{D}(i+1)\times M^{i}}{\sum_{i=0}^{D} M^{i}} = \frac{(D+1)\times M^{D+1}}{M^{D+1}-1} - \frac{1}{M-1} < \frac{(d+1)\times M^{D+1}}{M^{D+1}-1} \approx D+1 = O(D)$$

If there has enough insertion space without relabeling other nodes, the average numbers of re-labelled nodes is $O(D)$. The worst situation is that the whole XML data needs to be re-labelled, and the numbers of re-labelled nodes is $\sum_{i=0}^{D} M^{i}$. The fact is that XML data with huge numbers of nodes has relatively small numbers of depth (Yun and Chung, 2008), so the structural index updating is efficient.

7.5 Security analysis

As to index based encrypted XML data query scheme, the inference attack is the usually attack method. Inference attack mainly includes two points for XML data: leakage of content of encrypted XML data, leakage of structural relationship between two different nodes, and leakage of structural order between two nodes (Wang and Lakshmanan, 2006). (1) Leakage of structural relationship between two different nodes. By knowing the specific relationship between two nodes which may be either parent-child, ancestor-descendent or sibling-sibling, the attacker infers the nature and type of the sensitive information embedded in a sub data. (2) Leakage of structural order between two nodes x and y . By knowing the specific order of x and y , which is either y is to the left of x or right of x . The attacker infers sensitive information such as a temporal relationship between x and y . With XML encryption pool, these two kinds of structural attack can be avoided. The encrypted XML data has been removed to a pool, it cannot judge the relationship of encrypted XML data, and then the structure information can be protected.

7.6 Testing and Evaluation

7.6.1 The aims of evaluation

The aims of the evaluation focus on three points. The most important point is that the approach can obtain the correct results corresponding to client's XQuery submitted. Through the time cost comparison of index information updating, evaluating the efficiency of proposed approach for index information maintaining. The efficiency of proposed approach for encrypted XML data query processing is compared to existing solutions.

7.6.2 Evaluation methods

The input of the scheme is the query request and the portion encrypted XML document. The output is the encrypted cipher block or an empty result which denoting the data does not contain the relative information queried.

The testing cases deployed to execute evaluation are generated from XMark, and DBLP dataset. For the XMark dataset, various scaling factors (0-1, incremental step is 0.1) were selected to create from 26.5KB to 113MB of documents.

The queries used in experiments for XMark dataset:
(1) /site
(2) /site/regions
(3) /site/regions/europe
(4) /site/regions/europe/item

(5) /site/regions/europe/item/description

(6) /site/regions/europe/item/description/parlist/listitem

(7) /site/regions/europe/item/description/parlist/listitem/text/keyword

Table 7.3 lists the encrypted elements in XML data and the number of querying elements which are encrypted.

Table 7.3 Basic information for testing cases

Total elements in XML data	Number of encrypted elements or block	Number of queried elements which are encrypted
242	10	10
242	20	10
242	30	10
242	40	10
242	50	10
242	60	10
242	70	10
242	80	10
242	90	10
242	100	10

The experiment based on DBLP dataset mainly focuses on range query. Table 7.4 lists the basic information of testing cases based on DBLP dataset. The query 1 and 2 are used to evaluate factor of querying performance based on a very large XML data. The query 3 contains both confidential and non-confidential information. The query 4 and 5 contain highly selective predicates.

(1) /dblp/inproceedings/title

(2) //Thesis/author

(3) //Article [year> ”2002”]/url

//Article [year< ”2006”]/url

//Article [year>= ”2005”]/url

//Article [year<= ”2004”]/url

(4) //inproceedings [booktitle= “DASFAA”]/url

(5) //inproceedings [author=”Elisa Bertino”]/title

Table 7.4 Testing cases for range queries based on DBLP dataset

Total elements in XML data	Number of encrypted elements or block	Number of elements which year >2002	Number of elements which year <2006	Number of elements which year >=2005	Number of elements which year <=2004
321	10	6	7	4	5
321	20	9	14	9	12
321	30	17	23	13	18
321	40	22	31	21	23
321	50	28	37	28	29
321	60	32	43	34	37
321	70	33	50	42	43
321	80	42	54	47	47
321	90	49	61	51	52
321	100	64	67	58	56

7.6.2 Evaluation results

Corresponds to Table 7.3, the testing result is shown in Table 7.5. The number of elements, which has been decrypted, is corresponding to the number of queried elements which were encrypted and containing the query information. The results show that the decrypted blocks or elements only contain information relative to the submitted query. In addition, all the testing executed can achieve correct expected results. Each querying is related to 10 encrypted XML data blocks. With the total increasing encrypted XML data blocks, the query process can obtain expected results. This indicates that the proposed querying scheme can obtain a correct answer responding to XQuery submitted.

Table 7.5 Testing results based on XMark dataset

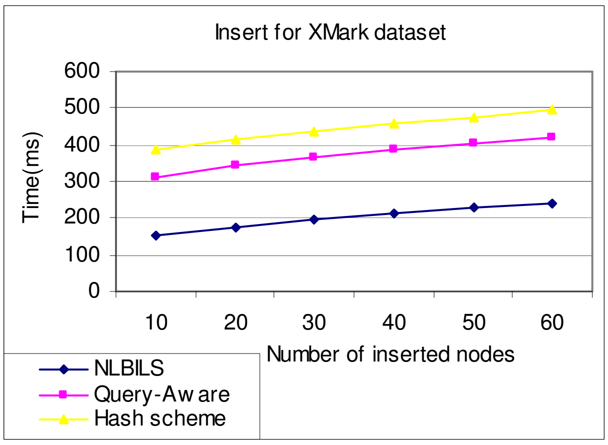
Total elements in XML data	Number of encrypted elements or block	Number of decrypted elements	Average time (Seconds)	Queried results
242	10	10	1.5624	Correct
242	20	10	1.6241	Correct
242	30	10	1.7068	Correct
242	40	10	1.7453	Correct
242	50	10	1.7908	Correct
242	60	10	1.8612	Correct
242	70	10	1.9287	Correct
242	80	10	1.9876	Correct
242	90	10	2.1178	Correct
242	100	10	2.2125	Correct

One of the steps is translating the XQuery submitted to another one, which support querying on encrypted XML data block. The aims of this step are completing a correct translation with index information. With four kinds of range query tested as shown in Table 7.6, the proposed scheme can obtain a correct result relative to range query.

Table 7.6 Results for range queries based on DBLP dataset

Total elements	Encrypted elements	Number of elements which year >2002		Number of elements which year <2006		Number of elements which year >=2005		Number of elements which year <=2004	
		Actual	Queried	Actual	Queried	Actual	Queried	Actual	Queried
321	10	6	6	7	7	4	4	5	5
321	20	9	9	14	14	9	9	12	12
321	30	17	17	23	23	13	13	18	18
321	40	22	22	31	31	21	21	23	23
321	50	28	28	37	37	28	28	29	29
321	60	32	32	43	43	34	34	37	37
321	70	33	33	50	50	42	42	43	43
321	80	42	42	54	54	47	47	47	47
321	90	49	49	61	61	51	51	52	52
321	100	64	64	67	67	58	58	56	56

With the frequency updating of XML data, it will lead to a changing of index information. The advantage of the proposed scheme considers the efficiency of index information updating. In order to evaluate the efficiency on index information updating, this section gives an XML data, which contains 242 elements, based on XMark dataset, and an XML data, which contains 321 elements, based on DBLP dataset. Through inserting the same number of elements (from 10 to 60) as shown in Figure 7.10, the proposed scheme has been compared to the scheme of Query-Aware, and hash scheme approaches (The details of hash scheme and Query-Aware can be found in Chapter 3). The position of XML data to be inserted is generated randomly.



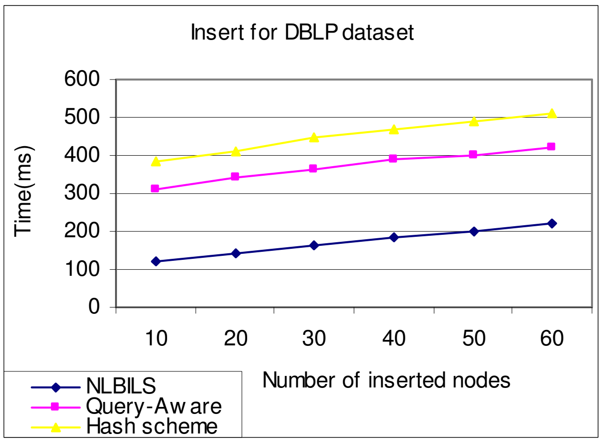
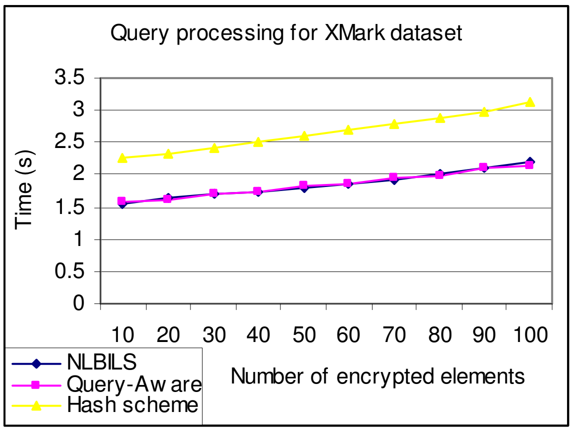


Figure 7.10 Efficiency of index information updating

Based on XMark dataset, the proposed scheme has an average of 199.95ms updating time cost. However, the time cost for Query-Aware and hash scheme are 372ms, and 445.6 respectively. As to DBLP dataset, the average time cost for proposed scheme is 189.65ms. Query-Aware, and hash scheme are 367.4ms and 452.58ms respectively. Although the time cost of index information updating is increasing as the numbers of inserted nodes increasing, the proposed scheme still has almost 48% higher efficiency than Query-Aware, and 57% higher efficiency than hash scheme.

Considering the efficiency of index information updating, especially XML data changing with a high frequency, existing approaches need to re-label the whole XML data to generate encoding values. Based on number list based interval labelling scheme, this problem is solved. This means that the XML data updating cannot lead to re-label the whole index information. Only the elements to be inserted into the original XML data tree will be labelled. Furthermore, hash function based scheme needs to hash each possible XPath when XML data changing, it will cost a huge of time.



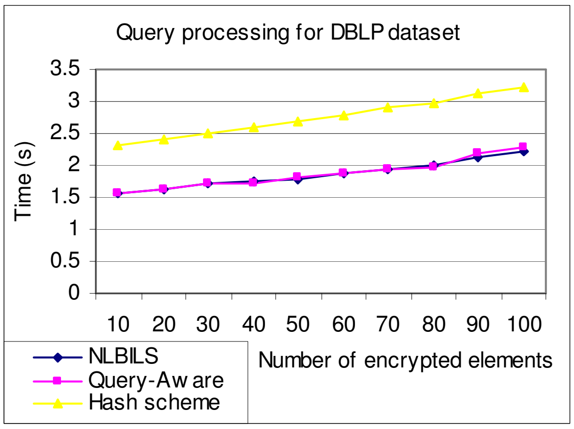


Figure 7.11 Efficiency evaluation for query processing

The efficiency on encrypted XML data query processing has been evaluated as shown in Figure 7.11. The proposed scheme is compared to Query-Aware scheme and hash scheme. The size of tested XML data is 34MB with totally 2232 elements generated from XMark. The evaluated XML data is 97MB with totally 3521 elements from DBLP dataset. The time required variously depends on the numbers of encrypted elements and the size of text node. The proposed scheme and Query-Aware scheme has a 31% higher efficiency than hash scheme as shown in Figure 7.11. The average time cost for proposed scheme and Query-Aware scheme are 1.87s and 1.85s respectively. This slight difference is because the proposed scheme supports range query, and it needs to compute hash values relative to range query. The scheme Query-Aware do not support range query. After a client submitted a query, the hash scheme needs to compute hash value of each path in query sentence, and it is a time cost task. So, the hash scheme has a low efficiency on encrypted XML data query processing, especially when XML data is huge.

7.7 Discussion and analysis

A comparison between existing approaches for encrypted XML data query is made in this section as shown in Table 7.7. The comparison aspects mainly contain index approach, querying process, range querying, and efficiency of index information updating.

- Index approach

  Existing schemes for encrypted XML data query are based on index mechanism. The index approach is main factor affecting the whole process of querying, and also has an important effect on querying efficiency.

- Querying process

  There are different querying processes based on different index approaches. It is embodied on communication process between server and client.

- Range querying

  Range querying is used to obtain group results related to a specific value. Existing relational database and native XML database both support this kind of query. The querying process for encrypted XML data also needs to compatible with them.

- Efficiency of index information updating

  The frequency changing of XML data will lead to index information updating. In order to improve the efficiency of updating index information, it needs to consider mechanism which provides efficiency index information updating, and avoids re-labeling all the XML data.

Table 7.7 Comparison for encrypted XML data query

Solutions	Index approach	Querying process	Range querying	Index information updating
Sem-Crypt (Schrefl et al., 2005)	Maintaining index information both at server side and client side. Adopt hash function GetValueForPath and GetPathInstance to generate index information.	Exact locate at cipher block with index information	Do not support range querying	Re-compute the hash value when XML data updating
Hash scheme (Feng and Jonker, 2003)	Base26Value hash results for DTD and document	Step 1: Generate hash value of XPath Step 2: Index PathInstance table Step 3: Get cipher block id with ValueInstance	Support range querying by using hash function	Re-compute the hash value when XML data updating
Query-Aware Decryption (Lee and Whang, 2006)	Server side index based on Dewey numbers	Step 1: decrypt the index table Step 2: query occurrence to get element type Step 3: get cipher block id	Do not support range querying	Re-label the index information when XML data updating
Efficient secure query (Wang and Lakshman an, 2006)	A discontinuous structural index (DSI)	Step 1: Find the DSI table for tags in the query Step 2: Query interval in DSI and join with cipher block id table to get result	Support range querying with B+ tree	Re-label the index information when XML data updating
XQenc (Yang et al., 2006)	The structure index in XQEnc is based on vectorization and skeleton.	Step 1: Decrypt data block Step 2: decompress the decrypted XML data Step 3: Get result with index information	Do not support range querying	Need to re-compute the vectorization and skeleton compression for index information
Approach in thesis	Number list based interval labeling scheme	Step 1: Translate XQuery Step 2: Analyze the range querying Step 3: Find in the index table Step 4: Get the cipher block	Support range querying with order preserve hash function	Has a high efficiency on index information updating with XML data changing, without re-labeling index information

Based on Table 7.7, the difference between existing approaches is listed.

- Existing querying scheme are based on index information. The index information is maintained at server side or client side. In existing scheme, only Sem-Crypt maintains the index information both at server side and client side, this increases the communication cost between the server and client.

