XML Encryption Syntax and Processing

Duan,Limiao

07,12,2006

Agenda

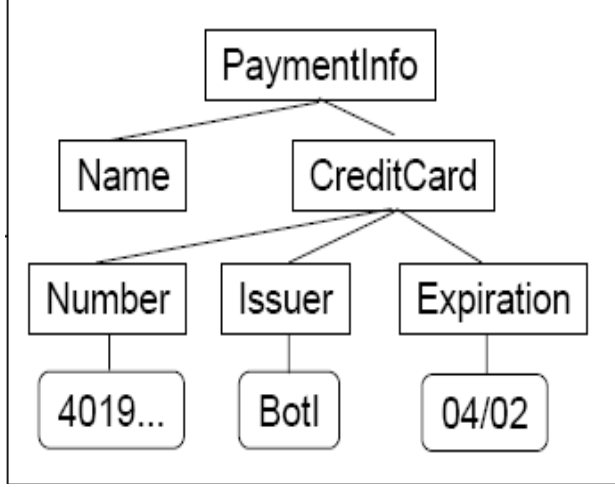
- Introduction
- Encryption Overview and Examples
  - **An XML Element**
  - **XML Element Content (Elements)**
  - **XML Element Content (Character Data)**
  - **Arbitary Data and XML documents**
- Encrytion Syntax
- Processing Rules
- Algorithms
  - **Symmetric Encryption Algorithms**
  - **Asymmetric Encryption Algorithms**
  - **Message Diges**
- Relationship with the XML Signature

Introduction

- XML Encryption is an encryption technology that is optimized for XML data.Its practical benefits include partial encryption, which encrypts specifc tags contained in XML data, multiple encryption, which encrypts data multiple times, and a complex encryption, such as the designation of recipients who were permitted to decrypt respective portions of data. The use of XML Encryption also helps solve security problems, including XML data eavesdropping

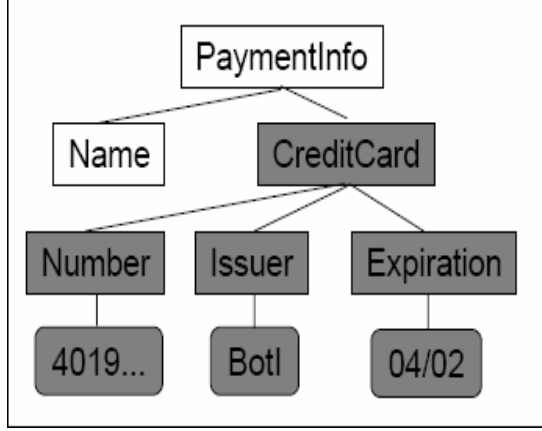
- XML Encryption was established by the W3C as formal version of W3C recommendations in Dec.2002. The W3C also established related specifications that solve problems raised when XML Encryption and XML Signature are used in combination .

XML File



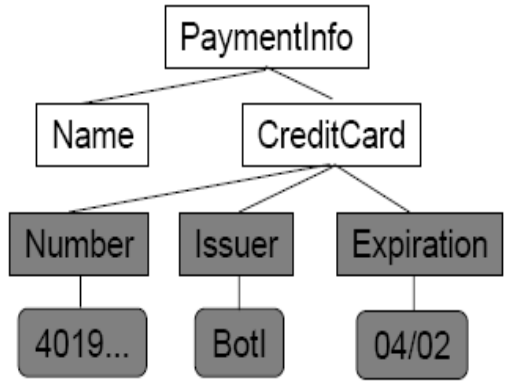
- Credit Card data to be encrypted
  - <?xml version='1.0'?>
  - <PaymentInfo xmlns='http://example.org/paymentv2'>
    - <Name>John Smith<Name/>
    - <CreditCard Limit='5,000' Currency='USD'>
    - <Number>4019 2445 0277 5567</Number>
    - <Issuer>Bank of the Internet</Issuer>
    - <Expiration>04/02</Expiration>
    - </CreditCard>
  - </PaymentInfo>

