

How to Break XML Encryption – Automatically

Dennis Kupser, Christian Mainka,
Jörg Schwenk, Juraj Somorovsky

Ruhr University Bochum

@jurajsomorovsky



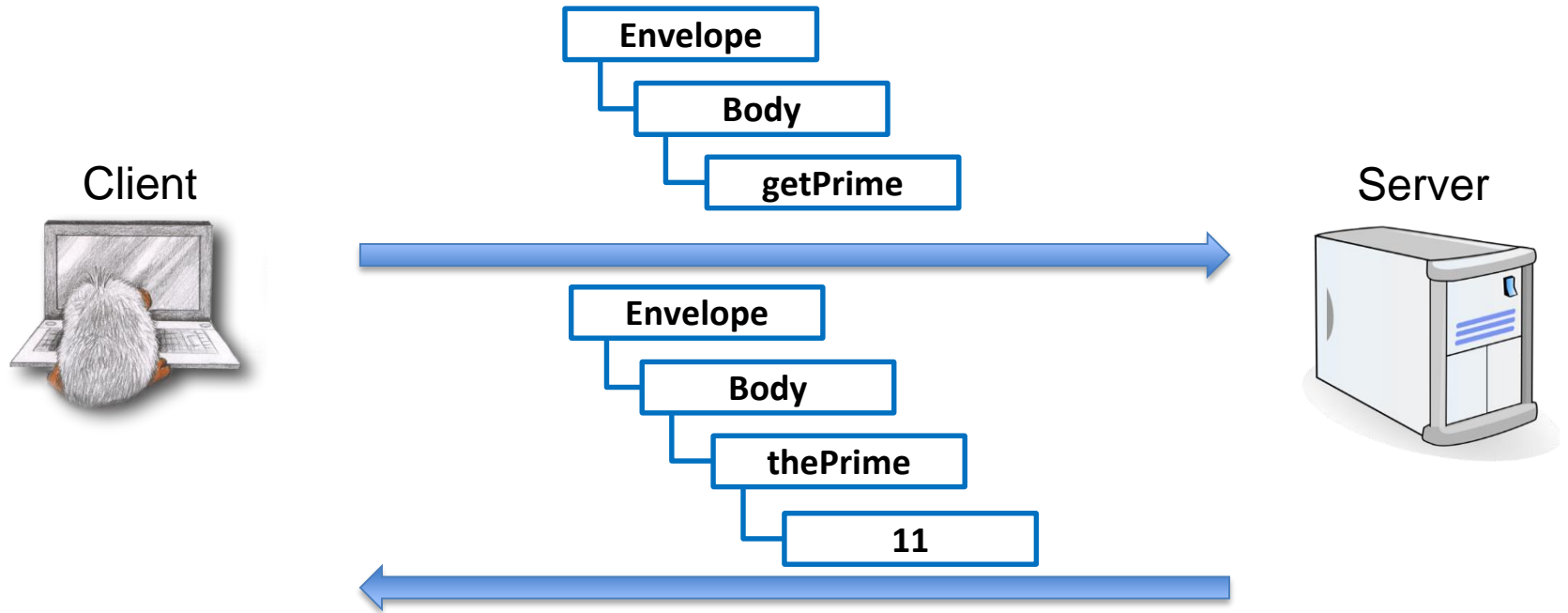
About Me and Our Institute

- Security Researcher at:
 - Chair for Network and Data Security
 - Prof. Dr. Jörg Schwenk
 - Web Services, Single Sign-On, (Applied) Crypto, SSL, crypto currencies
 - Provable security, attacks and defenses
 - Horst Görtz Institute for IT-Security
 - Further topics: embedded security, malware, crypto...
 - Ruhr University Bochum
- Penetration tests, security analyses, workshops...

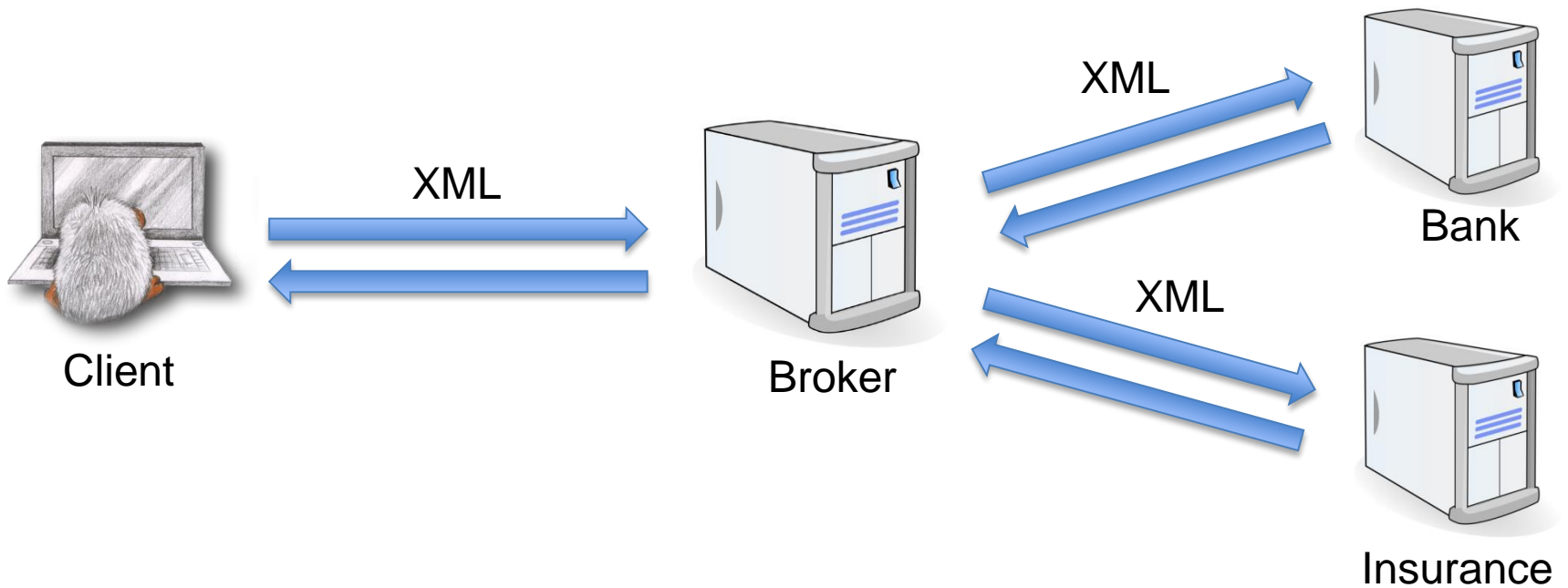
Overview

- 
- 1. What is a Web Service and XML Security**
 - 2. XML Signature Wrapping**
 - 3. Attacks on XML Encryption**
 - 4. Attacks on Symmetric Encryption Scheme**
 - 1. Attack Scenario**
 - 2. Plaintext Validity**
 - 3. Using Web Service for Plaintext Validation**
 - 4. Decrypting by Checking Plaintext Validity**
 - 5. Countermeasures and Problems**
 - 6. WS-Attacker**

What is a (SOAP) Web Service?