- Only the proposed query scheme considers the efficiency of index information updating. Except for the number list based interval labeling scheme, other schemes have not considered the efficiency of index information updating. When XML data changed, it needs to re-label the whole XML data to generate new index information with low efficiency.

- The proposed scheme supports range querying with a simple order preserver hash function. Most of existing scheme does not support range query. The scheme by Wang adopt B+ tree to support range query, however, it will lead to low efficiency of index information updating when XML data changed (Ünay and Gündem, 2008).

7.8 Summary

This chapter presented the number list based interval labeling scheme for encrypted XML data. The proposed scheme makes maintaining index information more efficient, and it is easy to update XML data with decreasing the number of affected nodes to the lowest. In order to improve the efficiency of index information updating for encrypted XML data query processing, especially when XML data changing frequently, this chapter proposed a structural index based on number list based interval labeling scheme. A novel approach was proposed to protect structural information for encrypted XML data. The basic idea is that encrypted nodes are removed from original XML data, and they consist of an encrypted XML data pool. The structural information is hidden through this method. The testing results show that the proposed scheme can complete a correct query processing, and support range query. The evaluation results show that the proposed scheme supports to maintain index information in an efficient way.

Chapter 8 XML security in calibration certificate management

This chapter describes a case study of XML security in calibration certificate management. The security requirement for calibration certificate management is analyzed. The system architecture is designed based on the requirement analysis. The algorithms relating to XML security are illustrated. The implemented results are also presented in this chapter.

8.1 Introduction

Based on approaches and schemes of previous chapters, this chapter describes a prototype of XML security which allows a programmer to specify the security details of XML data. The prototype is described with a working case study of calibration certificate management. It includes calibration certificate creation, editing, retrieve, and security management in hierarchical environment.

Figure 8.1 is a real calibration certificate expressed in XML format, some details are omitted. The tasks related to calibration certificate management are shown in Figure 8.2.

```
<Certificate>
   <Title>Certificate of calibration</Title>
   <ReferenceNumber>TDFRG</ReferenceNumber>
   <Description>A single-mode Fibre Attention
                        Standard...</Description>
   <Data>This reported expanded uncertainty is based
              on...</Data>
   <Measurements>
     <Description>The measurement of the spectral
                        attenuation...</Description>
     <Table>Designed figure used in measurement</Table>
   </Measurements>
   <Results>
     <Description>The total attenuation...</Description>
     <Graph>Chart related to measurement results</ Graph >
     <Table>Figure of measurement results</Table>
   <Results>
 ⋮
</Certificate>
```

Figure 8.1 A certificate report for fault detection

The system consists of five major tasks: authorization (T_1), certificate retrieve (T_2), certificate editing or creation (T_3), certificate information check (T_4), and certificate information confidentiality (T_5).

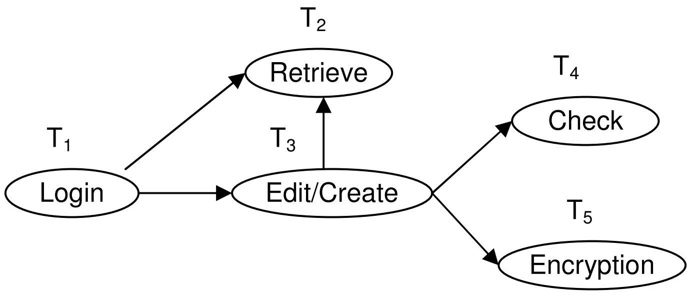


Figure 8.2 The tasks related to certificate management

The system provides the calibration certificate contents and security management in a hierarchical environment. There is more than one user handling a single certificate in workflow system. This process depends on workflow of calibration certificate generation. The prototype will provide the interface for certificate editing, transforming, saving, loading, and searching. XML security enables the secure transmission of information at element level of a document for a certificate. Integrity ensures that the contents of certificate is not being changed, and protect the structural integrity, and context-referential integrity. Authentication is satisfied using digital signature. This functionality should provide digital signature for any portions of a certificate, and further validate the signed certificate. The signature includes a single signature on an XML data, or multi-signature based on work-flow signing process. XML encryption will be used to protect sensitive information of a certificate. This service includes encrypting or decrypting an XML-based calibration certificate. The system can retrieve relative information of a certificate whether it is in cipher block or plaintext.

8.2 System requirements

- User authorization

  When user login the system, the system will judge the authorization with public key information provided. This process is used to decide the privilege of a user.
- Certificate generation and editing

  Certificate information can be created and edited by an authorized user.
- Certificate transforming

  A certificate can be viewed in HTML, XHTML, and PDF format. This requirement needs that an XML-based certificate can be transformed to other format easy to be viewed.
- Certificate integrity

  As a certificate described in XML format, it needs to protect integrity of certificate information, not only considering content integrity, but also protecting structural integrity and context-referential integrity.
- Certificate authentication

  X-certificate is applied to ensure the claimed identity of an entity. In authentication, an entity aims at proving its identity to a verifier, and the creation of a set of calibration certificate data, which is the whole XML-based data or portion of it, is the one claimed. The system allows an authorized user verify the validation of certificate.
- Certificate confidentiality

  Certificate confidentiality ensures that sensitive information of a certificate contents or structures may not be viewed by unauthorized entity. The prototype should provide a mechanism to keep certificate information or portions of information confidential, and the sensitive information can be viewed by specific users.
- Certificate retrieve

  A certificate can be retrieved by a user request. A query processor can identify the contents of encrypted certificate or a certificate in plaintext.

8.3 System architecture

Based on requirements above, the system architecture is shown in Figure 8.3. The architecture consists of six modules.

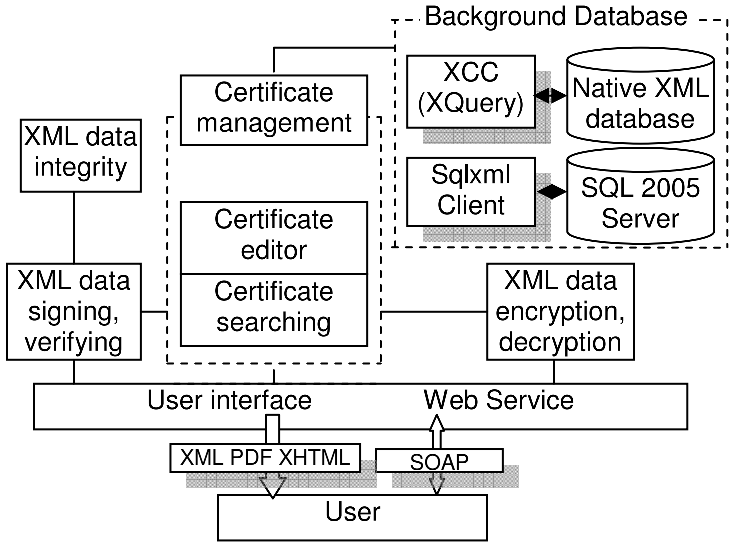


Figure 8.3 System Architecture

- Certificate management module

  Certificate management module is the crucial part of the system. It provides calibration certificate generation, editing, updating, and certificate retrieve service.

- XML data integrity module

  XML data integrity is applied to support XML signature. Before signing a certificate, this module generates hash values for certificate information to be signed. The hash value consists of three parts: content integrity, structural integrity, and context-referential integrity. The three parts combined using a concatenated hash function.

- XML signature and verification module

  XML signature and verification service provide the XML signature based on proposed XML data integrity scheme, and signature verification process. Once the user identity is identified, this service will return signed XML data or verified result for a signed XML data.

- XML encryption and decryption module

  XML encryption and decryption service provide data confidentiality. Once user identity is identified, this service will return encrypted XML data or decrypted XML data.
- Certificate retrieve

  The certificate retrieve is completed by the module of certificate searching. Client can set searching conditions for certificate, and the system returns the certificate or portions of information which satisfy client's request. The retrieve can be done on information of plain text or encrypted block. If the retrieve are relative to encrypted information, client needs to submit a private key at the same time.
- Database

  Because the calibration certificate is expressed in XML format, the system chooses XML native database as background database service. The deployed product is MarkLogic Server 3.1. It supports flexible XQuery over stored XML data.

8.4 Implementation

8.4.1 XML data Integrity

Based on approaches in Chapter 5, CSR based integrity value generation consists of three steps as shown in Figure 8.4. In Figure 8.4 CI denotes the algorithm for content integrity. STI denotes the algorithm for structure integrity, and CRI denotes the algorithm for context-referential integrity.

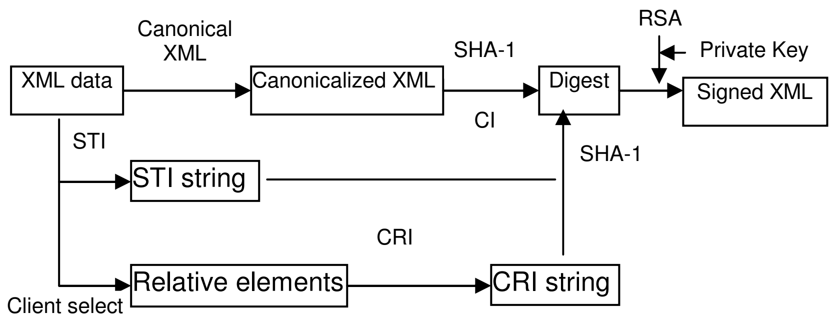


Figure 8.4 Process for CSR generation

Step 1: The securing process selects elements from X_D. S_b is the set containing all the selected elements in this step.

Step 2: The securing process performs algorithm CI, structure integrity, and context-referential integrity related to S_b.

Step 3: The securing process signs CI, STI, and CRI to generate signatures. S_a is the set containing all generated signature in this step, where $S_a = S(ci) \cup S(sti) \cup S(cri) \cup S(t)$, and t is the creation time of XML data.

The integrity verification also consists of three steps as shown in Figure 8.5.

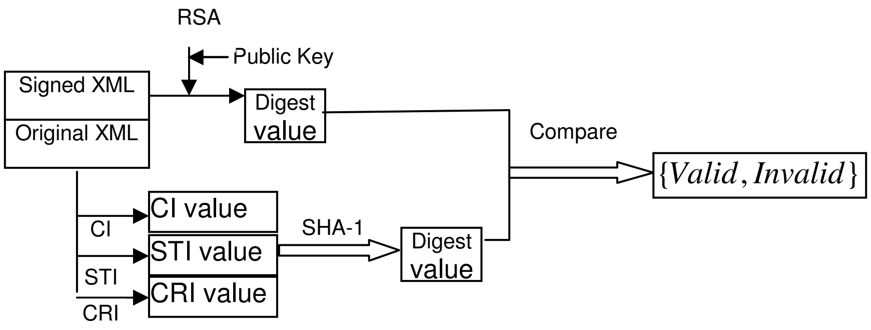


Figure 8.5 Verification process for CSR

Step 1: The securing process obtains hash values from signed results, and user's public key should be provided to the algorithm.

Step 2: System generates hash value of CI, STI, and CRI from original XML data, and then creates the final hash values.

Step 3: The two hash value generated from step1, and step2 are compared, and generate the verification results.

The relative algorithms for content integrity generation and structure integrity generation are listed in Appendix D.

8.4.2 XML data authentication

In this section, XML single signature and XML multi-signature generation are described separately, and focusing on XML signature generation and verification.

- XML single signature

Step 1: A hash value is calculated for each XML data fragment being signed. This involves applying a set of transforms to the XML fragment, calculating the digest on the transformed XML fragment. The transformations ensuring the XML fragment is in a normalized form. This usually is completed using XML canonicalization. The information from this step is represented using a “ds:Reference” element.

Step 2: The “ds:Reference” elements from the previous stage are added to a “ds:SignedInfo” element. A hash value is calculated on the “ds:SignedInfo” element which involves first applying XML canonicalization. This calculated hash value is signed using the signer’s private key to create the “ds:SignatureValue” element. A “ds:KeyInfo” element is used to specify which key was used to create the signature. The “ds:SignedInfo”, “ds:SignatureValue” and “ds:KeyInfo” elements are added to a “ds:Signature” element which is the signature results.

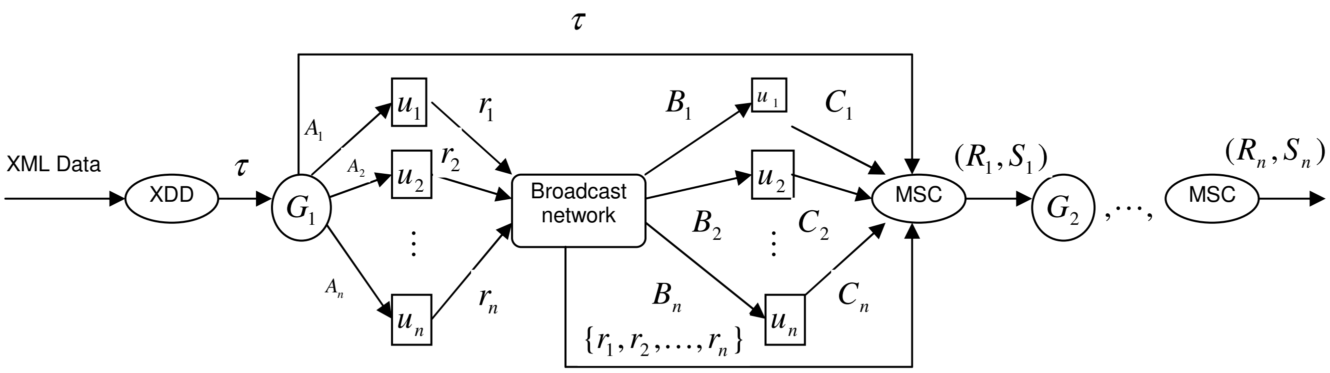
When user intends to verify a signature, the following steps can be executed.

Step 1: A hash value is calculated for each “ds:Reference” element within the signature. This involves applying the transforms specified in the reference, and then calculating the hash value on the transformed XML fragment. The calculated hash value is compared to the one that is within the “ds:Reference” element. When they don’t match, the signature validation fails.

Step 2: A hash value is calculated on the “ds:SignedInfo” element. This involves first applying canonicalization on this element. The hash value of the “ds:SignedInfo” element is retrieved from the signature value using the signer's public key. This hash value is compared with the calculated hash value. When they don't match, the signature validation fails.

- XML multi-signature

XML single signature only satisfies the requirements of one user authenticating an XML data. Based on proposed XML multi-signature scheme in Chapter 6, this section also gives a description on how to implement XML multi-signature. The process is similar to delegated multi-signature scheme proposed by Wu as shown in Figure 8.6.



$$A_i = \{\tau, p_i\}$$
$$B_i = \{r_1, r_2, \ldots, r_n\} \setminus \{r_i\}$$
$$C_i = \{p_i, r_i, s_i\}$$

Figure 8.6 The process for XML multi-signature generation

As shown in Figure 8.6, signers in same group can sign parallel, the different group sign in sequential. This signing model can satisfy multi-signature generation in a mixed signing process. The parameters transferred are identical to the solutions in Chapter 6. The relative algorithms for XML multi-signature is listed in Appendix E.