Encryption Examples I



- **Scenario 1: Encrypt the entire CreditCard XML block:**
  - <?xml version='1.0'?>
  - <PaymentInfo xmlns='http://example.org/paymentv2'>
  - <Name>John Smith<Name/>
  - *<EncryptedData Type='http://www.w3.org/2001/04/xmlenc#**Element**' xmlns='http://www.w3.org/2001/04/xmlenc#'>*
  - *<CipherData>*
  - *<CipherValue>A23B45C56</CipherValue>*
  - *</CipherData>*
  - *</EncryptedData>*
  - </PaymentInfo>

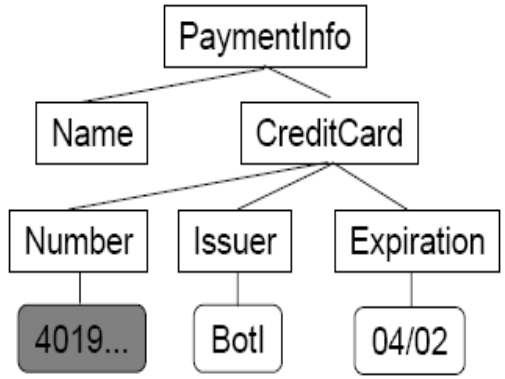
Encryption Examples II



- **Scenario 2: Encrypt only card's number, issuer, and expiration date**

```
– <?xml version='1.0'?>
– <PaymentInfo xmlns='http://example.org/paymentv2'>
–   <Name>John Smith<Name/>
–   <CreditCard Limit='5,000' Currency='USD'>
–           <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
            Type='http://www.w3.org/2001/04/xmlenc#Content'>
–               <CipherData>
–                   <CipherValue>A23B45C56</CipherValue>
–               </CipherData>
–           </EncryptedData>
–   </CreditCard>
– </PaymentInfo>
```

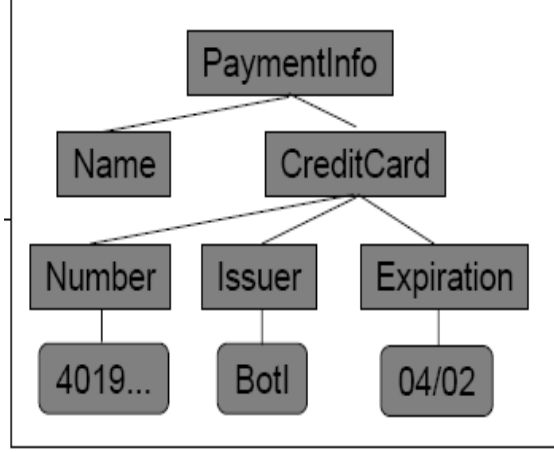
Encryption Examples III



- **Scenario 3: Encrypt only the card's Number, but indicate that the Number exists**

```
– <?xml version='1.0'?>
– <PaymentInfo xmlns='http://example.org/paymentv2'>
–   <Name>John Smith<Name/>
–   <CreditCard Limit='5,000' Currency='USD'>
–             <Number>
–             <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
                      Type='http://www.w3.org/2001/04/xmlenc#Content'>
–             <CipherData>
                      <CipherValue>A23B45C56</CipherValue>
–             </CipherData>
–             </EncryptedData>
–             </Number>
–             <Issuer>Bank of the Internet</Issuer>
–             <Expiration>04/02</Expiration>
–   </CreditCard>
– </PaymentInfo>
```

Encryption Examples IV



- **Scenario 4: Encrypt the whole XML documents**
  - <?xml version='1.0'?>
  - <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#' **MimeType=‘text/xml**'>
  - <CipherData>
  - <CipherValue>A23B45C56</CipherValue>
  - </CipherData>
  - </EncryptedData>