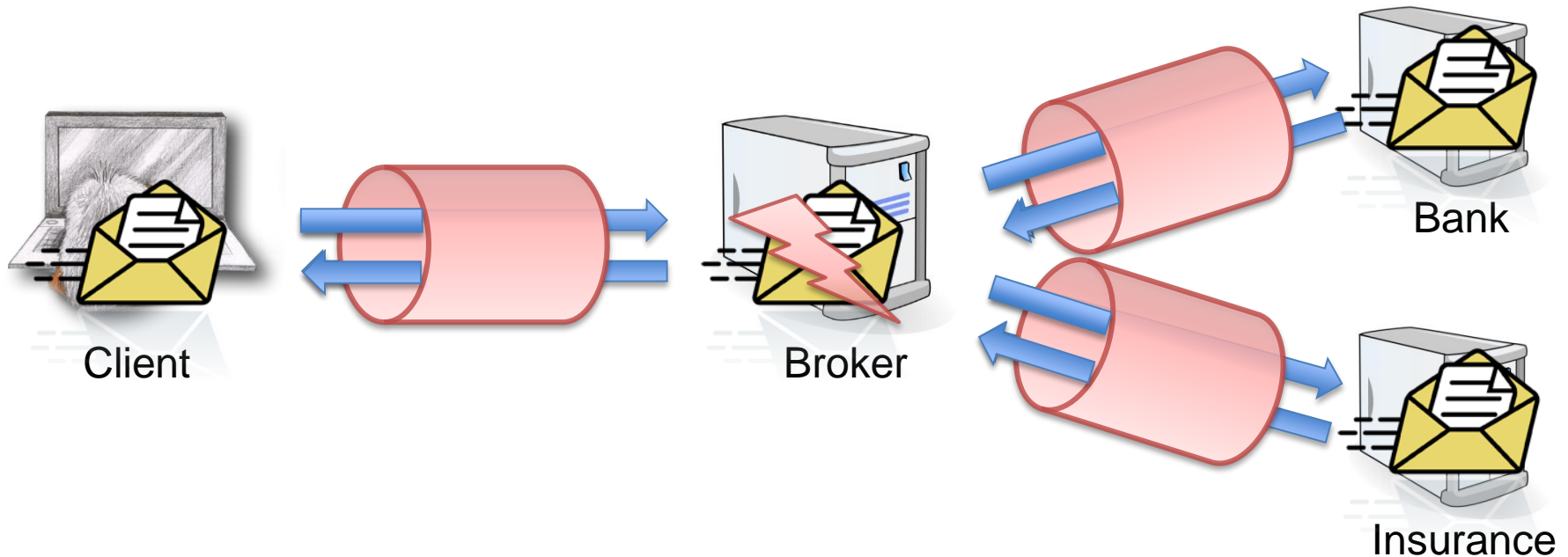


More complicated scenarios ...



Security?

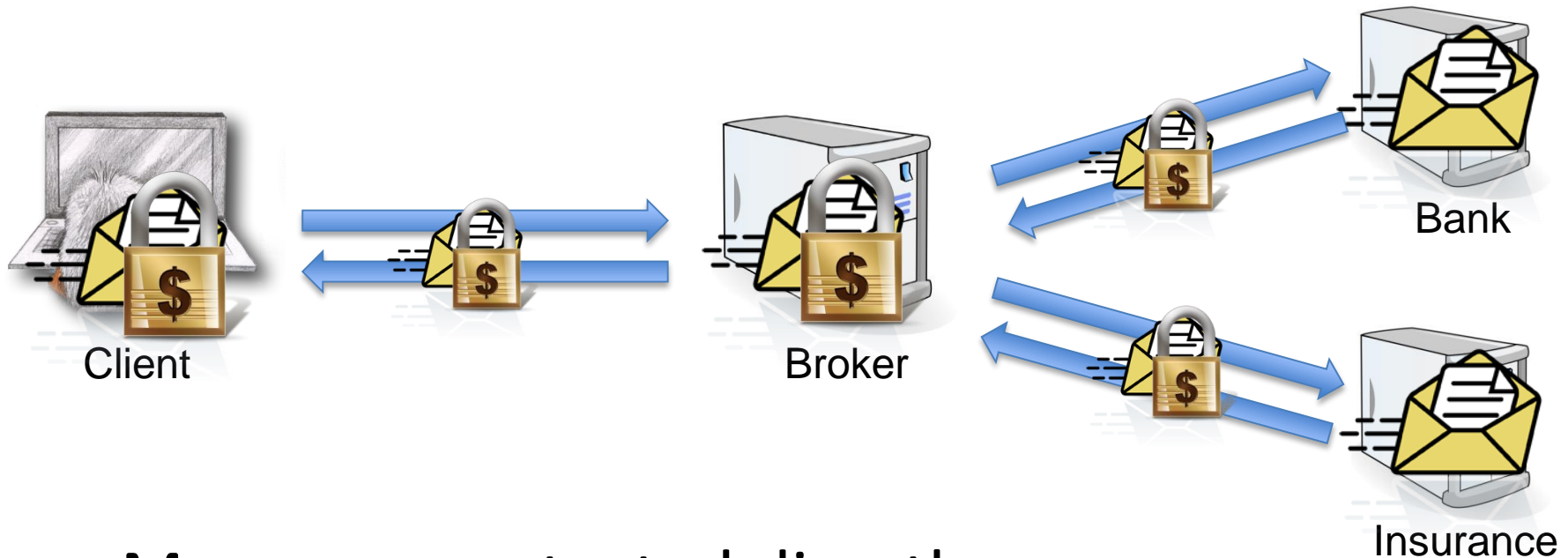
- SSL / TLS: Transport-Level Security



- Messages are only secured during transport

Motivation – XML Security

- Message Level Security



- Messages protected directly
- XML Security


XML Security

- Describes methods for applying cryptographic algorithms to XML
- **XML Signature:** protects authenticity and integrity
- **XML Encryption:** protects confidentiality
- Flexible