- Presentation for signed results

The presentation of XML signature view is using XSLT technology. XSL transformation can be performed on an XML data source and generate a result tree. A general application of XSLT is transforming XML data into HTML or

XHTML. The basic steps for transformation are shown in Figure 8.7. First, the signed XML data is validated against the XML signature schema, and then the basic information of each signature in XML data is extracted and delivered to XSLT, also the XPath expressions of the signatures are extracted from the element <Reference>. With each XPath and original signed XML data, the XSLT generates the resulting view of the document.

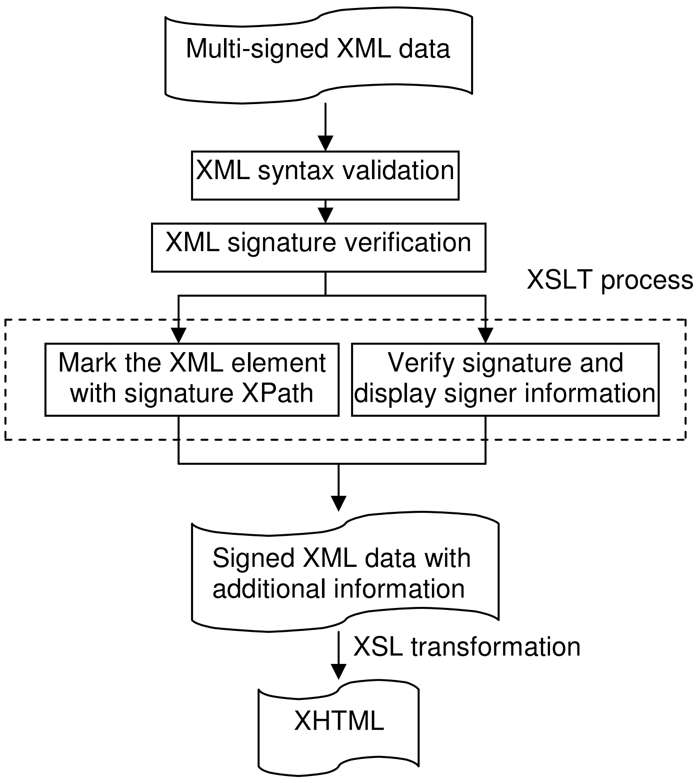

Multi-signed XML data
XML syntax validation
XML signature verification
XSLT process
Mark the XML element with signature XPath
Verify signature and display signer information
Signed XML data with additional information
XSL transformation
XHTML

Figure 8.7 Presentation for XML multi-signature

8.4.3 XML data encryption and decryption process

Encryption can be performed based on different types of data, not just XML data. The XML encryption specification defines how encryption is applied to XML data. It specifies the processes for encrypting and decrypting XML data and the representation of the encryption result in XML (Imamura et. al, 2002).

Data is encrypted using XML encryption by the following steps as shown in Figure 8.8.

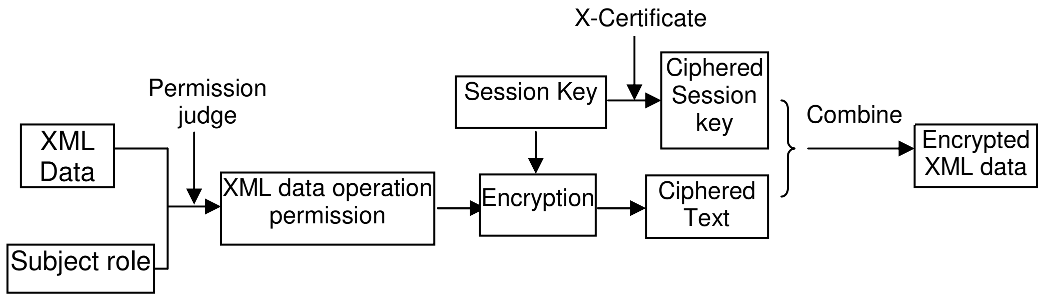


Figure 8.8 Process of role based XML encryption

Step 1: A random session key is generated.

Step 2: The data is encrypted using a symmetric algorithm with the session key. Symmetric encryption is used for the data for better performance. The encrypted data is represented using the “xenc:EncryptedData” element.

Step 3: The session key is encrypted using an asymmetric algorithm with the public key of the receiver. The encrypted session key is represented using the “enc:EncryptedKey” element. The “xenc:EncryptedKey” element can use a “ds:KeyInfo” element to specify which key was used. The encrypted key can be added to the “ds:KeyInfo” element of the “xenc:EncryptedData” element or it can exist independently.

Data is decrypted using XML encryption by the following steps as shown in Figure 8.9:

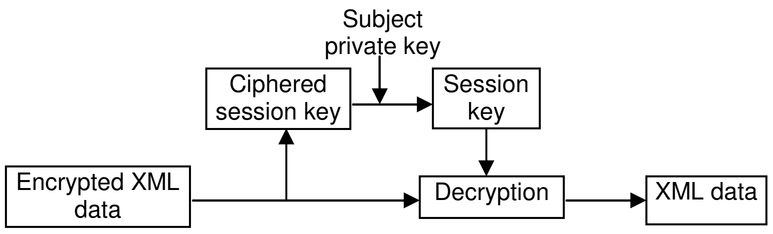


Figure 8.9 Process of XML decryption

Step 1: The encrypted session key within the “xenc:EncryptedKey” element is decrypted using the private key of the receiver. The decrypted session key is the key that was used to encrypt the data.

Step 2: The cipher text within the “xenc:EncryptedData” element is decrypted using the session key.

The process for encrypted XML data pool generation can be divided into three steps.

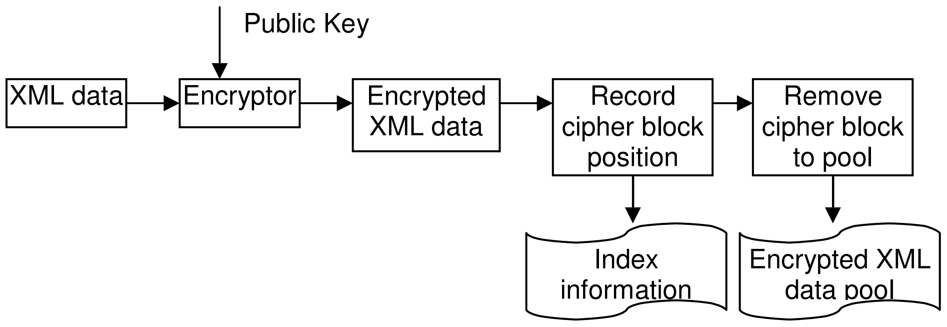


Figure 8.10 Process for encrypted XML data pool generation

Step 1: With XML encryptor, the original XML data can be encrypted as shown in Figure 8.10.

Step 2: Record the cipher block position in encrypted XML data. The position information can be used to generate structural index information.

Step 3: Remove the cipher block into encrypted XML data pool. With the index information, it is easy to find the original position of each cipher block.

8.4.4 Encrypted XML data query processing

The architecture for encrypted XML data query processing was illustrated in Chapter 7. This section only gives the algorithm for index information updating based on NLBILS.

- **Algorithm for index information updating**

```
Procedure InsertSub(SubTree,Pos)
// SubTree is the inserted sub-tree;
// Pos is the (left, right) pair
foreach node n of SubTree do
   Initialize the startList and endList of n to be the startList and endList of the current tree
   SpaceSize=getSpacesize(Pos)
```

```
InsertSize=getSubTreeSize(SubTree)
for i=1 to SpaceSize do
   l[i]=getNewLabel(Pos)
endfor
if SpaceSize>InsertSize then
  Label the nodes in SubTree, attach the start and end value to the startList and
      endList of the nodes     in SubTree
else if 0<SpaceSize<=InsertSize then
  m=l[SpaceSize/2]
  foreach node n in SubTree do
    Attach m to the startList and endList of n
    Label the nodes in SubTree by a new numbering
   Attach the start and end position to the startList and endList of the nodes in
   SubTree
else
      ParentSubTree=subtree rooted by the node that SubTree will be attached to
      foreach node in ParentSubTree do
         Remove the last start and end position from the startList and endList of n
      NewSubTree=ParentSubTree combined with SubTree
      NewSpaceSize=getSpaceSize(position of root of ParentSubTree)
      for i=1 to NewSpaceSize do
         l[i]=getNewLabel(position of root of ParentSubTree)
         k=l[NewSpaceSize/2]
      foreach node n in NewSubTree do
         Attach k to the startList and endList of n
         Label the nodes in NewSubTree by a new number
         Attach the start and end position to the startList and endList of the nodes in
          NewSubTree
   endif
```

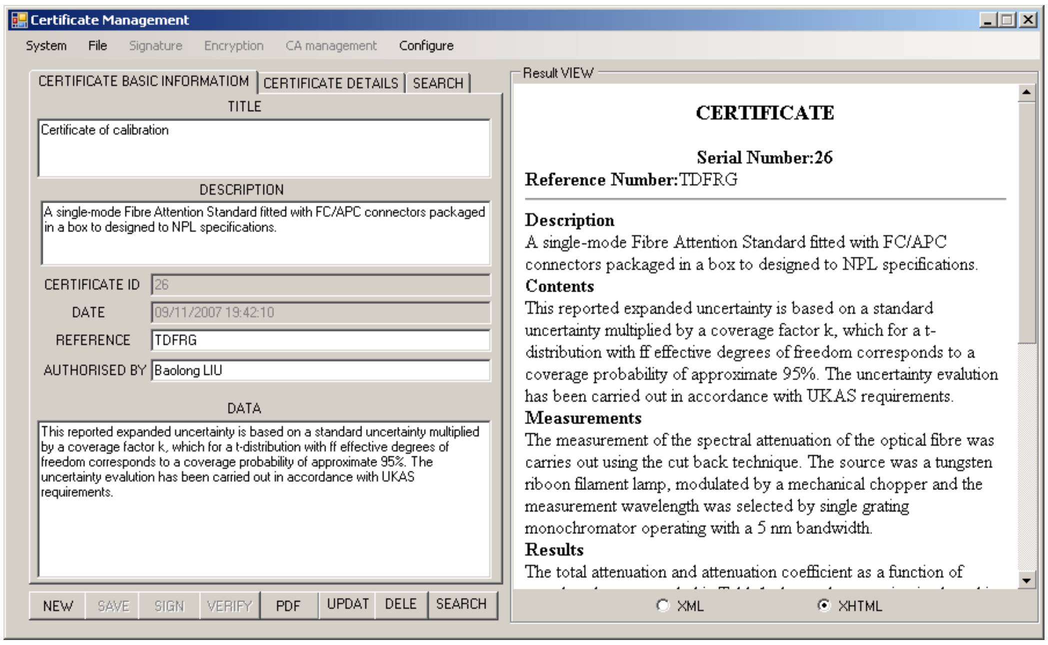
8.5 Implementation results

8.5.1 Environment of development

The prototype was developed on a PC with a 2.39 GHz Pentium (R) 4 processor, 0.99GB of RAM, and the MS Windows XP operating system. The programming language is the C#.net. The background database is deployed as MarkLogic 3.1.

8.5.2 Implementation results

This subsection presents the implementation results according to above system architecture and algorithms. Based on the system architecture, the system interface is shown in Figure 8.11. The functionality of the system includes five modules: certificate editing, user authorization, certificate integrity protecting and authentication, certificate information confidentiality, and certificate retrieve.



TITLE: Certificate of calibration

DESCRIPTION: A single-mode Fibre Attention Standard fitted with FC/APC connectors packaged in a box to designed to NPL specifications.

CERTIFICATE ID: 26

DATE: 09/11/2007 19:42:10

REFERENCE: TDFRG

AUTHORISED BY: Baolong LIU

DATA: This reported expanded uncertainty is based on a standard uncertainty multiplied by a coverage factor k, which for a t-distribution with ff effective degrees of freedom corresponds to a coverage probability of approximate 95%. The uncertainty evalution has been carried out in accordance with UKAS requirements.

CERTIFICATE

Serial Number:26

Reference Number:TDFRG

Description
A single-mode Fibre Attention Standard fitted with FC/APC connectors packaged in a box to designed to NPL specifications.
Contents
This reported expanded uncertainty is based on a standard uncertainty multiplied by a coverage factor k, which for a t-distribution with ff effective degrees of freedom corresponds to a coverage probability of approximate 95%. The uncertainty evalution has been carried out in accordance with UKAS requirements.
Measurements
The measurement of the spectral attenuation of the optical fibre was carries out using the cut back technique. The source was a tungsten riboon filament lamp, modulated by a mechanical chopper and the measurement wavelength was selected by single grating monochromator operating with a 5 nm bandwidth.
Results
The total attenuation and attenuation coefficient as a function of

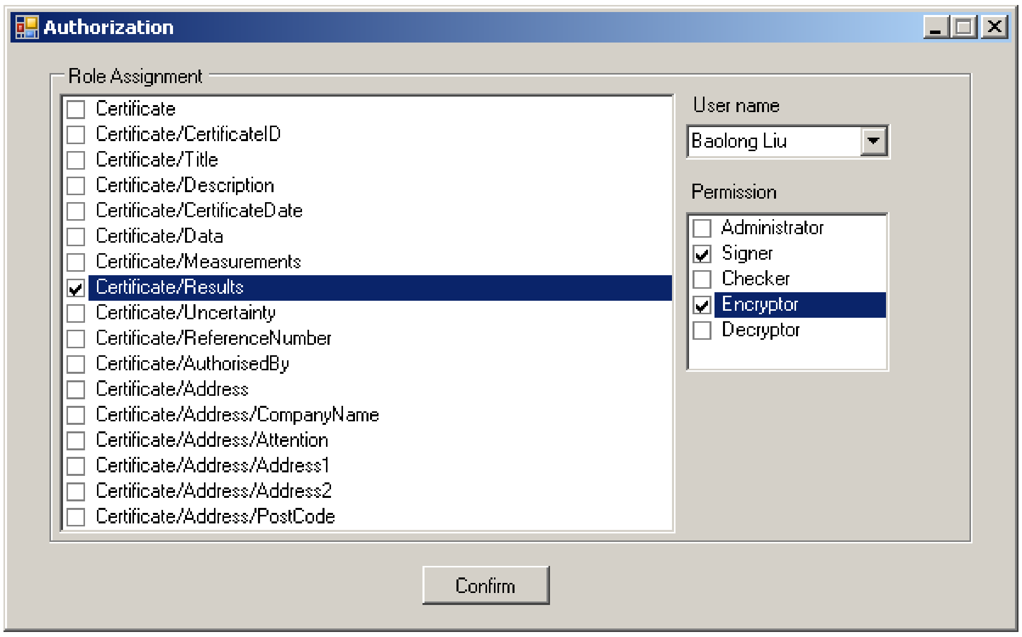
Figure 8.11 System interface

8.5.2.1 Certificate editing

The left side of the main interface in Figure 8.11 provides calibration certificate creation. The basic information for a certificate includes title, description, reference number, issue authority, data information, measurements, results, and so on. After inputting the information, user clicks on button “Save” to save created certificate. With the help of XSLT, the certificate can be viewed in PDF, XHTML format. User can open an existing certificate through menu item “File”, and the opened XML data will be displayed on right side of the interface.