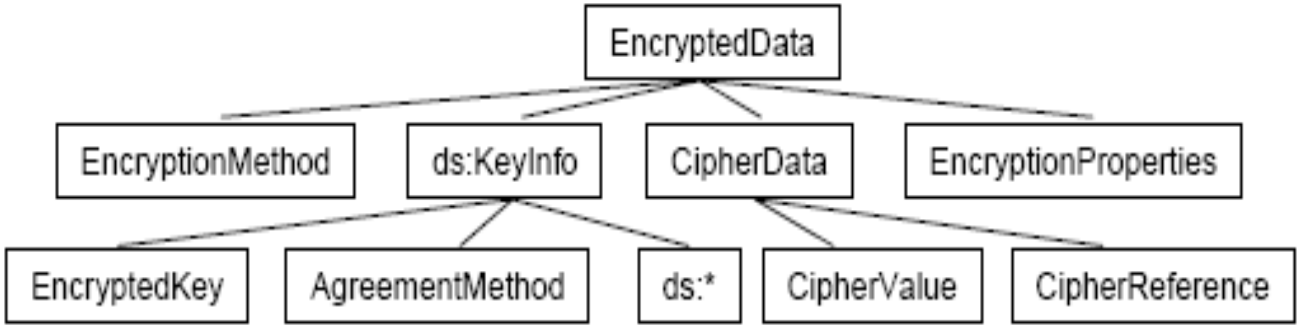
Encryption Syntax Structure

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

„?“ denotes zero or one occurence

„“ denotes zero or more occurence*

Empty element tag means the element must be empty



- The **EncryptedData** Element is the core element in the syntax. Not only it contains the encrypted data, but it's also the element that replaces the encrypted element,or serves as the new document root.
- **EncryptionMethod:** is optional, giving method of encryption, if such information is needed
- XML Encryption uses the **<ds:KeyInfo>** element defined in XML Signature spec to carry the key information. (this element is not encrypted so only use it to send Public keys)
- The **EncryptedKey** Element is used to transport encryption Keys from the originator to a known recipient.
- **CiphereData:** contains the encrypted data
- **CipherReference** identifies a source which, when Processed, yields the encrypted octet sequence
- **EncryptionProperties** can contain additional information, like a date/time stamp

Processing Rules I

- **Encryption**

For each data item to be encrypted as an `EncryptedData` *or* `EncryptedKey`*, the encryptor must*:

1. Select the algorithm to be used in encrypting this data.
2. Obtain and represent the key.
3. Encrypt the data
4. Build the `EncryptedType` (`EncryptedData` or `EncryptedKey`) structure:
5. Process EncryptedData

Processing Rules II

- **Decryption**

For each `EncryptedType` *derived element (*`EncryptedData` *or* `EncryptedKey`*), to be decrypted, the decryptor must:*

1.Process the element to determine the algorithm, parameters and `ds:KeyInfo` element to be used. If some information is omitted, the **application** MUST supply it.

2. Locate the data encryption key according to the `ds:KeyInfo` element, which may contain one or more children elements.

3. Decrypt the data contained in the `CipherData` element.

4. Process decrypted data of `Type` 'element' or element 'content'.

5. Process decrypted data if Type is unspecified or is *not* 'element' or element 'content'.

Encryption Algorithms Overview

- The following algorithms are recommended :
  - **Block Encryption:**
    - TRIPLEDES.
    - AES-128.
    - AES-256.
    - AES-192.
  - **Key Transport:**
    - RSA-v 1.5.
    - RSA-OAEP.
  - **Key Agreement:**
    - Diffie-Hellman.
  - **Symmetric Key Wrap.**
    - TRIPLEDES Key Wrap.
    - AES-128 KeyWrap.
    - AES-256 KeyWrap.
    - AES-192 KEyWrap.
  - **Message Digest:**
    - SHA1.
    - SHA256.
    - SHA512.
    - RIPEMD-160.