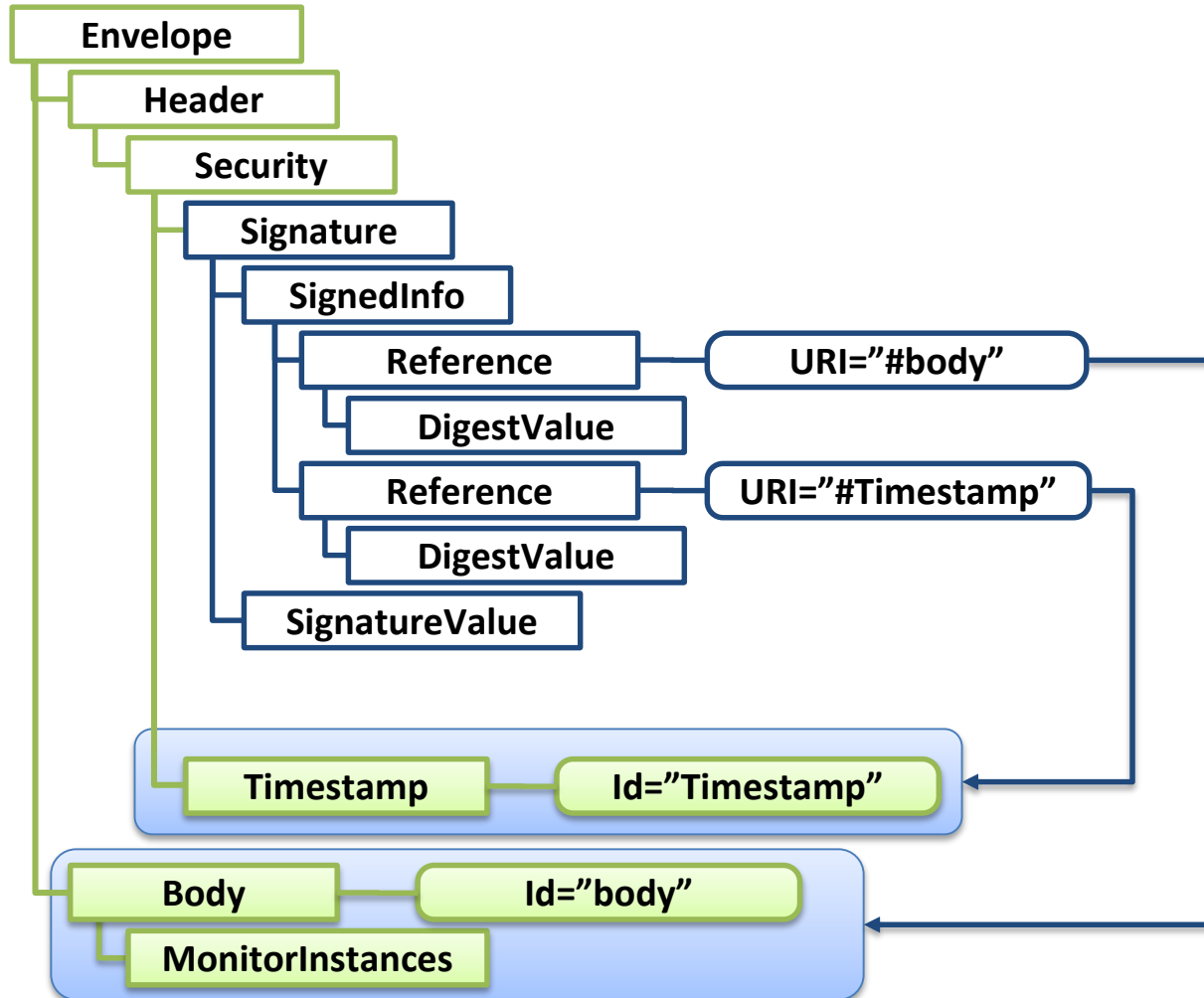
```
<PaymentInfo>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000'>
    <Number>4019 ...5567</Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```



Overview

1. What is a Web Service and XML Security
-  2. XML Signature Wrapping
3. Attacks on XML Encryption
4. Attacks on Symmetric Encryption Scheme
 1. Attack Scenario
 2. Plaintext Validity
 3. Using Web Service for Plaintext Validation
 4. Decrypting by Checking Plaintext Validity
5. Countermeasures and Problems
6. WS-Attacker

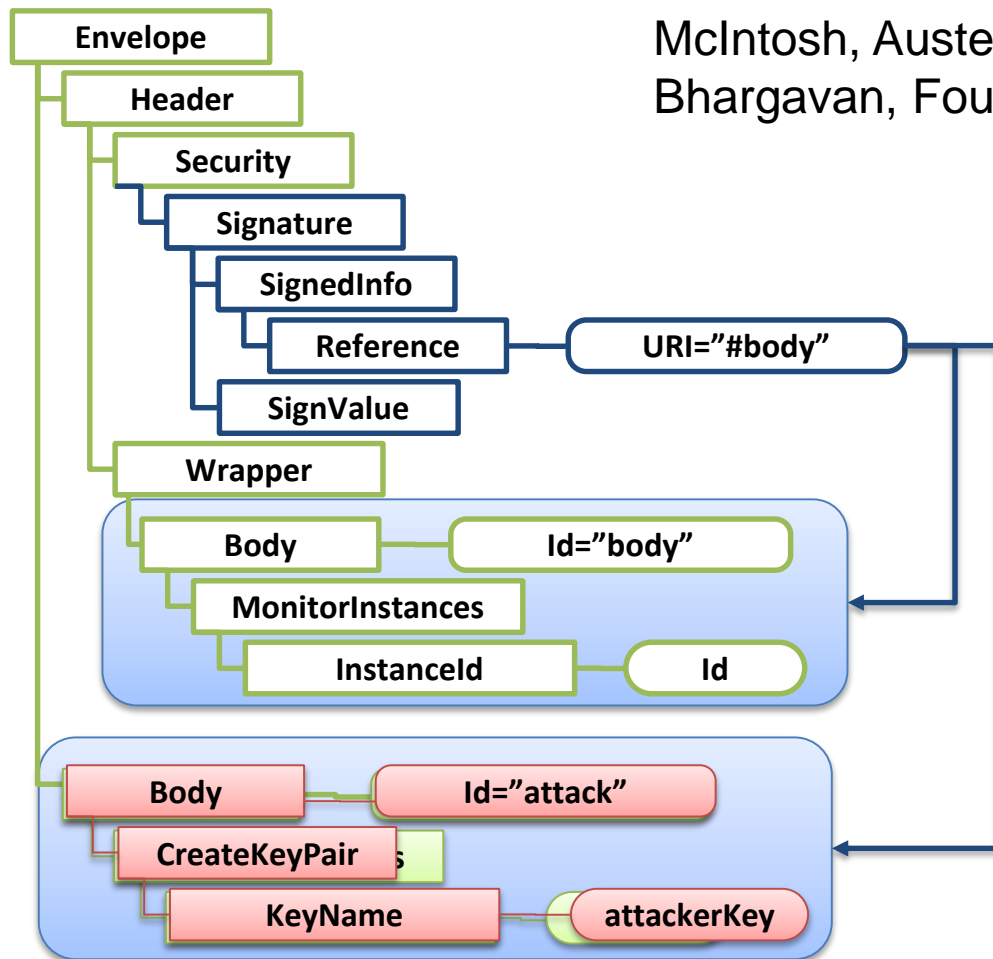
XML Signature



XML Signature Wrapping / Rewriting

McIntosh, Austel (2005)

Bhargavan, Fournet, Gordon, O'Shea (2005)

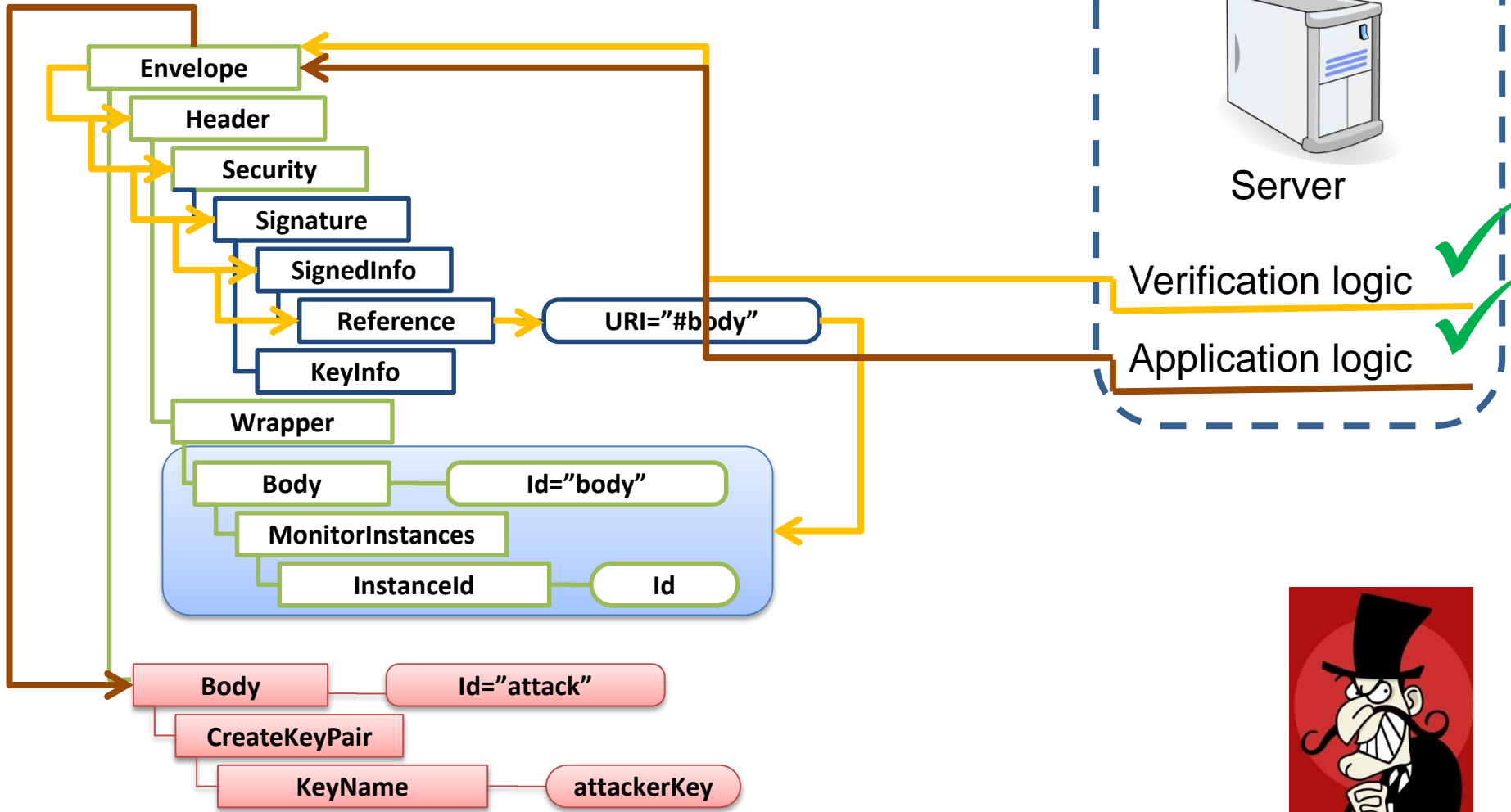


XML Signature Wrapping

Why does the attack work?

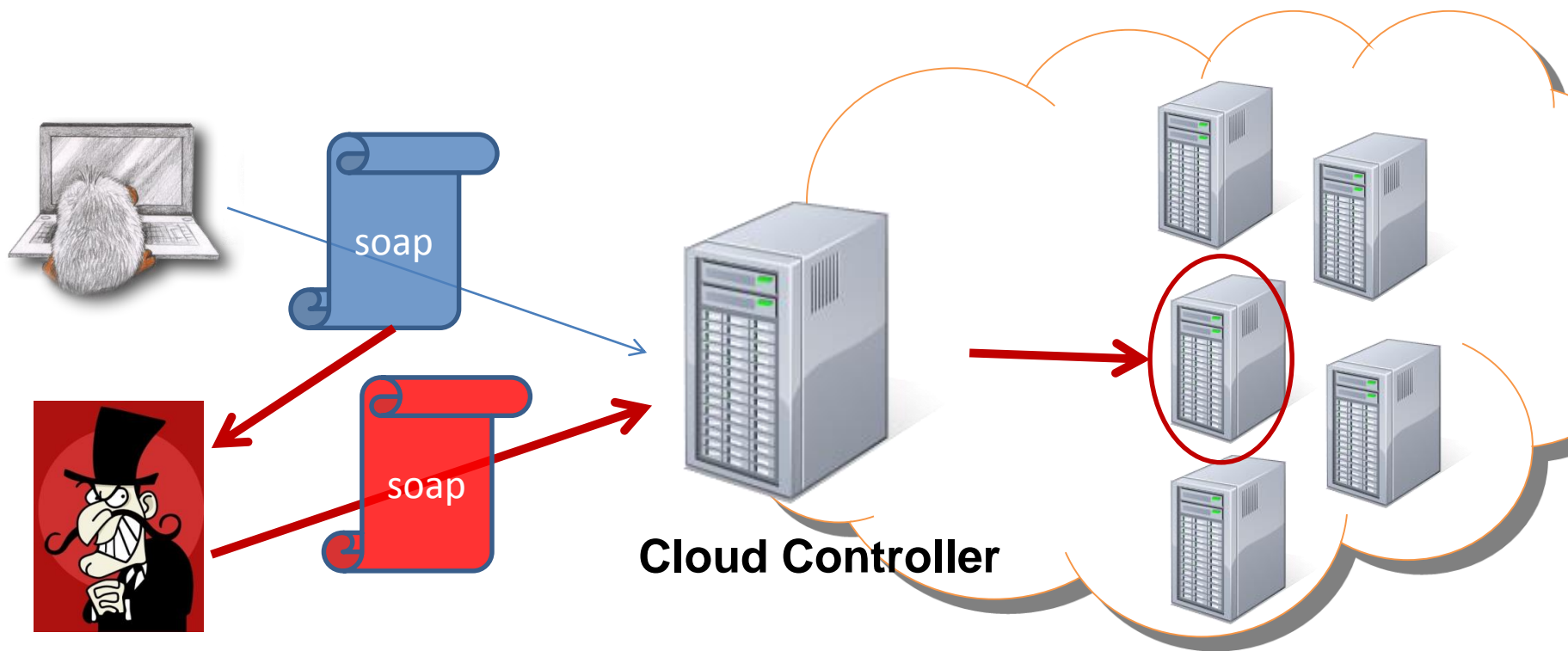


XML Signature Wrapping



XML Signature Wrapping

- Attacks on Amazon EC2 / Eucalyptus clouds
- Juraj Somorovsky, Mario Heiderich, Meiko Jensen, Jörg Schwenk, Nils Gruschka, Luigi Lo Iacono: **All Your Clouds Are Belong to Us – Security Analysis of Cloud Management Interfaces** - CCSW 2011.



Further Attacks: SAML

XXE

Signature Wrapping

Service Provider	AMI				2			AM3 CInj	Summary
	0Sig	CF	XXEA	XSLTA	RA	XSW	TRC		
Salesforce	x	x	x	x	x	x	x	x	x
Google Apps	x	x	x	x	x	x	x	x	x
Zoho	x	x	x	x	x	✓	x	x	✓
Zendesk	x	x	x	x	x	x	✓	x	✓
Clarizen	✓	x	✓	x	✓	✓	✓	x	✓
SAMange	x	x	✓	x	x	✓	✓	✓	✓
Shiftplanning	x	x	✓	x	x	x	✓	✓	✓
Panorama9	x	x	x	x	x	x	✓	x	✓
UserVoice (Marketing)	x	x	x	x	x	x	✓	x	✓
Instructure	x	x	x	✓	✓	✓	✓	x	✓
The Resumator	x	x	✓	x	x	x	✓	x	✓
BambooHR	x	x	x	x	x	x	✓	✓	✓
AppDynamics	x	x	✓	x	✓	✓	✓	x	✓
IdeaScale	x	x	✓	x	x	x	x	✓	✓