8.5.2.2 User authorization

After a certificate generated, the administrator can assign the role of each user to access the certificate. When a user login as an administrator, the user can open menu item “Management” and click the sub item “Authorization” as the result shown in Figure 8.12.



- [ ] Certificate
- [ ] Certificate/CertificateID
- [ ] Certificate/Title
- [ ] Certificate/Description
- [ ] Certificate/CertificateDate
- [ ] Certificate/Data
- [ ] Certificate/Measurements
- [✓] Certificate/Results
- [ ] Certificate/Uncertainty
- [ ] Certificate/ReferenceNumber
- [ ] Certificate/AuthorisedBy
- [ ] Certificate/Address
- [ ] Certificate/Address/CompanyName
- [ ] Certificate/Address/Attention
- [ ] Certificate/Address/Address1
- [ ] Certificate/Address/Address2
- [ ] Certificate/Address/PostCode

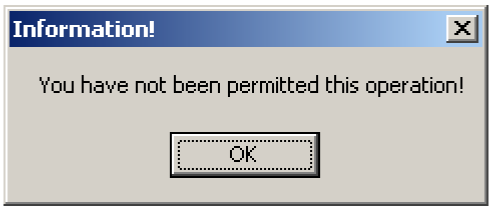
User name: Baolong Liu

Permission:
- [ ] Administrator
- [✓] Signer
- [ ] Checker
- [✓] Encryptor
- [ ] Decryptor

Figure 8.12 Access control authorization

First, the administrator chooses the user name in the list;
Second, administrator selects the certificate information at the left side in Figure 8.12;
Third, the role is assigned to the user with selected privilege.

With three steps above, the system stores the privilege of each user for different certificate information. When users do some operation later, the system checks their privilege first, if the operation forbidden, system will give information as shown in Figure 8.13, otherwise, the operation will be done successfully.

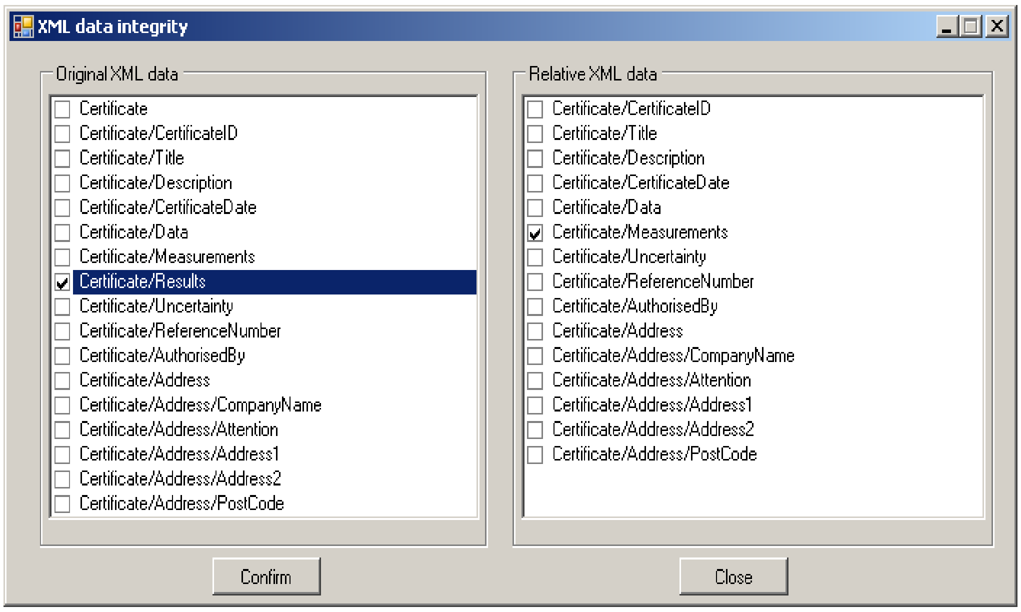


You have not been permitted this operation!

Figure 8.13 Warning information

8.5.2.3 Certificate information integrity and authentication

Certificate authentication is completed based on certificate integrity. This means that before signing a certificate, user should generate the certificate integrity results, and then sign it. When user selects “Signing” under the menu item “Signature”, the system will show the interface as in Figure 8.14.


XML data integrity

Original XML data
- ☐ Certificate
- ☐ Certificate/CertificateID
- ☐ Certificate/Title
- ☐ Certificate/Description
- ☐ Certificate/CertificateDate
- ☐ Certificate/Data
- ☐ Certificate/Measurements
- ☑ Certificate/Results
- ☐ Certificate/Uncertainty
- ☐ Certificate/ReferenceNumber
- ☐ Certificate/AuthorisedBy
- ☐ Certificate/Address
- ☐ Certificate/Address/CompanyName
- ☐ Certificate/Address/Attention
- ☐ Certificate/Address/Address1
- ☐ Certificate/Address/Address2
- ☐ Certificate/Address/PostCode

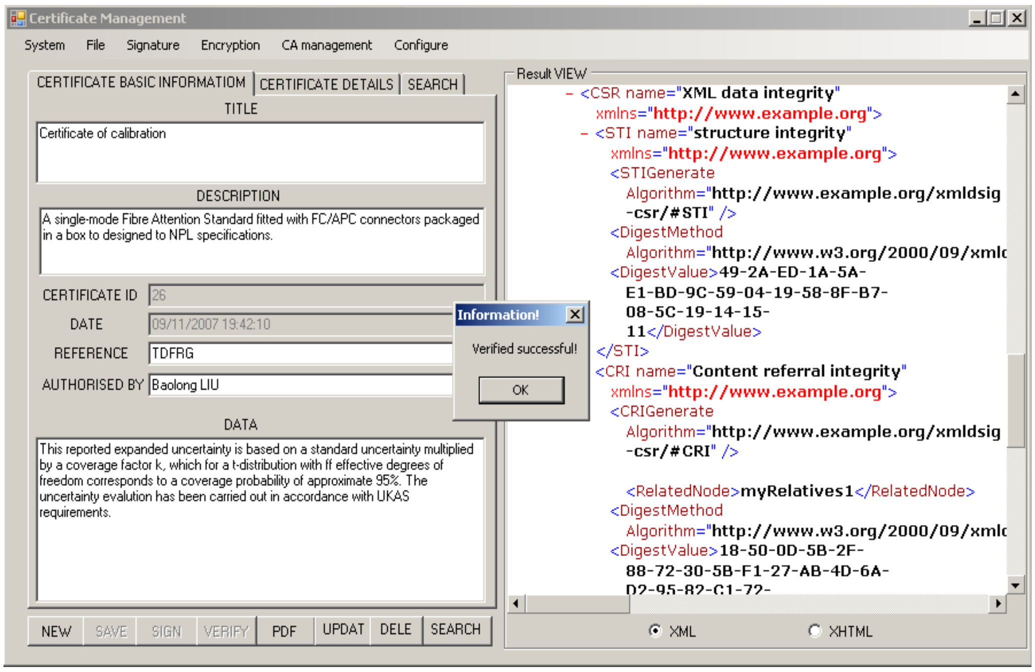
Relative XML data
- ☐ Certificate/CertificateID
- ☐ Certificate/Title
- ☐ Certificate/Description
- ☐ Certificate/CertificateDate
- ☐ Certificate/Data
- ☑ Certificate/Measurements
- ☐ Certificate/Uncertainty
- ☐ Certificate/ReferenceNumber
- ☐ Certificate/AuthorisedBy
- ☐ Certificate/Address
- ☐ Certificate/Address/CompanyName
- ☐ Certificate/Address/Attention
- ☐ Certificate/Address/Address1
- ☐ Certificate/Address/Address2
- ☐ Certificate/Address/PostCode

Figure 8.14 Certificate information integrity

The left side of Figure 8.14 is the XML data to be signed. The context-related elements are shown in right side of Figure 8.14. When user selects an element at left side, right side will display the relative elements automatically according to default records. User can delete or add the new relative XML data in practice. This improves the flexibility of context-referential integrity selection. This process can be summarized into three steps.

First, user needs to select XML data to be signed by selecting possible XPath listed in list-box. Second, user selects context-related XML elements. Finally, through clicking on “confirm” button, system will generate integrity results.

Based on generated integrity results, the system can perform a signing process or verifying process as shown in Figure 8.15. The right side in Figure 8.15 is the signed results based on integrity CSR, and the signed results can be verified by the user. As shown in Figure 8.15, after signed the certificate of calibration, the user can verify it successfully.



TITLE
Certificate of calibration

DESCRIPTION
A single-mode Fibre Attention Standard fitted with FC/APC connectors packaged in a box to designed to NPL specifications.

CERTIFICATE ID 26
DATE 09/11/2007 19:42:10
REFERENCE TDFRG
AUTHORISED BY Baolong LIU

DATA
This reported expanded uncertainty is based on a standard uncertainty multiplied by a coverage factor k, which for a t-distribution with ff effective degrees of freedom corresponds to a coverage probability of approximate 95%. The uncertainty evalution has been carried out in accordance with UKAS requirements.

Verified successful!

```
- <CSR name="XML data integrity"
    xmlns="http://www.example.org">
  - <STI name="structure integrity"
      xmlns="http://www.example.org">
      <STIGenerate
        Algorithm="http://www.example.org/xmldsig
        -csr/#STI" />
      <DigestMethod
        Algorithm="http://www.w3.org/2000/09/xmlc
      <DigestValue>49-2A-ED-1A-5A-
        E1-BD-9C-59-04-19-58-8F-B7-
        08-5C-19-14-15-
        11</DigestValue>
    </STI>
    <CRI name="Content referral integrity"
      xmlns="http://www.example.org">
      <CRIGenerate
        Algorithm="http://www.example.org/xmldsig
        -csr/#CRI" />

        <RelatedNode>myRelatives1</RelatedNode>
      <DigestMethod
        Algorithm="http://www.w3.org/2000/09/xmlc
      <DigestValue>18-50-0D-5B-2F-
        88-72-30-5B-F1-27-AB-4D-6A-
        D2-95-82-C1-72-
```

Figure 8.15 Signed results based on CSR

The following contents depict the detailed components contained in the signature results based on CSR and the details of CSR generation. A completely integrity results and signed results based on CSR can be found in Figure 8.16.

- Structure integrity result

Structure integrity result is generated from formula $ST(v) = h(path(r, v))$, h is a one-way hash function, r is the root node, and v is the node to be signed. In this case, r ="Certificate", and v ="myData" denotes the element of "Measurements".

- XML data content integrity

$CI(v)$ is used to generate hash values of node to be signed, and v ="myData".

- Context-referential integrity

This result is generated by using function $CRI(v) = h(CI(w) \| ST(w))$, where, v is the node to be signed, w is the context-related element. In this case, w ="myRelate" denote the element of "Results", and v ="myData".

- Signature value based on integrity results

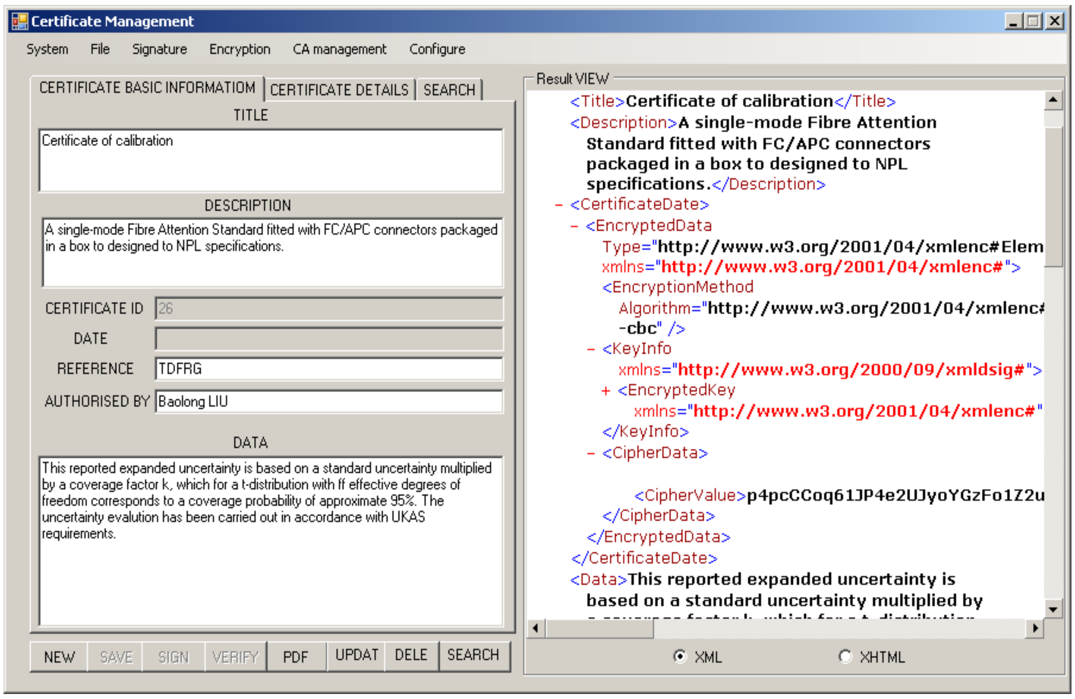
After obtained integrity results, the signature can be created by using function $sign(h(CI(v) \| ST(v) \| CRI(v)), K_{priv})$, where, "||" denotes the concatenation operator.