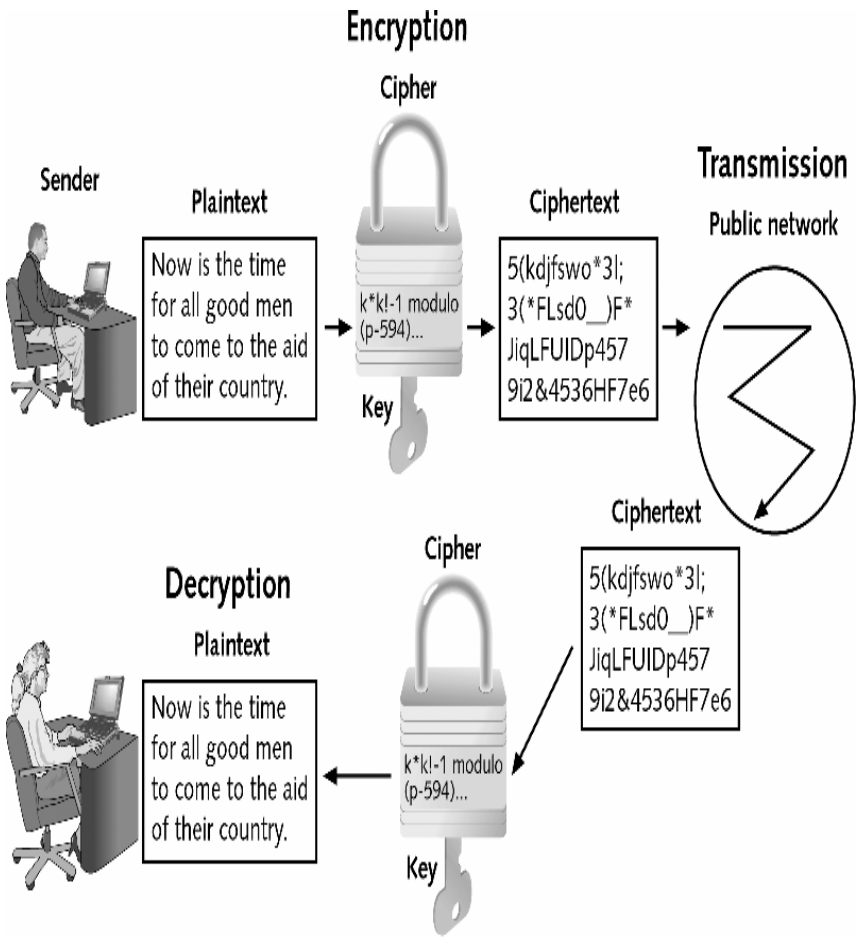


Figure 8-1 Cryptography

Symmetric Encryption Algorithms

- Most common type of cryptographic algorithm (also called private key cryptography)
- Use a single key to encrypt and decrypt a message
- With symmetric encryption, algorithms are designed to decrypt the ciphertext
  - It is essential that the key be kept confidential: if an attacker secured the key, she could decrypt any messages