Panopto	x	x	x	x	x	✓	✓	x	✓
TimeOffManager	x	x	✓	x	✓	✓	✓	x	✓
HappyFox	x	x	x	x	x	✓	✓	x	✓
SpringCM	x	x	x	x	x	✓	x	x	✓
ScreenSteps Live	x	x	✓	x	x	✓	✓	x	✓
LiveHive	x	x	✓	x	✓	✓	✓	x	✓
Howlr	x	x	x	x	x	x	✓	✓	✓
CA Service Management	x	x	✓	x	✓	x	✓	✓	✓
Total	1	0	10	1	6	11	17	6	20/ 22

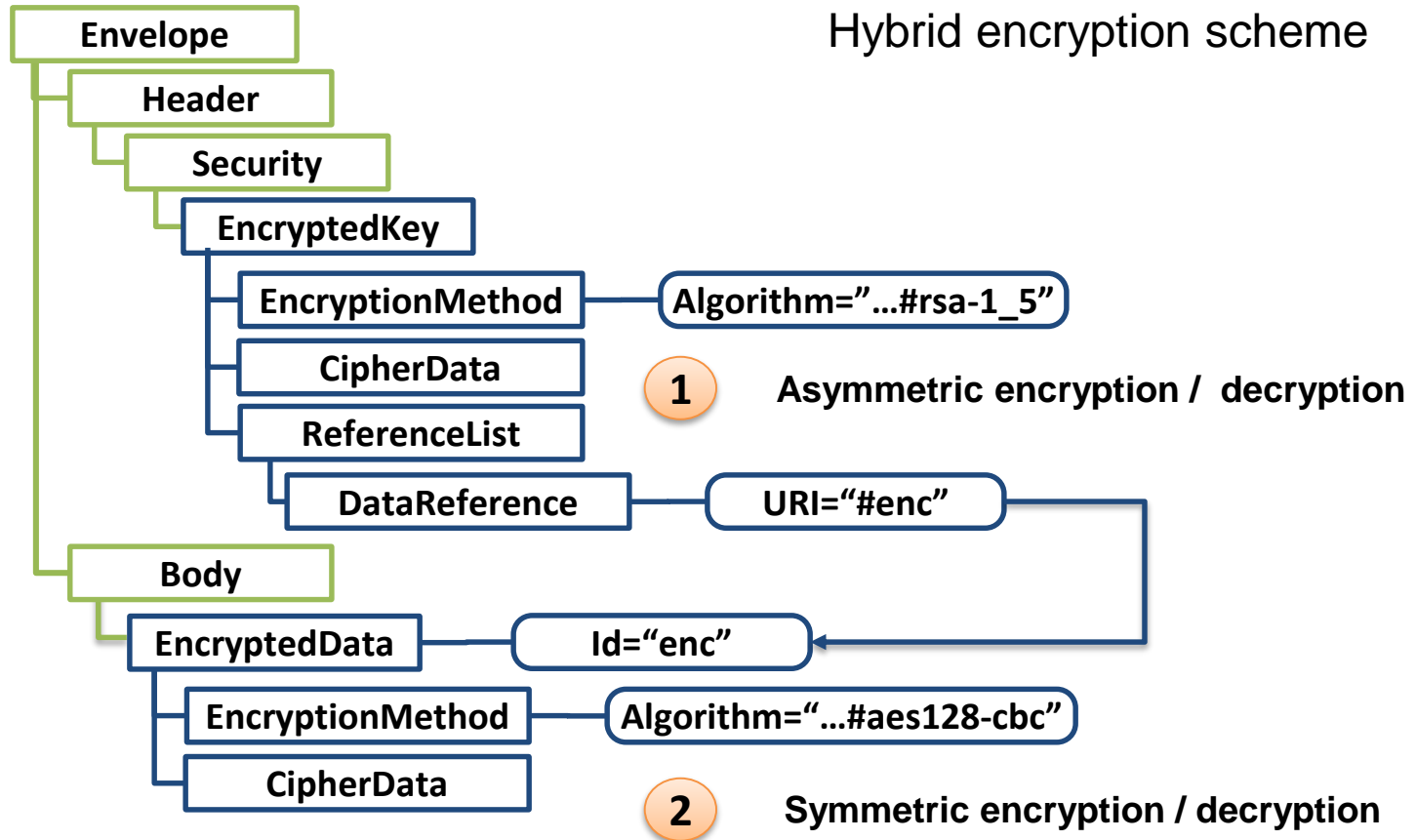
Vladislav Mladenov, Christian Mainka, Florian Feldmann, Julian Krautwald, Jörg Schwenk: **Your Software at my Service**, CCSW 2014

Overview

1. What is a Web Service and XML Security
2. XML Signature Wrapping
-  3. Attacks on XML Encryption
4. Attacks on Symmetric Encryption Scheme
 1. Attack Scenario
 2. Plaintext Validity
 3. Using Web Service for Plaintext Validation
 4. Decrypting by Checking Plaintext Validity
5. Countermeasures and Problems
6. WS-Attacker

XML Encryption

Hybrid encryption scheme



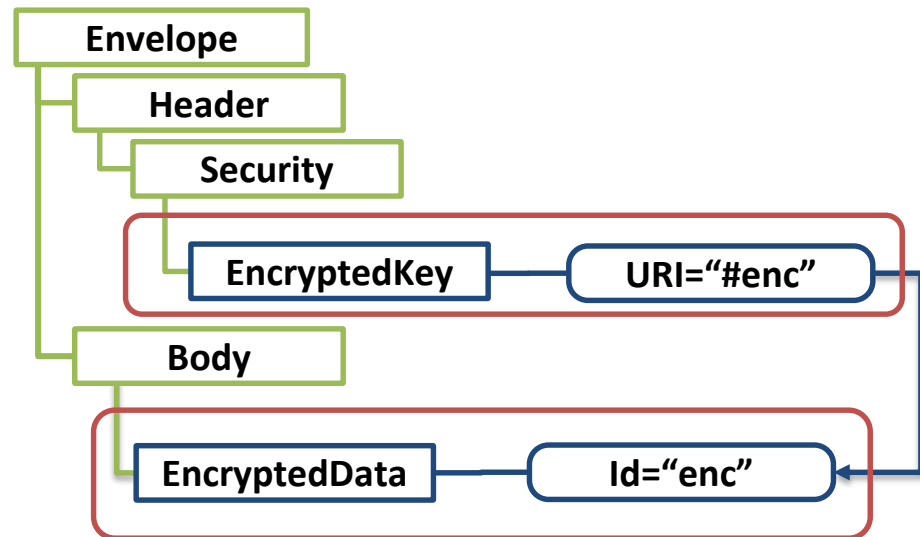
Attacks on XML Encryption

- Attacks on EncryptedKey

- Bleichenbacher’s Attack Strikes Again: Breaking PKCS#1 v1.5 in XML Encryption.
Tibor Jager, Sebastian Schinzel, Juraj Somorovsky. ESORICS 2012

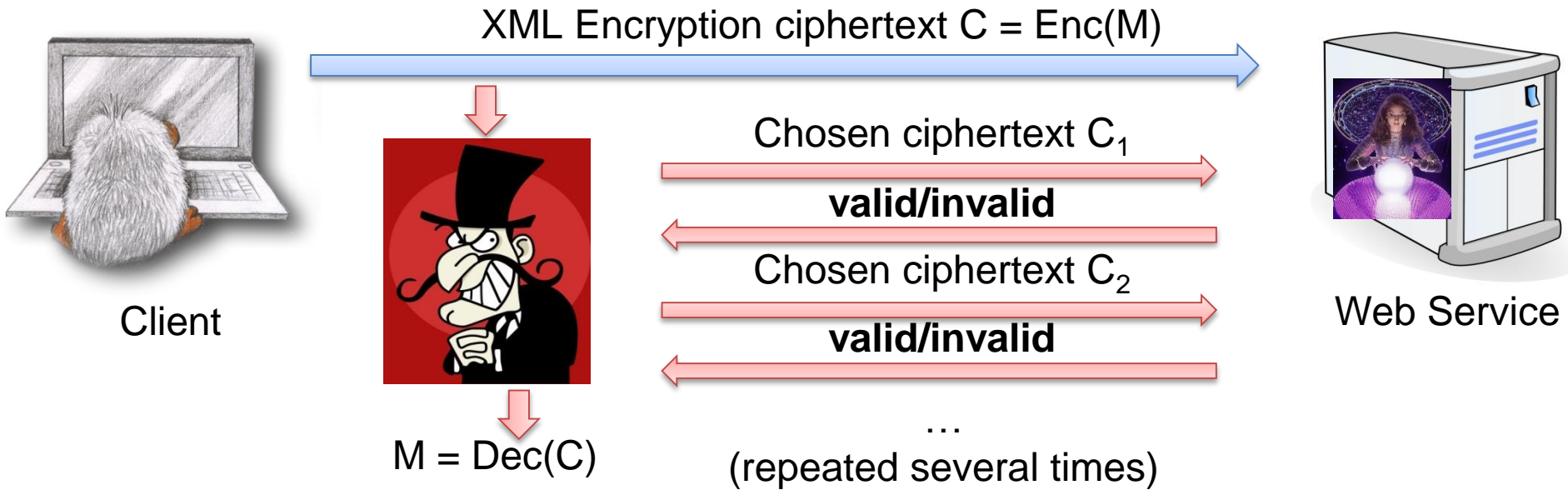
- Attacks on EncryptedData

- How to Break XML Encryption.
Tibor Jager, Juraj Somorovsky. CCS 2011



Adaptive chosen-ciphertext attacks

Adaptive chosen-ciphertext attack



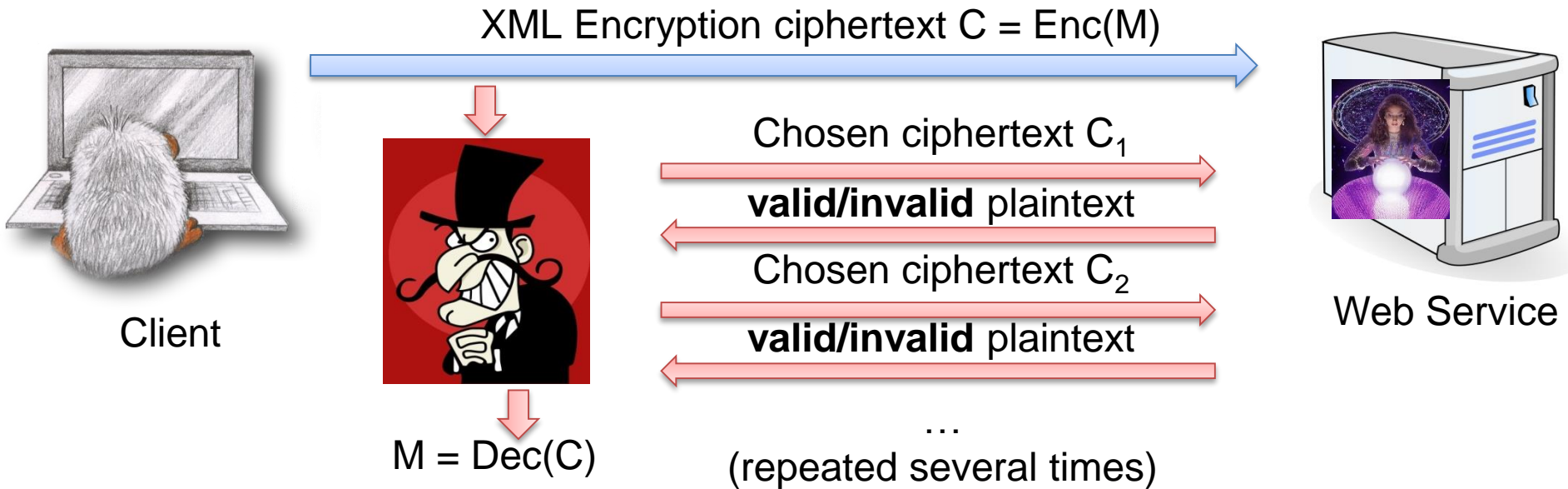
	CCS 2011	ESORICS 2012
Encryption	symmetric	asymmetric
Server-Queries	14 / plaintext byte	400k to 82M / key

Overview

1. What is a Web Service and XML Security
2. XML Signature Wrapping
3. Attacks on XML Encryption
4. Attacks on Symmetric Encryption Scheme
 1. Attack Scenario
 2. Plaintext Validity
 3. Using Web Service for Plaintext Validation