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
 <SignedInfo>
  <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
  <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
  <Reference URI="#myData">
   <Transforms>
     <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
    </Transforms>
   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
   <DigestValue>bDDbRiQzAsaD8e5K4svNt/6Mhr8=</DigestValue>
  </Reference>
 <CSR name="XML data integrity" xmlns="http://www.example.org">
  <STI name="structure integrity" xmlns="http://www.example.org">
    <STIGenerate Algorithm="http://www.example.org/xmldsig-csr/#STI" />
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
    <DigestValue>49-2A-ED-1A-5A-E1-BD-9C-59-04-19-58-8F-B7-08-5C-19-14-15-11</DigestValue>
  </STI>
  <CRI name="Content referential integrity" xmlns="http://www.example.org">
   <CRIGenerate Algorithm="http://www.example.org/xmldsig-csr/#CRI" />
   <RelatedNode>#myRelate</RelatedNode>
   <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
   <DigestValue>36-C3-C5-A4-02-41-A9-0F-38-B7-C1-7C-7A-A0-A5-DE-7D-3A-75-E9</DigestValue>
 </CRI>
</CSR>
</SignedInfo>
 <SignatureValue>Q2GGAc1bBIf9076W9uXOv3OwwDaAFP/WcO1AArZpGK8QCUoKn6j2ANbdxSX
 BuTqqwK50NjGyRN2VxbI3IxIXLFsHIw5rt/BoK7gkiGOXQTiwQV9AXK109dsfaqIvuesjZx2zHY0+8T
 QOKaJBXOsa9zjjbuHSxRyJLTnaLRstdnA=</SignatureValue>
 <Object Id="myData" />
 </Signature>
```

Figure 8.16 Integrity CSR and signed results

8.5.2.4 Certificate information confidentiality

When sensitive information needs to be encrypted, user can click on menu item "Encryption". The system allows user to load a public key, and then selects the nodes to be encrypted. The encryption result is shown in Figure 8.17. The original XML data is replaced by the element "EncryptedData". The element "EncryptedData" contains the element "EncryptedKey", and the "CipherData". The element "EncryptedKey" is the encrypted session key using algorithm RSA with public key. The element "CipherData" is the encrypted XML data using session key with algorithm AES-256. In Figure 8.17, the encrypted element is "CertificateDate". The original XML data element "CertificateDate" can be viewed by process of decryption and relative private key. Through clicking the "Decrypt" menu item in "Encryption", user can obtain the original XML data.



TITLE	Certificate of calibration
DESCRIPTION	A single-mode Fibre Attention Standard fitted with FC/APC connectors packaged in a box to designed to NPL specifications.
CERTIFICATE ID	26
DATE	
REFERENCE	TDFRG
AUTHORISED BY	Baolong LIU
DATA	This reported expanded uncertainty is based on a standard uncertainty multiplied by a coverage factor k, which for a t-distribution with ff effective degrees of freedom corresponds to a coverage probability of approximate 95%. The uncertainty evalution has been carried out in accordance with UKAS requirements.

(• XML) XHTML

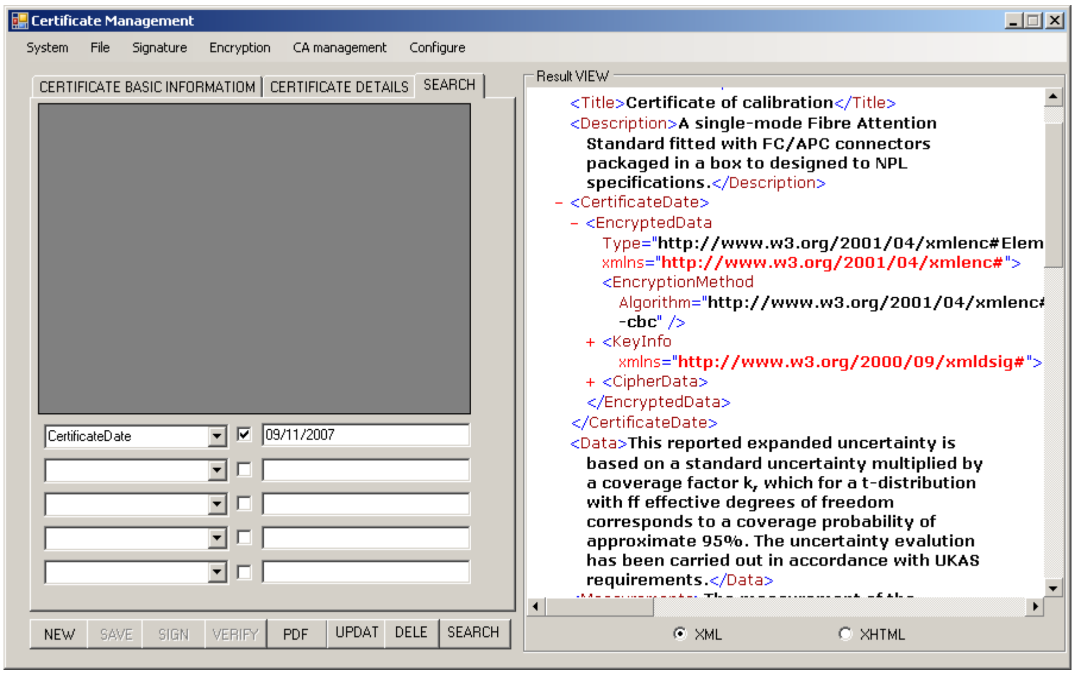
```
  <Title>Certificate of calibration</Title>
  <Description>A single-mode Fibre Attention
    Standard fitted with FC/APC connectors
    packaged in a box to designed to NPL
    specifications.</Description>
- <CertificateDate>
  - <EncryptedData
      Type="http://www.w3.org/2001/04/xmlenc#Elem
      xmlns="http://www.w3.org/2001/04/xmlenc#">
      <EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#
        -cbc" />
    - <KeyInfo
        xmlns="http://www.w3.org/2000/09/xmldsig#">
      + <EncryptedKey
          xmlns="http://www.w3.org/2001/04/xmlenc#"
      </KeyInfo>
    - <CipherData>

          <CipherValue>p4pcCCoq61JP4e2UJyoYGzFo1Z2u
      </CipherData>
    </EncryptedData>
  </CertificateDate>
  <Data>This reported expanded uncertainty is
    based on a standard uncertainty multiplied by
```

Figure 8.17 Certificate information confidentiality

8.5.2.5 Certificate retrieve

The system also provides functionality of certificate retrieving as shown in Figure 8.18. The left side is used to input the query condition, and the right side is used to display queried results.



CertificateDate [✓] 09/11/2007