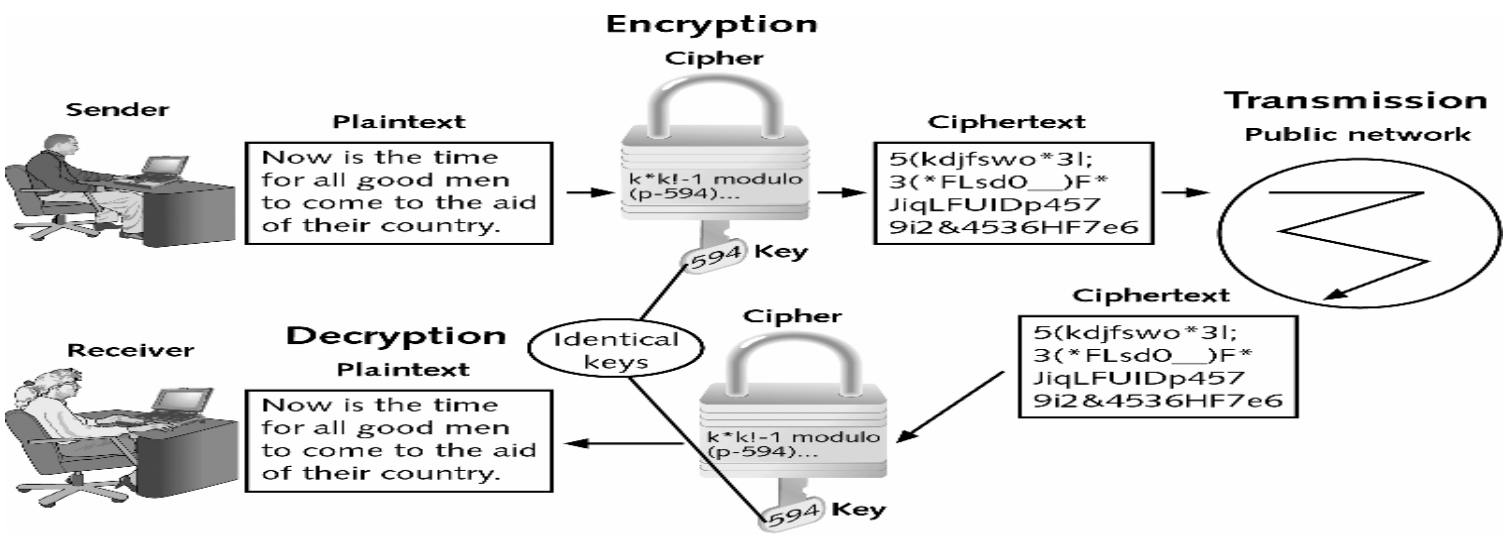


Figure 8-5 Symmetric encryption

Symmetric Encryption Algorithms (continued)

- A homoalphabetic substitution cipher maps a single plaintext character to multiple ciphertext characters
- A transposition cipher rearranges letters without changing them
- With most symmetric ciphers, the final step is to combine the cipher stream with the plaintext to create the ciphertext

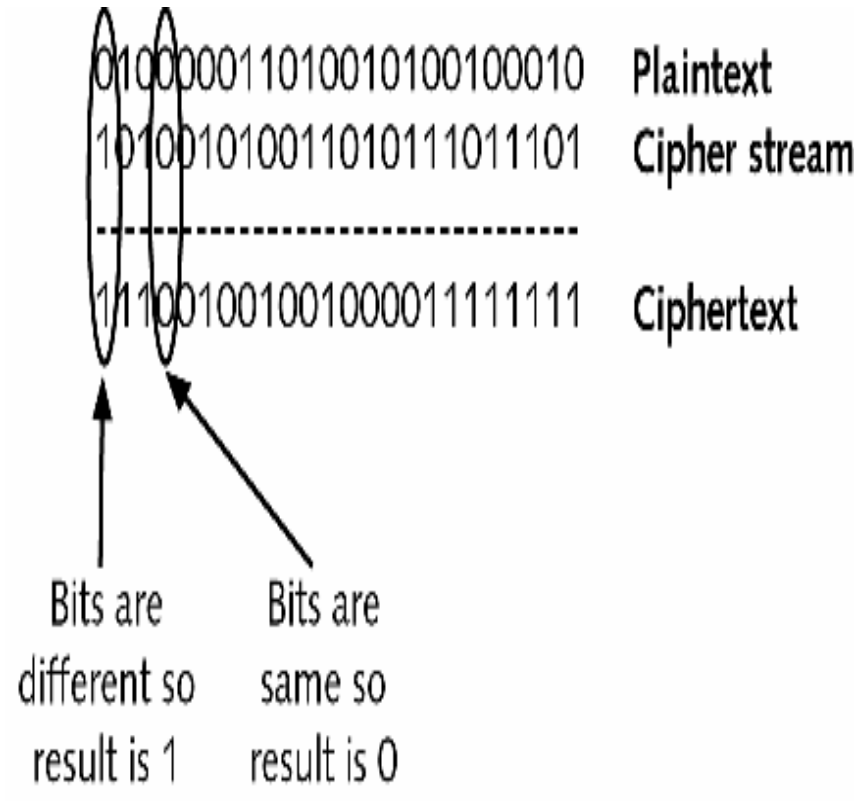


Figure 8-9 XOR operation

Asymmetric Encryption Algorithms

- The primary weakness of symmetric encryption algorithm is keeping the single key secure
- This weakness, known as key management, poses a number of significant challenges
- Asymmetric encryption (or public key cryptography) uses two keys instead of one
  - The private key typically is used to encrypt the message
  - The public key decrypts the message