 4. Decrypting by Checking Plaintext Validity
5. Countermeasures and Problems
6. WS-Attacker

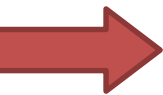


Attack Scenario



- What is a “valid” plaintext?
- How to use Web Service as “plaintext validity oracle”?
- How to use this oracle to decrypt C ?

Overview

1. What is a Web Service and XML Security
 2. XML Signature Wrapping
 3. Attacks on XML Encryption
 4. Attacks on Symmetric Encryption Scheme
 1. Attack Scenario
 2. Plaintext Validity
 3. Using Web Service for Plaintext Validation
 4. Decrypting by Checking Plaintext Validity
 5. Countermeasures and Problems
 6. WS-Attacker
- 

Plaintext Validity

- XML is a text-based data format
- XML parsing
- Characters (usually) encoded in ASCII

ASCII

0x00	NUL	0x20		0x40	@	0x60	'
0x01	(Type A)	0x21	!	0x41	A	0x61	a
0x02	(Type A)	0x22	"	0x42	B	0x62	b
0x03	(Type A)	0x23	#	0x43	C	0x63	c
0x04	(Type A)	0x24	\$	0x44	D	0x64	d
0x05	(Type A)	0x25	%	0x45	E	0x65	e
0x06	(Type A)	0x26	&	0x46	F	0x66	f
0x07	BEL	0x27	'	0x47	G	0x67	g
0x08	BS	0x28	(0x48	H	0x68	h
0x09	HT	0x29)	0x49	I	0x69	i
0x0A	LF	0x2A	*	0x4A	J	0x6A	j
0x0B	(Type A)	0x2B	+	0x4B	K	0x6B	k
0x0C	(Type A)	0x2C	,	0x4C	L	0x6C	l
0x0D	CR	0x2D	-	0x4D	M	0x6D	m
0x0E							n
0x0F							o
0x10							p
0x11							q
0x12							r
0x13							s
0x14							t
0x15	(Type A)	0x35	5	0x55	U	0x75	u
0x16	(Type A)	0x36	6	0x56	V	0x76	v
0x17	(Type A)	0x37	7	0x57	W	0x77	w
0x18	(Type A)	0x38	8	0x58	X	0x78	x
0x19	(Type A)	0x39	9	0x59	Y	0x79	y
0x1A	(Type A)	0x3A	:	0x5A	Z	0x7A	z
0x1B	ESC	0x3B	:	0x5B	[0x7B	{
0x1C	(Type A)	0x3C	<	0x5C	\	0x7C	
0x1D	(Type A)	0x3D	=	0x5D]	0x7D	}
0x1E	(Type A)	0x3E	>	0x5E	^	0x7E	~
0x1F	(Type A)	0x3F	?	0x5F	_	0x7F	DEL