```
  <Title>Certificate of calibration</Title>
  <Description>A single-mode Fibre Attention
    Standard fitted with FC/APC connectors
    packaged in a box to designed to NPL
    specifications.</Description>
- <CertificateDate>
  - <EncryptedData
      Type="http://www.w3.org/2001/04/xmlenc#Elem
      xmlns="http://www.w3.org/2001/04/xmlenc#">
      <EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#
        -cbc" />
    + <KeyInfo
        xmlns="http://www.w3.org/2000/09/xmldsig#">
    + <CipherData>
    </EncryptedData>
  </CertificateDate>
  <Data>This reported expanded uncertainty is
    based on a standard uncertainty multiplied by
    a coverage factor k, which for a t-distribution
    with ff effective degrees of freedom
    corresponds to a coverage probability of
    approximate 95%. The uncertainty evalution
    has been carried out in accordance with UKAS
    requirements.</Data>
```

[✓ XML] XHTML

Figure 8.18 Certificate retrieve

The querying process can be executed on plaintext or encrypted XML data according to the scheme described in Chapter 7. When the user inputs the condition of query, the system will search the certificate which is stored in database MarkLogic. As shown in Figure 8.18, the querying condition is the “CertificateDate”, and the queried results are displayed. In displayed case, the element “CertificateDate” is in cipher block.

8.6 Discussion and analysis

The relationship of XML data integrity, authentication, and confidentiality is an important factor affecting the generation of each result. Generally speaking, XML data integrity is the basis of XML digital signature. XML signature signs the hash value of XML data instead of XML data itself, and hash value is used to check the integrity of XML data. The sequence for XML signature and XML encryption generation is various. However, different sequence could generate totally different results. This section discusses the relationship of XML data integrity, authentication, and confidentiality.

8.6.1 The basis of XML signature

XML digital signature is applied to ensure XML data authentication. Strictly speaking, XML signature supports to protect XML data integrity, as well as ensuring XML data authentication. Figure 8.19 shows the integrity position in XML signature.

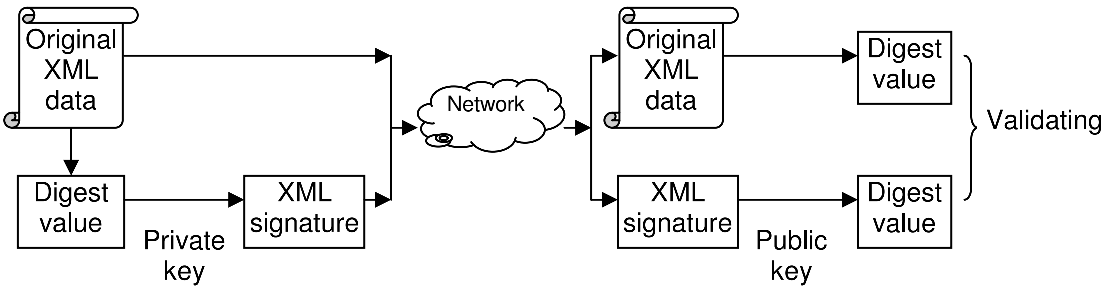


Figure 8.19 XML signature validating data integrity

The original XML data and the signature are transferred to the recipient. The hash value generated by one-way hash function is used to ensure XML data integrity, and it is encrypted with the signer's private key. The recipient first uses the signer's public key to decrypt the hash result, and uses the same hashing algorithm to generate a new hash value of the same XML data. Through comparing the new hash result against the original hash value, the integrity is ensured.

Compared to traditional data integrity, the XML data integrity model proposed in Chapter 5 has advantage of preventing XML signature tampering. Without the structure integrity and context-referral integrity, it is easy to copy a signature into another XML data and still keeping the valid signature.

8.6.2 The sequence of XML signature and XML encryption

Anyone can sign or encrypt portions of an XML data at any order, which mainly are encrypted-then-signed, and signed-then-encrypted. The signing or encrypting sequence will generate completely different results. The principle for XML signature presented by W3C is the practicable rules for XML signature application (Bartel et al., 2008). It has presented approach how to handle different sequence relating to XML signature and XML encryption as follows.

- Principle 1: Only what is "Seen" should be Signed

XML signature signs any information indicated by a transform: *"only what is "seen" should be signed"*. It is necessary to secure as exactly as practical the information that was presented to the user (Bartel et al., 2008). Note that this can be accomplished by literally signing what was presented, such as the screen images, auditory or other media. However, this may result in data which is difficult for security software to manipulate. Under this situation, one can sign the data along with whatever filters, style sheets or other information that generates its presentation.

- Principle 2: "See" What is Signed

"*Persons and automated mechanism that trust the validity of a transformed document on the basis of a valid signature should operate over the data that was transformed (including canonicalization) and signed, not the original pre-transformed data. This recommendation applies to transforms specified within the signature as well as those included as part of the document itself*" (Bartel et al., 2008).

8.6.2.1 Encrypted-then-signed

No one should be asked to sign a data that they cannot see, and this situation opposite the basic principles of "Only What is "Seen" should be Signed" (Hughes et al., 2002). When a data is encrypted, a user cannot infer the information through the cipher text. The encrypted-then-signed is meaninglessness in applications.

8.6.2.2 Signed-then-encrypted

If one intends to sign the plain text which is later encrypted, the person can use the transform specified by the W3C (Hughes et al., 2002). It has been noted by David Solo that both XML encryption and XML signature can be performed on an XML data in any order and any time (Hughes et al., 2002). An example has been described by W3C as follows, when a user wishes to order and pay for a product from a company using the trusted payment system Paypal. The company creates an order form including the product name, quantity, price, and account information. The company signs all of these information (Hughes et al., 2002), and encrypts the account information for Paypal only. The company sends the order form to the user for confirmation with user's signature. To validate both signatures, Paypal will have to know the encrypted information for validating the company's signature.

However, encryption applied to the signed content may result a signature not to be verifiable, and it needs to decrypt the encrypted XML data before the signature is verified (Hughes et al., 2002). It needs a mechanism to decrypt only signed-then-encrypted portions. There are two cases: one is that the encryption and signature order can be derived directly from the content. The other is that encrypted content is the signed resources, and it is difficult to derive it directly from the content, which defined as order issue within signed resources. W3C has proposed the specification of "Decryption Transform for XML Signature" to handle these two kinds of situation.

8.7 Summary

A case study of XML security in calibration certificate management is designed and implemented conforming to the approaches and schemes in previous chapters. The results of the tests and analysis show that the prototype can benefit the security management of calibration certificate.

Chapter 9 Conclusions and future works

This chapter summarizes the outcomes of this research and highlights the contributions in the relevant research topics, which were described in previous chapters. Future works relative to XML security are also discussed.

9.1 Contributions and conclusions

This dissertation aims at improving XML security relative technologies, and makes it more practicable and secure. The proposed works have demonstrated the feasibility and applicability of presented approaches and schemes with systemic validation over the performances of the solutions. It is perceived that the dissertation has made several contributions to the domain knowledge.

9.1.1 Revocation information validation for x.509 digital certificate

The first main contribution of this dissertation is that a novel approach for revocation information validation for X.509 digital certificate was proposed. In order to alleviate the burden of XKMS for certificate revocation query, the thesis proposed a novel idea to make certificate revocation handling and validation easier using XML signature technology. Certificate owner's signature is applied to provide evidence for revocation information of the certificate. It does not need to query XKMS or CA for revocation information of such certificate, because the certificate already contains the status information. It improves the efficiency on revocation information checking, further alleviates the burden of XMKS server.

9.1.2 XML data integrity

The second main contribution of this dissertation is that an overall XML data integrity requirements was presented combining XML data features, and then satisfies the requirements with an integrity model for XML data with a high efficiency.

- XML data integrity requirements combining XML data features were presented under fine-grained XML security. Three aspects are considered including content integrity, structure integrity, and context-referential integrity.

- Based on proposed requirements, an integrity approach CSR for XML data was set up based on the concatenated hash function.

- Based on the concatenated hash function to generate hash values for XML data, the approach has a higher efficiency than the Merkle hash function-based hash value-generation process.

9.1.3 Series-parallel XML multi-signature scheme

The third main contribution of this research is that an XML multi-signature scheme was proposed to satisfy a dependent and independent signing process. To the domain knowledge, this is the first XML multi-signature scheme supporting series and parallel signing process.

- An XML data integrity-checking pool to provide integrity-checking for decomposed XML data was presented. XML data integrity-checking pool makes signing XPath expression practicable for XML data.

- A series-parallel XML multi-signature scheme according to a mixed dependent and independent signing process was proposed based on series-parallel signing group and XML data integrity-checking pool.

9.1.4 Efficient index information updating for encrypted XML data

The fourth main contribution of this dissertation is that a structural index for encrypted XML data with considering both efficiency of index information updating and query processing security was proposed.

- The number list based interval labeling scheme for encrypted XML data was presented. The proposed scheme is easy to update XML data with decreasing the number of affected nodes to the lowest.

- In order to improve the efficiency of index information maintaining for encrypted XML data query processing, especially when XML data changing frequently, the thesis proposed a structural index based on the number list based interval labeling scheme.

- A novel approach was proposed to protect structural information for encrypted XML data. The basic idea is that encrypted nodes are removed from original XML data, and they consist of an encrypted XML data pool. The structural information is hided through this method.

9.2 Future works

The major disadvantage of proposed integrity approach is that user needs to select the context-related elements in the process of signature creation. The disadvantage increases complexity of interaction between the user and the system. One of the future works will focus on integrating XML data integrity constraints into presented solution to capture context-related elements automatically.

The implemented prototype only is a demonstration of the proposed solutions to solve security issues in calibration certificate management. It needs common XML security tools which easy to be integrated into existing applications. Another future works is to focus on integrating XML security into native XML database to solve the security issues existing in native XML database system, and further developing XML security common tools.

9.2.1 Context-related elements selection

The problem of selecting the context-related elements within an XML data was discussed in Chapter 5. As mentioned, with the development of integrity constraints for XML, it is possible to integrate the constraints for XML into context-related elements selection. Integrity constraints for XML are defined to limit the relationship among XML elements. Existing types of integrity constraints include the XML key constraints, referential constraints, and XML semantic constraints. These constraints is used to protect the integrity when XML data updating or storage. In the future, these constraints will be introduced into XML data integrity for context-related elements selection.

9.2.2 Integrate XML security into native XML database system

Native XML database system has been built for several years, such as Marklogic, dbXML, Xindice, eXist. These systems just provide a mechanism for XML data storage and query, the security issues relative to XML data have not been considered. When an XML data is encrypted, how to execute a query on these data is not taken into account. The work related to native XML database security will be carried out.

- Access control model for portions of XML data in native XML database.
  The major advantage of XML is that it provides a fine-grained access. Although native XML database system provides access control mechanism, the access control rules only can be defined on entire XML data. It has not considered the access control for portions of XML data. This indicates that the current access control mechanism has not taken XML data feature of fine-grained accessibility into account.

- Development a mechanism for encrypted XML data query processing in native XML database
  When a user encrypts portions of XML data for security problem, the query processor of native XML database cannot deal with it. Another future work is that deploy the proposed encrypted XML data query processing into native XML database. In other words, a query processor for encrypted XML data will be developed.

References

Adams, C., Cain, P., Pinkas, D., Zuccherato, R., 2001. “Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)”, (RFC 3161), IETF.

Aiello, W., Lodha, S., Ostrovsky, R., 1998. “Fast digital identity revocation (extended abstract)”, In :CRYPTO, pp. 137-152.

Agrawal, R., Kiernan, J., Srikant, R., Xu, Y., 2004. Order preserving encryption for numeric data. In: SIGMOD Conference. (2004) pp 563-574.

Altinel, M., Franklin, M., 2000. Efficient filtering of XML documents for selective dissemination of information. In proceeding of the 26th international conference on Very Large Data Bases, Cairo, Egypt, 2000, pp 53-64.

Arnes, A., 2000. “Public key certificate revocation schemes”, Master thesis, Queen’s University, Kingston, Ontario, Canada.

Arsenault, A., Turner, S., 1999. PKIX Roadmap. IETF Internet Draft.

Barker, W. C., 2004. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher. NIST special publication 800-67m version 1.1, available at: http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf (Accessed on October 2010).

Bartel, M., Boyer, J., Fox, B., LaMacchia, B., Simon, E., 2002. XML-Signature Syntax and Processing, W3C Recommendation 12 February 2002, available at: http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/ (Accessed on October 2010).

Bartel, M., Boyer, J., Fox, B., LaMacchia, B., Simon, E., 2008. XML signature syntax and processing (second edition), Available at: http://www.w3.org/TR/xmldsig-core/ (Accessed on 6 December 2008)

Benjumea, V., Choi, S. G., Lipez, J., Yung, M., 2007. “Anonymity 2.0 – X.509 extensions supporting privacy-friendly authentication”, CANS 2007, LNCS 4856, pp. 265-281.

Berglund, A., Boag, S., Chamberlin, D., Fernández, Mary F., Kay, M., Robie, J., Siméon, J., 2007. XML path language (XPath) 2.0. Available at: http://www.w3.org/TR/xpath20/ (Accessed on December 2008).

Bertino, E., 2001. XML security. Information security technical report, Vol. 6 No. 2, pp 44-58.

Bertino, E., Carminati, B., Ferrari, E., Thuraisingham, B., Gupta, A., 2004. Selective and authentic third-party distribution of XML documents. Knowledge and Data Engineering, IEEE Transactions, 16(10) pp 1263–1278.

Blobel, B., 2004. Authorisation and access control for electronic health record systems. International Journal of Medical Informatics, 73(3) pp 251-257.

Boag, S., Chamberlin, D., Fernandez, M., Florescu, D., Robie, J., Simeon, J., 2007. XQuery 1.0: An XML Query Language. Available at: http://www.w3.org/TR/xquery/ (Accessed on December 2009).

Boritz, J.E., No, W.G., 2005. Security in XML-based financial reporting services on the Internet. Journal of Accounting and Public Policy, 24(1), pp 11-35.

Boyd, C., 1991. Multisignatures based on zero knowledge schemes. Electronics letters 27, pp 2002-2004.

Boyer, J., 2001. Canonical XML Version 1.0, W3C Recommendation 15 March 2001, available at: http://www.w3.org/TR/xml-c14n (Accessed on October 2010).

Boyer, J., Eastlake, D. E., Reagle, J., 2002a. Exclusive XML Canonicalization Version 1.0, W3C Recommendation 18 July 2002, available at: http://www.w3.org/TR/xml-exc-c14n/ (Accessed on October 2010).

Boyer, J., Hughes, M., Reagle, J., 2002b. XML-Signature XPath Filter 2.0, W3C Recommendation 08 November 2002, available at: http://www.w3.org/TR/xmldsig-filter2/ (Accessed on October 2010).

Boyer, J., Marcy, G., 2007. Canonical XML 1.1, W3C Candidate Recommendation 21 June 2007, available at: http://www.w3.org/TR/2007/CR-xml-c14n11-20070621/ (Accessed on July 2009).

Brandt, P., Bonte, F., 2000. Towards secure XML. Availabel at: http://lists.w3.org/Archives/Public/xml-encryption/2000Oct/att-0016/02-Discussion_paper_sXML.doc (Accessed on December 2009).

Bray, T., Paoli, J., Sperberg-McQueen., C.M., Maler, E., Yergeau, F., 2008. Extensible Markup Language (XML) 1.0 (Fifth Edition). Availabel at: http://www.w3.org/TR/REC-xml/ (Accessed on December 2009)

Brown, R.D., 1999. Digital signature for XML, GlobeSet. Inc. XMLDSIG Working Group. Available at: http://www.w3.org/Signature/Drafts/draft-ietf-xmldsig-signature-00.txt (Accessed 5 December 2008)

Brinkman, R., Feng, L., Doumen, J., Hartel, P.H., Jonker, W., 2004. Efficient Tree Search in Encrypted Data. Information systems security, 13 (3). pp 14-21.

Brinkman, R., Schoenmakers, B., Doumen, J., Jonker, W., 2005. Experiments with Queries over Encrypted Data Using Secret Sharing. Lecture Notes in Computer Science, Volume 3674/2005, pp 33-46.

Burmester, M, Desmedt, Y., Doi, H., Mambo, M., 2004. A structured ELGamal-Type Multisignature Scheme. Lecture Notes in Computer Science, Volume 1751/2004, pp 466-483.

Carminati, B., Ferrari, E., Bertino, E., 2005. Securing XML data in third-party distribution systems. The ACM conference on information and knowledge management, Bremen, Germany, 2005, pp 99-106.

Celko, J., (2004). Trees and Hierachies in SQL: Adjacency List Model, Available at: http://www.sqlsummit.com/AdjacencyList.htm (Access on February 2009).

Chan, C., Felber, P., Garofalakis, M., Rastogi, R., 2002. Efficient filtering of XML documents with XPath expressions. The VLDB Journal, Vol. 11, pp 354-379.

Chang, T.K, Hwang, G.H., 2007. A processing model for the optimal querying of encrypted XML documents in XQuery. In: proceedings of Eighteeth Australasian Database Conference (ADC2007), Ballarat, Victoria, Australia. Pp 43-51.

Chen, Y.H., Lu, E.J., 2004. Design of a secure fine-grained official document exchange model for e-government. Information & Security, 15(1) (2004), pp 55-71.

Chow, R., Johnson, T., 1997. Distributed Operating System & Algorithms. Addison Wesley.

Cid, C., 2006. Recent developments in cryptographic hash functions: Security implications and future directions. Information Security Technical Report. II (2006), pp 100-107.

Cody, E., Sharman, R., Rao, R.H., Upadhyaya, S., 2008. Security in grid computing: A review and synthesis. Decision Support Systems, 44 pp 749–764.

Czech, Z. J., Havas, G., Majewski, B. S., 1992. An optimal algorithm for generating minimal perfect hash functions. Information Processing Letters, Vol. 43 Issue 5, pp 257 – 264.

Damiani, E., De, S., Di, C., Samarati, P., 2002. Towards securing XML web services. ACM workshop on XML security, November 22, 2002, USA, pp 90-96.

Dankers, J., Garefalakis, T., Schaffelhofer, R., Wright, T., 2002. Public key infrastructure in mobile system. Electronics and Communication Engineering Journal, 14(5) pp 180-190.

Devanbu, P., Gertz, M., Kwong, A., Martel, C., Nuckolls, G.,2001. Stubblebine, Flexible authentication of XML documents. In Proceeding of the 8th ACM conference on Computer and Communications Security. ACM Press, Philadelphia, USA, 2001, pp 136-145.

Diffie, W., Hellman, M., 1976. New Direction in Cryptography. IEEETransactions of Information Theory, Vol. 22 Issue 6, pp 644-654.

Dobbertin, H., Bosselaers, A., Preneel, B., 1996. RIPEMD-160: a strengthened version of RIPEMD, Fast Software Encryption, LNCS 1039, Springer-Verlag, 1996, 71-82.

Doi, H., Mambo, M., Okamoto, E., 2000. On the security of the RSA-based multisignature scheme for various group structures. ACISP 2000, LNCS 1841, pp. 352-367.

Ekelhart, A., Fenz, S., Goluch, G., Steinkellner, M., Weippl, E., 2008. XML security – A comparative literature review. Journal of Systems and Software, 81 pp 1715-1724.

Fan, W., Chan, C., Garofalakis, M., 2004. Secure XML querying with security views. In: SIGMOD Conference. pp 587-598.

Feng, L., Jonker, W., 2003. Efficient processing of secured XML metadata, OTM workshop 2003, LNCS 2889, pp 704-717.

FIPS180-2, 2002. Secure Hash Standard, available at http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf (Accessed on July 2009)

Ford, W., 1995. Advances in public-key certificate standards. ACM SIGSAC Review, Volume 13, Issue 3, pp9-15.

Ford, W., Baum, M. S., 1997. Secure Electronic Commerce. Prentice Hall PTR.

Gao, J., Wang, T., Yang, D., 2008. Xflat: Query-friendly encrypted XML view publishing, information sciences 178 (2008), pp 774-787.

Georgiadis, Christos K., Mavridis, Ioannis K., Nikolakopoulou, G., Pangalos, George I., 2002. Implementing Context and Team Based Access Control in Healthcare Intranets, Informatics for Health and Social Care, Volume 27, pp 185 – 201.

Geuer-Pollmann, C., 2002. XML Pool Encryption. In proceedings of the 2002 ACM Workshop on XML Security, Nov. 22, 2002, Fairfax VA, USA. ISBN: 1-58113-632-3. pp 1-9.

Geuer-Pollmann, C., 2004. Confidentiality of XML documents by pool encryption (Unpublished PhD thesis, University Siegen, 2004)

Goyal, V., 2004a. “Certificate revocation lists or online mechanisms”, In Eduardo Fernandez-Medina, Julio Cesar Hernandez Castro, and L. Javier Garca-Villalba, editors, OSIS, INSTICC Press, pp. 261-268.

Goyal, V., 2004b. “Fast Digital Certificate Revocation”. IFIP International Federation for Information Processing. Security and Protection in Information Processing Systems. pp 488-500.

Goyal, V., 2007. Certificate Revocation using Fine Grained Certificate Space Partitioning. Financial Cryptography 2007. pp247-259.

Gudgin, M., Hadley, M., Medelsohn, N., Moreau, J., Nielsen, H., Karmarkar, A., Lafon, Y., 2007. SOAP Version 1.2 Part1: Messaging Framework (Second Edition). Available at: http://www.w3.org/TR/soap12-part1/#intro (Accessed on December 2009).

Hallam-Baker, Phillip M., Ford, W., 2001. XML Key Management Specification (XKMS). Available at: http://www10.org/cdrom/posters/1129.pdf. (Accessed on July, 2009)

Hallam-Baker, P., Mysore, S.H., 2005. XML Key Management Specification (XKMS 2.0), W3C Recommendation 28 June 2005, http://www.w3.org/TR/xkms2/. (Accessed on July, 2009)

Hardjono, T., Zheng, Y., 1992. A practical digital multisignature scheme based on discrete logarithms. Advances in Cryptology, AUSCRYPT’92, Springer, Berlin, pp 122-132

Harn, L., Kiesler, T., 1989. New scheme for digital multisignature. Electronics letters 25, pp 1002-1003.

Harn, L., 1999. Digital multisignature with distinguished signing authorities. Electronics Letters, Volume 35, Issue 4, pp 294 – 295.

Harn, L., 1994a. Group-oriented (t,n) threshold digital signature scheme and digital multisignature. IEE Proceedings Computers and Digital Techniques 141, pp 307-313

Harn, L., 1994b. New digital signature scheme based on discrete logarithms. Electronics letters 30, pp 396-398.

Hirsch, F., 2002. Getting Started With XML Security, Available at: http://home.comcast.net/~fjhirsch/xml/xmlsec/starting-xml-security.html (Accessed on December 2008)

Hirsch, F., Just, M., 2003. XML Key Management (XKMS 2.0) Requirements. W3C Note 05 May 2003. Available at: http://www.w3.org/TR/xkms2-req (Accessed on July 2009)

Hormann, T. Perlines, Wrona, H., K., Holtmanns, S., 2006. “Evaluation of certificate validation mechanisms”, Computer Communications, Volume 29, Issue 3, pp 291-305.

Housley, R., Ford, W., Polk, W., Solo, D., 2002. “Internet X.509 Public Key Infrastructure, Certificate and CRL Profile”, the IETF, RFC 3280.

Huang, H.F., Chang, C.C., 2005. Multisignatures with distinguished signing authorities for sequential and broadcasting architectures. Computer Standards & Interfaces, Volume 27, Issue 2, pp 169-176.

Hughes, M., Imamura, T., Maruyama, H., 2002. Decryption transform for XML signature, W3C recommendation 10 December 2002 (Accessed on July 2009).

Hussain, O.K., Soh, B., 2004. Maintaining the integrity of XML signatures by using the Manifest element. 30th Annual Conference of IEEE’, IEEE computer Society, Vol.1, Busan, South Korea, 2004, pp 493–195.

Imamura, T., Dillaway, B., Simon, Ed., 2002. XML Encryption Syntax and Processing, December 2002, available at http://www.w3.org/TR/xmlenc-core/ ((Accessed on February 2009)

IBM, 2008. Data integrity. Available at: http://publib.boulder.ibm.com/infocenter/tpfhelp/current/index.jsp?topic=/com.ibm.ztpf-ztpfdf.doc_put.cur/gtps5/s5dint.html (Accessed 12 December 2008)

ISO/IEC, 1997. Information technology -- Open Systems Interconnection -- Security frameworks for open systems Part 4: Non-repudiation framework.

ISO 7498-2, 1989. Information processing systems -- Open Systems Interconnection -- Basic Reference Model -- Part 2: Security Architecture

ITU. X.500, 1997. ITU-T Recommendation. Available at: http://www.itu.int/rec/T-REC-X.500-200508-I/en (Accessed on Feb. 2009)

ITU. X.509, 1997. ITU-T Recommendation. Available at: http://www.itu.int/ITU-T/asn1/database/itu-t/x/x509/1997/index.html (Accessed on Feb. 2009)

ITU-T Recommendation X.509 version 3, 1997. “Information Technology - Open Systems Interconnection - The Directory Authentication Framework”, ISO/IEC 9594-8:1997, (Accessed on Feb. 2009)

Itakura, K., Nakamura, K., 1983. A public-key cryptosystem suitable for digital multisignatures. NEC Research and Development 71, pp 1-8.

Jammalamadaka, R.C., Mehrotra, S., 2006. Querying encrypted XML documents. Proceedings of the IEEE International Database Engineering & Applications Symposium, IDEAS, 2006, pp129-136.

Jones, S., Wilikens, M., Morris, P., Masera, M., 2000. Trust requirement in e-business. Communications of the ACM, 43(12), pp 81-87.

Jonker, W., Feng, L., 2008. Method of searching in a collection of documents, USPTO Application #: 20080059404.

Joux, A., 2004. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. LNCS (3152), pp 306-316.

Kammer, R. G., 1999. Data Ecryption Standard (DES). FIPS PUB 46-3, available at: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf (Accessed on October 2010).

Karnouskos, S., 2005. Security-enabled code deployment for heterogeneous networks. Computer Standards & Interfaces, 27(5) pp 547-560.

Kiesler, T., Harn, L., 1990. RSA blocking and multisignature schemes with no bit expansion. Electronics letters 26, pp.1490-1491

Kocher, P. C., 1998. “On certificate revocation and validation”, In Financial Cryptography, pp. 172-177.

Komar, B., Kinder, C., Ben-Menahem, A., 2010. Certificate Revocation and Status Checking. Available at: http://technet.microsoft.com/en-us/library/cc770413%28WS.10%29.aspx (Accessed on October, 2010).

Krumm, J., 2007. Inference Attacks on Location Tracks. Fifth International Conference on Pervasive Computing (Pervasive 2007), LNCS 4480, pp 127-143.

Kubbilun, W., Gajek, S., Psarros, M., Schwenk, J., 2005. Trustworthy Verification and Visualisation of Multiple XML-Signatures. LNCS, Volume 3677/2005, pp 311-320.

Lee, J-G., Whang, K-Y., 2006. Secure query processing against encrypted XML data using Query-Aware decryption, Information sciences, 176 (2006), pp 1928-1947.

Leung, K.R.P.H., Hui, L.C.K., 2001. Handling signature purposes in workflow systems. The Journal of System and Software, 55 pp 245-259.

Li, Q., Moon, B., 2001. Indexing and querying XML data for regular path expressions. In proceedings of the VLDB 2001, pp. 361-370.

Li, Z.C., Hui, L.C.K., Chow, K.P., Chong, C.F., Tsang, W.W., Chan, H.W., 2000. Cryptanalysis of Harn digital multisiganture scheme with distinguished signing authorities. Electronics Letters, Volume 36, Issue 4, pp 314 – 315.

Liu, H., Luo, P., Wang, D., 2008. "A scalable authentication model based on public keys", Journal of Network and Computer Applications, 31, pp 375-386.

Liu, B., Lu, J., Yip, J., 2009a. "A Series-parallel XML Multisignature Scheme for XML Data Authentication", International Journal of Computer Science and Network Security, VOL.9 No.2, February 2009, pp. 236-247.

Liu, B., Lu, J., Yip, J., 2009b. "XML Data Integrity Based on Concatenated Hash Funcation", International Journal of Computer Science and Information Security, Vol. 1, No. 1, May 2009, pp. 31-40.

Lu, E. J-L., Chen, R-F., 2004. An XML multisignature scheme. Applied Mathematics and Computation, 149 pp 1-14.

Lu, J., Cripps, N., Chen, H., Chen, Y., 2005. An Approach to XML Key Management Specification in X-Certificator. International Conference on Internet Computing 2005: 488-493

Lu, J., Cripps, N., Chen, H., 2006. XML Security in Certificate Management, 4th Workshop on XML Technology and Applications - XML TECH'06. pp 340-346

Maruyama, H., Tamura, K., Uramoto, N., 1999. Digest Values for DOM (DOM-HASH), RFC2803. Available at: http://www.landfield.com/rfcs/rfc2803.html (Accessed 13 November 2008)

McIntosh, M., Austel, P., 2005. XML signature element wrapping attacks and countermeasures. in: SWS’05: Proceedings of the 2005 ACM Workshop on Secure Web Services. ACM Press, Fairfax, USA, 2005, pp 20-27.

Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V., 2006. On the Collision Resistance of RIPEMD-160. Lecture Notes in Computer Science, ISBN: 978-3-540-38341-3, pp 101-116.

Micali, S., 1997. “Efficient certificate revocation”, In: Proceedings 1997 RSA Data Security Conference.

Micali, S., 2002. “Novomodo: Scalable certificate validation and simplified PKI management”, In: Proceeding of 1st Annual PKI Research Workshop, Gaithersburg, Maryland, USA, pp. 15-26.

Michels, M., Horster, P., 1996. On the risk of disruption in several multiparty signature schemes. Advances in Cryptology, ASIA Crypt’96, Springer, Berlin, pp 125-132.

Mitomi, S., Miyaji, A., 2000. A general model of Multisignature Scheme with Message Flexibility, Order Flexibility and Order Verifiability. ACISP 2000, pp 298-312.

Merkle, R.C., 1989. A Certified Digital Signature. In proceedings of Advances in Cryptology, Lecture Notes in Computer Science (435), Springer-Verlag, California, USA, 1989, pp 218-238.

Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C., 1999. “X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP”, the IETF, RFC 2560.

Naor, M., Nissim, K., 1998. “Certificate revocation and certificate update”, In: Proceedings 7th USENIX Security Symposium (San Antonio, Texas).

Nielsen, R., Hamilton, B. A., 2005. “Observations from the Deployment of a Large Scale PKI”, In: Proceedings of 4th Annual PKI R&D Workshop “Multiple Paths to Trust”, NIST, Gaithersburg MD, USA, pp. 159-165.

NIST, 2001. Specification for the Advanced Encryption Standard (AES), FIPS PUB 197, available at: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (Accessed on October 2010).

NIST, 2006. Digital Signature Standard, FIPS Publication 186-3.

Noor, A., 2008. “Securing the core with an enterprise key management infrastructure (EKMI)”, in: Proceedings of the 7th symposium on Identity and trust on the Internet, Gaithersburg, Maryland, pp 98-111.

Ohta, K., Okamoto, T., 1991. A digital multisignature scheme based on the fiat-shamir scheme. Advances in Cryptology, ASIA Crypt’91, Springer, Berlin, pp. 139-148

Oliveria, E., Abdelouahab, Z., Lopes, D., 2006. Security on MASs with XML Security Specifications. Proceedings of the 17th International Conference on Database and Expert System Applications, IEEE Computer Society, Krakow, Poland, 2006, pp 5-9.

O’Neill, M., 2007. Case Notes from a Vulnerability Assessment of a Bank's Web Services. XML2007 conference & Exposition, Massachusetts, USA, 2007, pp 18-24.

Pfleeger, C. P., 1997. Security in Computing. Prentice Hall PTR.

Polivy, D. J., Tamassia, R., 2002. Authenticating distributed data using Web services and XML signatures. In proceedings of the 2002 ACM workshop on XML security, pp 80-89.

Qiao, J., 2007. Research on XML United-Signature Technology and Its Implementation. Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, IEEE Computer Society, Qingdao, China, 2007, pp 979-983.

Randall, J., 2005. Hash function update due to potential weakness found in sha-1. RSA laboratories, Technical Note.

Reagle, J., 1999. XML-Signature Requirements, Available at: http://www.w3.org/TR/xmldsig-requirements (Accessed 6 November 2008)

Rivest, R.L., 1992. The MD5 message digest algorithm. RFC 1320. Available at: http://www.ietf.org/rfc/rfc1321.txt (Accessed on Feb. 2009)

Rivest, R. L., 1998. "Can we eliminate certificate revocation lists?", Financial Cryptography, vol. 1465, pp. 178-183.

Rushinek, A., Rushinek, S., 2002. E-commerce security measures: are they worth it?. Ubiquity 3(39), pp 1.

Schmidt, A. R., Waas, F., Kersten, M. L., Florescu, D., Manolescu, I., Carey, M. J., and Busse, R., 2001. The XML Benchmark Project. Technical Report INS-R0103, CWI, Amsterdam, The Netherlands, 2001. http://monetdb.cwi.nl/xml/ (Accessed on April 2009)

Schneier, B., 1995. Applied Cryptography, 2.nd ed. John Wiley & Sons.

Schrefl, M., Grun, K., Dorn, J., 2005. SemCrypt-Ensuring privacy of electronic documents through semantic-based encrypted query processing. Proceedings of the 21st International Conference on Data Engineering Workshops, April, 2005, pp 1191-1191.

Secure hash standard (SHA), 2002. Federal Information Processing Standard Publication 180-2. Available at: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf (Accessed on Feb. 2009).

Smart, N., 2010. Cryptography: An Introduction (3rd Edition). Available at: http://www.cs.bris.ac.uk/~nigel/Crypto_Book/ (Accessed on October 2010).

Stallings, W., 2006. Cryptography and Network Security: Principles and Practices, Fourth Edition, ISBN: 0131873164, by Person Education, Inc., pp 341.

Sun, L., Li, Y., 2005. XML undeniable signature. In proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation. IEEE computer society. pp 981-985.

Tamassia, R., Triandopoulos, N., 2003. On the Cost of Authenticated Data Structures. In Proc. European Symposium on Algorithms, LNCS (2832), Budapest, Hungary, 2003, pp 2-5.

Tada, M., 2002. An Order-Specified Multisignature Scheme Secure against Active Insider Attacks. ACISP 2002, LNCS 2384, pp 328-345.

Ünay, O., Gündem, T.I., 2008. A survey on querying encrypted XML documents for databases as a service, SIGMOD Record, Vol.37 No. 1, 2008, pp 12-20.

Wang, H., Lakshmanan, L.V.S., 2006. Efficient secure query evaluation over encrypted XML databases, in proceedings of the 32nd international conference on Very large data bases, Seoul, Korea, 2006, pp 127-138.

Wang, L., Okamoto, E., Miao, Y., Okamoto, T., Doi, H., 2006. ID-based series-parallel multi signature scheme for multi-message from bilinear maps. International Workshop on Coding and Cryptograph (WCC2006), LNCS 3969, Springer-Verlag, Berlin, 2006, pp 291–303.

Wang, X., Yin, L., Yu, X., 2005a. Finding collisions in the full SHA-1, In: Eurocrypt 2005, LNCS, vol. 3494, Springer, pp19-35.

Wang, X., Yu, X., 2005b. How to break MD5 and other hash functions. In: Crypto. 2005. LNCS, vol. 3621. Springer; pp. 17-36

Wazan, A. S., Laborde, R., Barrere, F., Benzekri, A., 2008. “Validating X.509 certificates based on their quality”, in: Proceeding of the 9th International Conference for Young Computer Scientists, Hunan, China, pp. 2055-2060.

Weerasinghe, D., Elmufti, K., Rajarajan, M., Pakocevic, V., 2006. XML Security based Access Control for Healthcare Information in Mobile Environment. In proceedings of Pervasive Health Conference and Workshops, 2006. ISBN: 1-4244-1085-1, pp 1-6.

Woerner, J., Woern, H., 2005. A security architecture integrated co-operative engineering platform for organised model exchange in a Digital Factory environment. Computers in Industry, 56(4) pp 347-360.

Wu, C., Shan, H., Wang, W., Shieh, D., Chang, M., 2002. E-Government Electronic Certification Services in Taiwan, proceedings of the Second

International Workshop for Asian Public Key Infrastructures, Taipei, Taiwan, 2002, pp 1-8.

Wu, T-C., Huang, C-C., Guan, D.J., 2001. Delegated multisignature scheme with document decomposition. The Journal of Systems and Software, 55 pp 321-328.

Wu, T.S., Hsu, C.L., 2002. ID-based multisignatures with distinguished signing authorities for sequential and broadcasting architectures. Applied Mathematics and Computation, Volume 131, Issues 2-3, pp 349-356.

Xia, S., Ke, Y., Wang, C., 2009. Model design on DAS and research of XML encrypted data querying. Proceedings of the 2009 Sixth Web Information Systems and Applications Conference, IEEE Computer Society, pp 32-36.

Yamamoto, D., Ogata, W., 2007. A General Model of Structured Multisignatures with Message Flexibility. IEICE Trans. Fundamentals, Vol. E90-A, No. 1 pp 83-90.

Yang, M., Su, L., Li, J., Hong, F., 2006. Secure order-specified multisignature scheme based on DSA. Wuhan University Journal of Natural Science, Vol. 11 No. 6 pp 1614-1616.

Yang, Y., Ng, W., Lau, H.L., Cheng, J., 2006. An efficient approach to support querying secure outsourced XML information. CaiSE 2006, LNCS 4001, 2006, pp 157-171.

Yee, G., Xu, Y., Korba, L., El-Khatib, K., 2006. Privacy and Security in E-Learning. Future Directions in Distance Learning and Communication Technologies. Idea Group, Inc. 2006.NRC Publication Number: NRC 48120.

Yun, J., Chung, C., 2008. Dynamoc interval-based labeling schme for efficient XML query and update processing. The journal of systems and software 81 (2008), pp 56-70.

Zhang, C., Naughton, J., Dewitt, D., Luo, Q., Lohman, G., 2001. On supporting containment queries in relational database management systems. In proceedings of the ACM SIGMOD 2001, pp. 425-436.

Zhang, P., 2003. "Tradeoffs in certificate revocation schemes", ACM SIGCOMM Computer Communication Review, Volume 33, Issue 2, pp. 103-112.

Appendix A: List of Publications

1. Liu, B., Lu, J., Yip, J. (2009) “A Series-parallel XML Multisignature Scheme for XML Data Authentication”, International Journal of Computer Science and Network Security, VOL.9 No.2, February 2009, pp. 236-247.

2. Liu, B., Lu, J., Yip, J., (2009) “XML Data Integrity Based on Concatenated Hash Funcation”, International Journal of Computer Science and Information Security, Vol. 1, No. 1, May 2009, pp. 31-40.

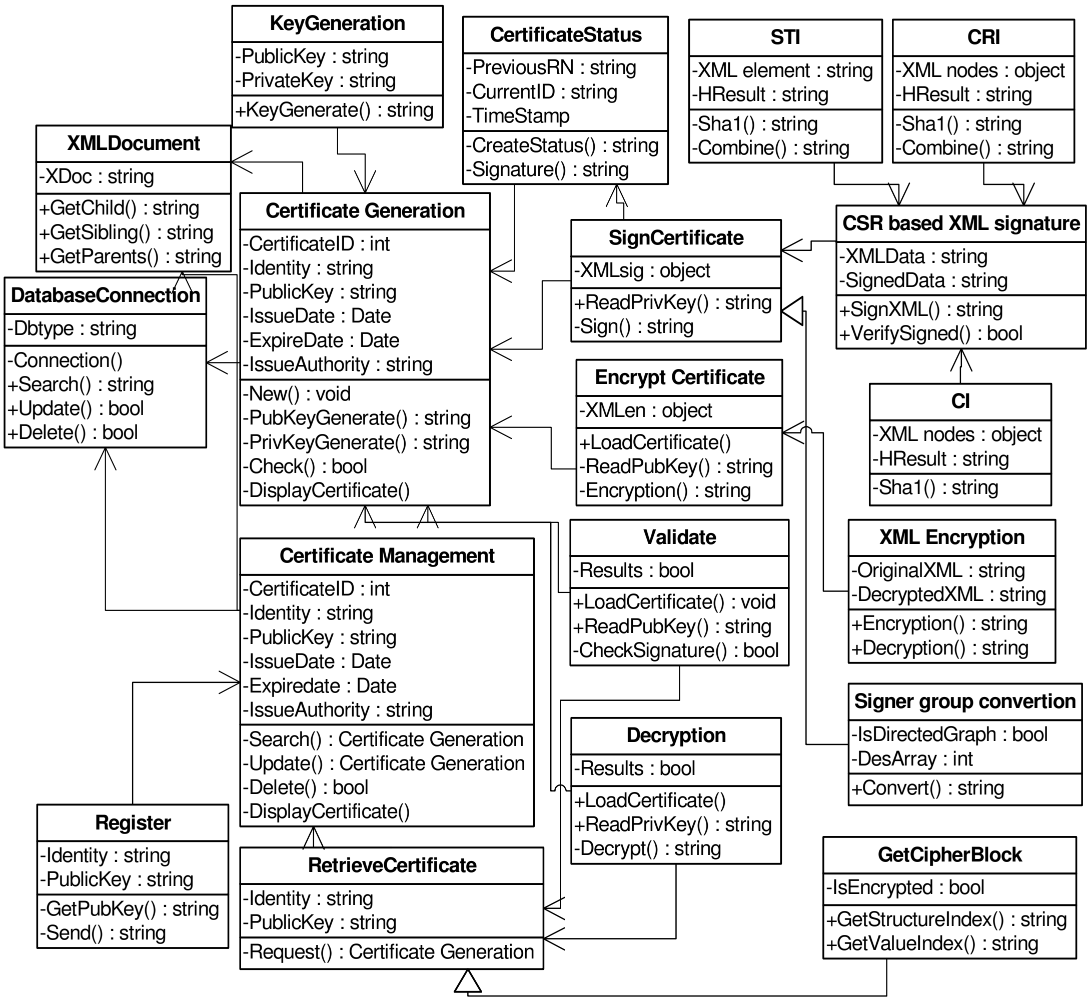
Appendix B: XMark's Auction DTD