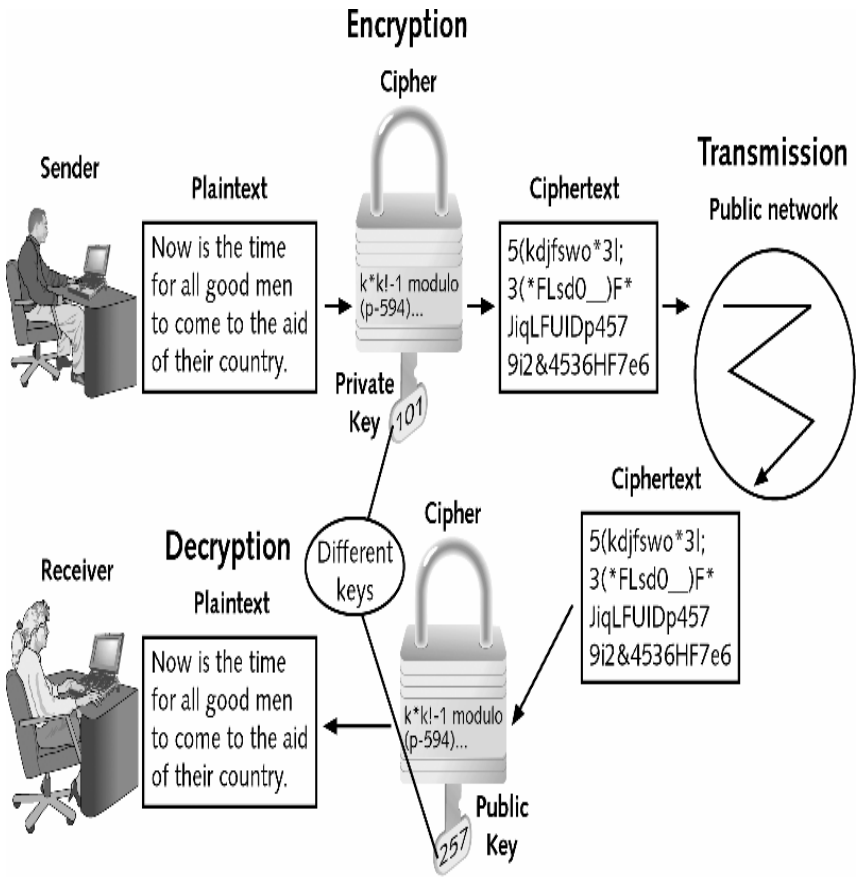


Figure 8-11 Asymmetric encryption

Message Digest (MD)

- Message digest 2 (MD2) takes plaintext of any length and creates a hash 128 bits long
  - MD2 divides the message into 128-bit sections
  - If the message is less than 128 bits, data known as padding is added
- Message digest 4 (MD4) was developed in 1990 for computers that processed 32 bits at a time
  - Takes plaintext and creates a hash of 128 bits
  - The plaintext message itself is padded to a length of 512 bits

Relationship with the XML Signature

- It is common to have encryption and signing operations to be performed many times on the same document.
- XML Encryption has proposed a **decrypt transform** mechanism for such operations.
- The Decryption Transform: defines a new <dcrpt:Except> as a child element of the <ds:Transform> element defined in the XML Signature spec.
- The <dcrpt:Except> contains a list of URIs of all the <EncryptedData> elements that have been encrypted prior to signing.
- The signature validation engine makes use of this list of URIs in order to decipher which <EncryptedData> elements need to be decrypted and which are to be left for signature validation.

Reference

- www.w3c.org
- http://publib.boulder.ibm.com/infocenter/wasinfo/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/cwbs_encrypt.html

Base64 encoding: (RFC1341)

```
30:81:99: ...
```

0011 0000 : 1000 0001 : 1001 1001 :

12 8 6 25

M I G Z

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	I	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		