„Valid“ Plaintext contains only Type B characters

Not Parsable:

Type A

Parsable:

Type B

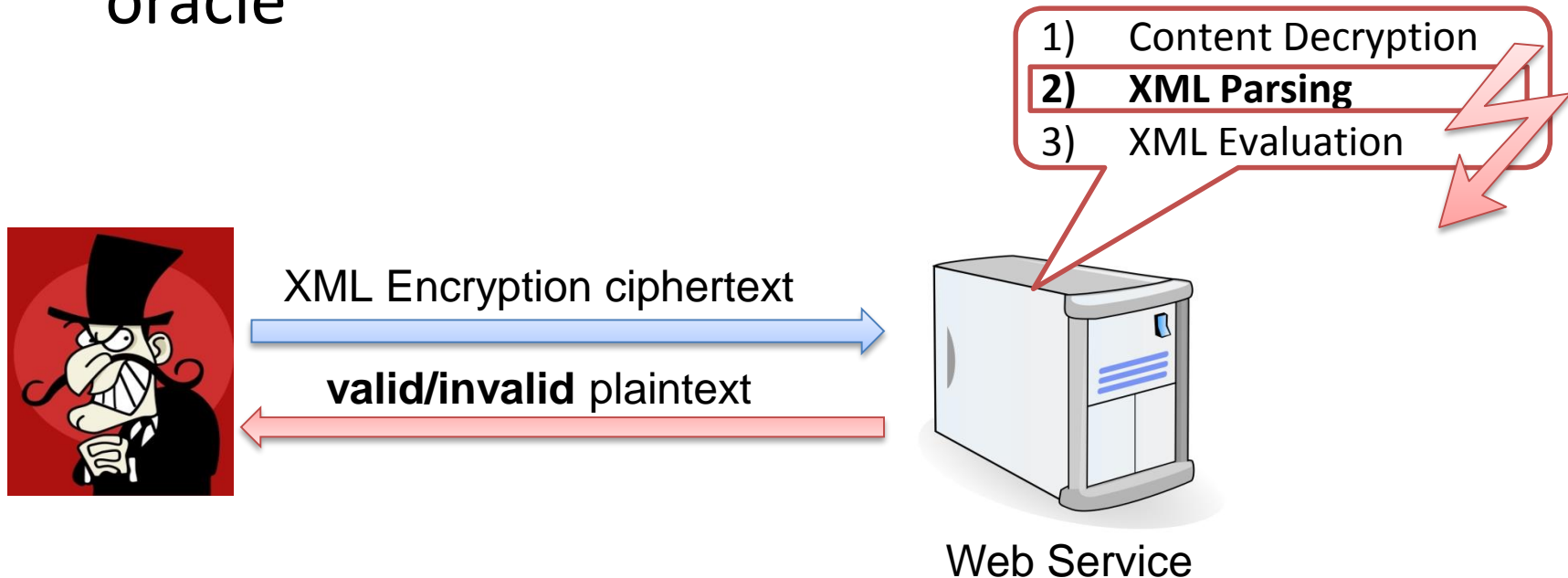
Overview

1. What is a Web Service and XML Security
2. XML Signature Wrapping
3. Attacks on XML Encryption
4. Attacks on Symmetric Encryption Scheme
 1. Attack Scenario
 2. Plaintext Validity
 3. Using Web Service for Plaintext Validation
 4. Decrypting by Checking Plaintext Validity
5. Countermeasures and Problems
6. WS-Attacker




Validity Oracle

- Using Web Services Server as plaintext validity oracle



- Invalid plaintext => Parsing error
- Parsing error => Fault message (or another side channel)

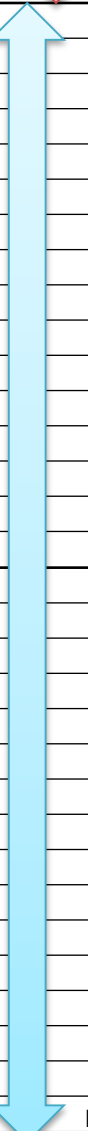
Overview

1. What is a Web Service and XML Security
 2. XML Signature Wrapping
 3. Attacks on XML Encryption
 4. Attacks on Symmetric Encryption Scheme
 1. Attack Scenario
 2. Plaintext Validity
 3. Using Web Service for Plaintext Validation
 4. Decrypting by Checking Plaintext Validity
 5. Countermeasures and Problems
 6. WS-Attacker
- 

Consider ASCII character $M_1 = (0, b_1, b_2, b_3, b_4, b_5, b_6, b_7)$



0x00	NUL	0x20		0x40	@	0x60	'
0x01	(Type A)	0x21	!	0x41	A	0x61	a
0x02	(Type A)	0x22	"	0x42	B	0x62	b
0x03	(Type A)	0x23	#	0x43	C	0x63	c
0x04	(Type A)	0x24	\$	0x44	D	0x64	d
0x05	(Type A)	0x25	%	0x45	E	0x65	e
0x06	(Type A)	0x26	&	0x46	F	0x66	f
0x07	BEL	0x27	'	0x47	G	0x67	g
0x08	BS	0x28	(0x48	H	0x68	h
0x09	HT	0x29)	0x49	I	0x69	i
0x0A	LF	0x2A	*	0x4A	J	0x6A	j
0x0B	(Type A)	0x2B	+	0x4B	K	0x6B	k
0x0C	(Type A)	0x2C	,	0x4C	L	0x6C	l
0x0D	CR	0x2D	-	0x4D	M	0x6D	m
0x0E	(Type A)	0x2E	.	0x4E	N	0x6E	n
0x0F	(Type A)	0x2F	/	0x4F	O	0x6F	o
0x10	(Type A)	0x30	0	0x50	P	0x70	p