```
<!ELEMENT site          (regions, categories, catgraph, people, open_auctions,
closed_auctions)>
<!ELEMENT categories    (category+)>
<!ELEMENT category      (name, description)>
<!ATTLIST category      id ID #REQUIRED>
<!ELEMENT name          (#PCDATA)>
<!ELEMENT description   (text | parlist)>
<!ELEMENT text          (#PCDATA | bold | keyword | emph)*>
<!ELEMENT bold                  (#PCDATA | bold | keyword | emph)*>
<!ELEMENT keyword       (#PCDATA | bold | keyword | emph)*>
<!ELEMENT emph                  (#PCDATA | bold | keyword | emph)*>
<!ELEMENT parlist       (listitem)*>
<!ELEMENT listitem      (text | parlist)*>
<!ELEMENT catgraph      (edge*)>
<!ELEMENT edge          EMPTY>
<!ATTLIST edge          from IDREF #REQUIRED to IDREF #REQUIRED>
<!ELEMENT regions       (africa, asia, australia, europe, namerica, samerica)>
<!ELEMENT africa        (item*)>
<!ELEMENT asia          (item*)>
<!ELEMENT australia     (item*)>
<!ELEMENT namerica      (item*)>
<!ELEMENT samerica      (item*)>
<!ELEMENT europe        (item*)>
<!ELEMENT item          (location, quantity, name, payment, description, shipping,
incategory+, mailbox)>
<!ATTLIST item          id ID #REQUIRED         featured CDATA #IMPLIED>
<!ELEMENT location      (#PCDATA)>
<!ELEMENT quantity      (#PCDATA)>
<!ELEMENT payment       (#PCDATA)>
<!ELEMENT shipping      (#PCDATA)>
<!ELEMENT reserve       (#PCDATA)>
<!ELEMENT incategory    EMPTY>
<!ATTLIST incategory    category IDREF #REQUIRED>
```