0x11	(Type A)	0x31	1	0x51	Q	0x71	q
0x12	(Type A)	0x32	2	0x52	R	0x72	r
0x13	(Type A)	0x33	3	0x53	S	0x73	s
0x14	(Type A)	0x34	4	0x54	T	0x74	t
0x15	(Type A)	0x35	5	0x55	U	0x75	u
0x16	(Type A)	0x36	6	0x56	V	0x76	v
0x17	(Type A)	0x37	7	0x57	W	0x77	w
0x18	(Type A)	0x38	8	0x58	X	0x78	x
0x19	(Type A)	0x39	9	0x59	Y	0x79	y
0x1A	(Type A)	0x3A	:	0x5A	Z	0x7A	z
0x1B	ESC	0x3B	;	0x5B	[0x7B	{
0x1C	(Type A)	0x3C	<	0x5C	\	0x7C	
0x1D	(Type A)	0x3D	=	0x5D]	0x7D	}
0x1E	(Type A)	0x3E	>	0x5E	^	0x7E	~
0x1F	(Type A)	0x3F	?	0x5F	_	0x7F	DEL



Type A

Type B

Decrypting by checking plaintext validity

- ASCII exhibits nice pattern of Type A/B characters
- Suppose we can **flip** arbitrary plaintext bits and use a plaintext validity oracle

Decrypting by checking plaintext validity

- **Example**
- We have eavesdropped a ciphertext
 $C = \text{Enc}(\text{“Blackhat”})$
- We recover $M = \text{“Blackhat”}$ character-wise
- How to determine (b_1, b_2) of $M_1 = \text{“B”}$?

Consider ASCII character $M_1 = (0, b_1, b_2, b_3, b_4, b_5, b_6, b_7)$



0x00	NUL	0x20	@	0x60	'
0x01	(Type A)	0x21	A	0x61	a
0x02	(Type A)	0x22	B	0x62	b
0x03	(Type A)	0x23	C	0x63	c
0x04	(Type A)	0x24	D	0x64	d
0x05	(Type A)	0x25	E	0x65	e
0x06	(Type A)	0x26	F	0x66	f
0x07	BEL	0x27	G	0x67	g
0x08	BS	0x28	H	0x68	h
0x09	HT	0x29	I	0x69	i
0x0A	LF	0x2A	J	0x6A	j
0x0B	(Type A)	0x2B	K	0x6B	k
0x0C	(Type A)	0x2C	L	0x6C	l
0x0D	CR	0x2D	M	0x6D	m
0x0E	(Type A)	0x2E	N	0x6E	n
0x0F	(Type A)	0x2F	O	0x6F	o
0x10	(Type A)	0x30	P	0x70	p
0x11	(Type A)	0x31	Q	0x71	q
0x12	(Type A)	0x32	R	0x72	r
0x13	(Type A)	0x33	S	0x73	s
0x14	(Type A)	0x34	T	0x74	t
0x15	(Type A)	0x35	U	0x75	u
0x16	(Type A)	0x36	V	0x76	v
0x17	(Type A)	0x37	W	0x77	w