```
<!ELEMENT mailbox        (mail*)>
<!ELEMENT mail           (from, to, date, text)>
<!ELEMENT from           (#PCDATA)>
<!ELEMENT to             (#PCDATA)>
<!ELEMENT date           (#PCDATA)>
<!ELEMENT itemref        EMPTY>
<!ATTLIST itemref        item IDREF #REQUIRED>
<!ELEMENT personref      EMPTY>
<!ATTLIST personref      person IDREF #REQUIRED>
<!ELEMENT people         (person*)>
<!ELEMENT person         (name, emailaddress, phone?, address?, homepage?,
creditcard?, profile?, watches?)>
<!ATTLIST person         id ID #REQUIRED>
<!ELEMENT emailaddress   (#PCDATA)>
<!ELEMENT phone          (#PCDATA)>
<!ELEMENT address        (street, city, country, province?, zipcode)>
<!ELEMENT street         (#PCDATA)>
<!ELEMENT city           (#PCDATA)>
<!ELEMENT province       (#PCDATA)>
<!ELEMENT zipcode        (#PCDATA)>
<!ELEMENT country        (#PCDATA)>
<!ELEMENT homepage       (#PCDATA)>
<!ELEMENT creditcard     (#PCDATA)>
<!ELEMENT profile        (interest*, education?, gender?, business, age?)>
<!ATTLIST profile        income CDATA #IMPLIED>
<!ELEMENT interest       EMPTY>
<!ATTLIST interest       category IDREF #REQUIRED>
<!ELEMENT education      (#PCDATA)>
<!ELEMENT income         (#PCDATA)>
<!ELEMENT gender         (#PCDATA)>
<!ELEMENT business       (#PCDATA)>
<!ELEMENT age            (#PCDATA)>
<!ELEMENT watches        (watch*)>
<!ELEMENT watch          EMPTY>
<!ATTLIST watch          open_auction IDREF #REQUIRED>
```

```
<!ELEMENT open_auctions   (open_auction*)>
<!ELEMENT open_auction    (initial, reserve?, bidder*, current, privacy?, itemref, seller,
annotation, quantity, type, interval)>
<!ATTLIST open_auction    id ID #REQUIRED>
<!ELEMENT privacy         (#PCDATA)>
<!ELEMENT initial         (#PCDATA)>
<!ELEMENT bidder          (date, time, personref, increase)>
<!ELEMENT seller          EMPTY>
<!ATTLIST seller          person IDREF #REQUIRED>
<!ELEMENT current         (#PCDATA)>
<!ELEMENT increase        (#PCDATA)>
<!ELEMENT type            (#PCDATA)>
<!ELEMENT interval        (start, end)>
<!ELEMENT start           (#PCDATA)>
<!ELEMENT end             (#PCDATA)>
<!ELEMENT time            (#PCDATA)>
<!ELEMENT status          (#PCDATA)>
<!ELEMENT amount          (#PCDATA)>
<!ELEMENT closed_auctions (closed_auction*)>
<!ELEMENT closed_auction  (seller, buyer, itemref, price, date, quantity, type,
annotation?)>
<!ELEMENT buyer           EMPTY>
<!ATTLIST buyer           person IDREF #REQUIRED>
<!ELEMENT price           (#PCDATA)>
<!ELEMENT annotation      (author, description?, happiness)>
<!ELEMENT author          EMPTY>
<!ATTLIST author          person IDREF #REQUIRED>
<!ELEMENT happiness       (#PCDATA)>
```

Appendix C: Class diagram for implemented prototype



Appendix D: Relative algorithms for XML data integrity

- **Algorithm for content integrity (CI)**

Input: 1. An element or sub XML data

        2. Hash algorithm, default value is SHA1.

Output: Hash value of inputted XML data.

```
XmlNode xnodworking;
string TempNode = xnod.Name;
string strValue = (string)xnod.Value;
if (strValue != null)
   seinode = seinode + "-" + strValue;
else
   seinode = seinode + "-" + xnod.Name;
endif
if (xnod.NodeType == XmlNodeType.Element)
  if (xnod.HasChildNodes)
      xnodworking = xnod.FirstChild;
       while (xnodworking != null)
          CI(xnodworking);
          xnodworking = xnodworking.NextSibling;
       endwhile
   endif
endif
//Multi variant hash result
byte[] btr = UTF8Encoding.UTF8.GetBytes(seinode);
SHA1CryptoServiceProvider shar = new SHA1CryptoServiceProvider();
byte[] outputr = shar.ComputeHash(btr);
TempNode = BitConverter.ToString(outputr);
return TempNode //Return hashed result of content integrity
```

- **Algorithm for structure integrity**

Input: an element or a sub XML data, and start level, default value is 1

Output: Path string from root to target element

```
XmlNode xnodeworking;
string strVal = (string)xnode.Value;
if (strVal!= null)
      strVal = ":" + strVal;
else
    Tpath = Tpath + "/" + xnode.Name + intLevel.ToString();
   // Record parent  information and level information
    XmlNamedNodeMap mapAttributes = xnod.Attributes;
    foreach (XmlNode xnodAttributes in mapAttributes)
        if ((xnodAttributes.Value == "myData") && (getpath == false))
            Fpath = Tpath;
            getpath = true;
        endif
 endif
if (xnode.NodeType == XmlNodeType.Element)
    if (xnod.HasChildNodes)
        position = 0; //Record position information of an element among its sibling
        xnodeworking = xnode.FirstChild;
        while (xnodeworking!= null)
            STI(xnodeworking, intLevel + 1);
            xnodeworking = xnodeworking.NextSibling;
            if (xnodeworking!= null)
                position = position + 1;
            endif
        endwhile
    endif
endif
Return Tpath
```

Appendix E: Relative algorithms for XML multi-signature

- **Algorithm for XPath possible in DTD**

Input: XML documents

Output: XPath sets

```
private void structure(XmlNode xnod)
      XmlNode xnodworking;
      XmlNode TempNode;
      string TempPath = "";
      if (xnod.NodeType == XmlNodeType.Element)
        TempNode = xnod;
        TempPath = TempNode.Name;
        TempNode = TempNode.ParentNode;
        while (TempNode.Name != "#document")
          TempPath = TempNode.Name + "/" + TempPath;
          TempNode = TempNode.ParentNode;
        endwhile
        myCheck.Items.Add(TempPath);
        if (TempPath != myCheck.Items[0].ToString())
         myRelatives.Items.Add(TempPath);
        endif
        if (xnod.HasChildNodes)
          xnodworking = xnod.FirstChild;
          if (xnodworking.NodeType == XmlNodeType.Element)
            while (xnodworking != null)
               structure(xnodworking);
               xnodworking = xnodworking.NextSibling;
            endwhile
          endif
        endif
      endif
```

- **Algorithm for series-parallel graph to sub signing group**

Node set: $N = [1..n]$ of integer, edge set G , $G = \begin{bmatrix} u_i, u_j \\ \vdots \\ u_m, u_n \end{bmatrix}$, $m > i$.

Converted results $G_K = \begin{bmatrix} u_i & \cdots & u_j \\ \vdots & \vdots & \vdots \\ u_m & \cdots & u_n \end{bmatrix}$

Input: node set N and edge set G

Output: Converted subgroup G_k

```
// Initial matrix
For (i=0 to row($G_k$)-1) do
   For (j=0 to col($G_k$)-1) do
     $G_k$(i,j)=0;
endfor
For (i=0 to row(G)-1) do
     m=G(i,0);
     $flag_I$=true;
     $j_1$=0;
     while ($flag_I$) and ($j_1$<=row($G_K$)-1) do
          $R_1$=0;
          while ($flag_I$) and ($R_1$<=row($G_K$)-1) do
               If ($G_K$($j_1$, $R_1$)==m) then
                  $flag_I$=false;
               $R_1$= $R_1$+1
           endwhile
           $j_1$= $j_1$+1;
      endwhile
    $flag_T$=true;
     $j_2$=0;
     while ($flag_T$) and ($j_2$<=row($G_K$)-1) do
          $R_2$=0;
          While ($flag_T$) and ($R_2$<=row($G_K$)-1) do
              If ($G_K$($j_1$, $R_2$)==m) then
                  $flag_T$ =false;
               $R_2$= $R_2$+1
```

```
            endwhile
            $j_2$= $j_2$+1;
      endwhile
If ((not $flag_I$) and (not $flag_T$)) then
    G[0]=m;  G[1]=n;
endif
If (($flag_I$) and (not $flag_T$)) then
   G[$j_1$]=n;
endif
If (($flag_I$) and ($flag_T$)) then
  While ($j_2$<= $j_1$) do
    G[$j_2$]=0; G[$j_1$+1]=n;
  endwhile
endif
endfor
```