0x18	(Type A)	0x38	X	0x78	x
0x19	(Type A)	0x39	Y	0x79	y
0x1A	(Type A)	0x3A	Z	0x7A	z
0x1B	ESC	0x3B	[0x7B	{
0x1C	(Type A)	0x3C	\	0x7C	
0x1D	(Type A)	0x3D]	0x7D	}
0x1E	(Type A)	0x3E	^	0x7E	~
0x1F	(Type A)	0x3F	_	0x7F	DEL

Type A

Type B

Consider ASCII character $M_1 = (0, b_1, b_2, b_3, b_4, b_5, b_6, b_7)$



0x00	NUL	0x20		0x40	@	0x60	'
0x01	(Type A)	0x21	!	0x41	A	0x61	a
0x02	(Type A)	0x22	"	0x42	B	0x62	b
0x03	(Type A)	0x23	#	0x43	C	0x63	c
0x04	(Type A)	0x24	\$	0x44	D	0x64	d
0x05	(Type A)	0x25	%	0x45	E	0x65	e
0x06	(Type A)	0x26	&	0x46	F	0x66	f
0x07	BEL	0x27	'	0x47	G	0x67	g
0x08	BS	0x28	(0x48	H	0x68	h
0x09	HT	0x29)	0x49	I	0x69	i
0x0A	LF	0x2A	*	0x4A	J	0x6A	j
0x0B	(Type A)	0x2B	+	0x4B	K	0x6B	k
0x0C	(Type A)	0x2C	,	0x4C	L	0x6C	l
0x0D	CR	0x2D	-	0x4D	M	0x6D	m
0x0E	(Type A)	0x2E	.	0x4E	N	0x6E	n
0x0F	(Type A)	0x2F	/	0x4F	O	0x6F	o
0x10	(Type A)	0x30	0	0x50	P	0x70	p
0x11	(Type A)	0x31	1	0x51	Q	0x71	q
0x12	(Type A)	0x32	2	0x52	R	0x72	r
0x13	(Type A)	0x33	3	0x53	S	0x73	s
0x14	(Type A)	0x34	4	0x54	T	0x74	t
0x15	(Type A)	0x35	5	0x55	U	0x75	u
0x16	(Type A)	0x36	6	0x56	V	0x76	v
0x17	(Type A)	0x37	7	0x57	W	0x77	w
0x18	(Type A)	0x38	8	0x58	X	0x78	x
0x19	(Type A)	0x39	9	0x59	Y	0x79	y
0x1A	(Type A)	0x3A	:	0x5A	Z	0x7A	z
0x1B	ESC	0x3B	;	0x5B	[0x7B	{
0x1C	(Type A)	0x3C	<	0x5C	\	0x7C	
0x1D	(Type A)	0x3D	=	0x5D]	0x7D	}
0x1E	(Type A)	0x3E	>	0x5E	^	0x7E	~
0x1F	(Type A)	0x3F	?	0x5F	_	0x7F	DEL

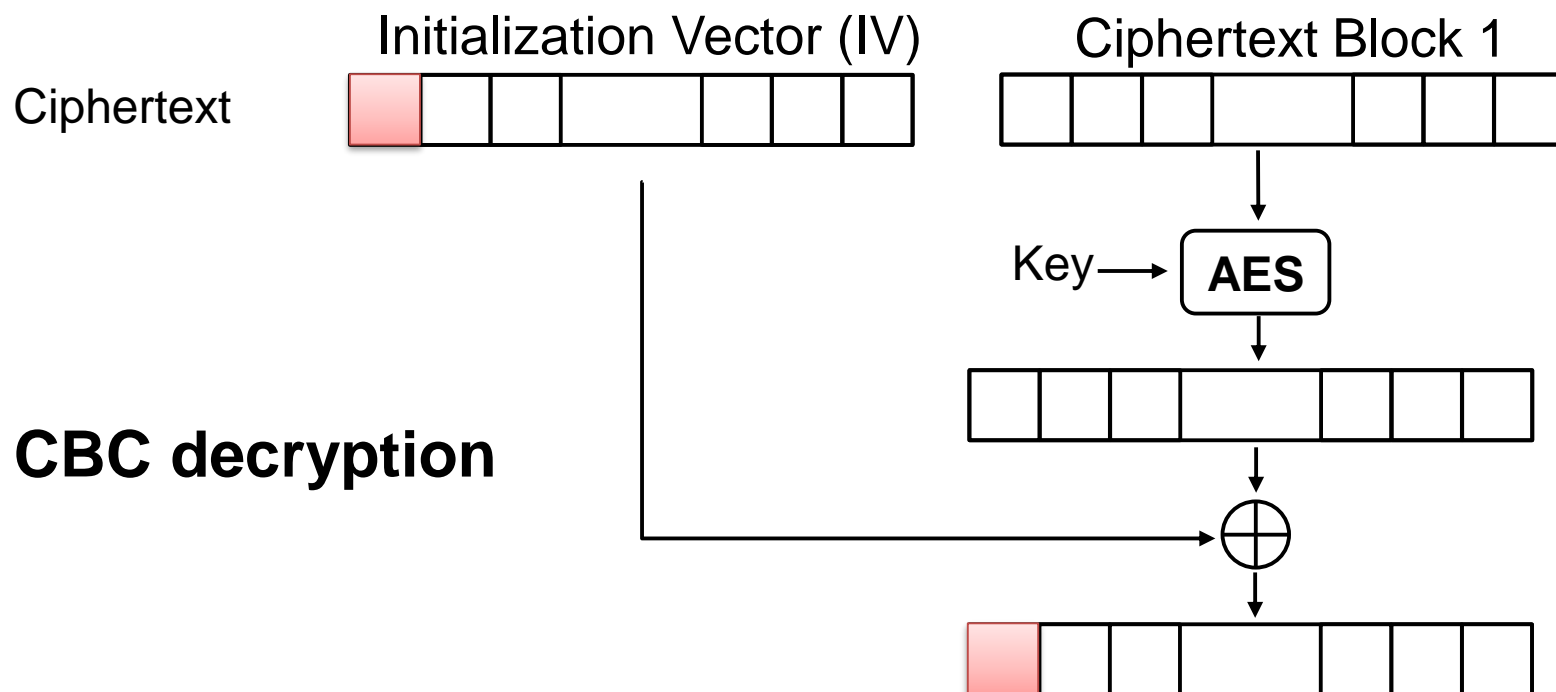
Type A

Type B

Why Flipping Possible?

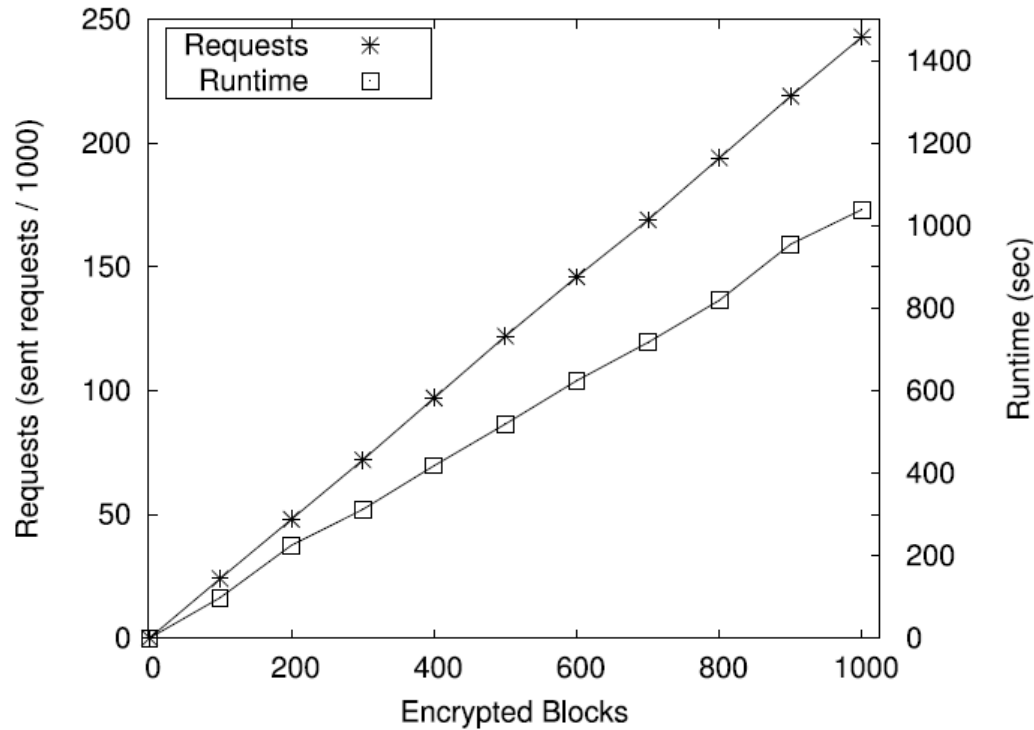


Cipher Block Chaining Mode



- Flip arbitrary bits in plaintext
- Applied in padding oracle attacks

Performance



- 14 queries / byte

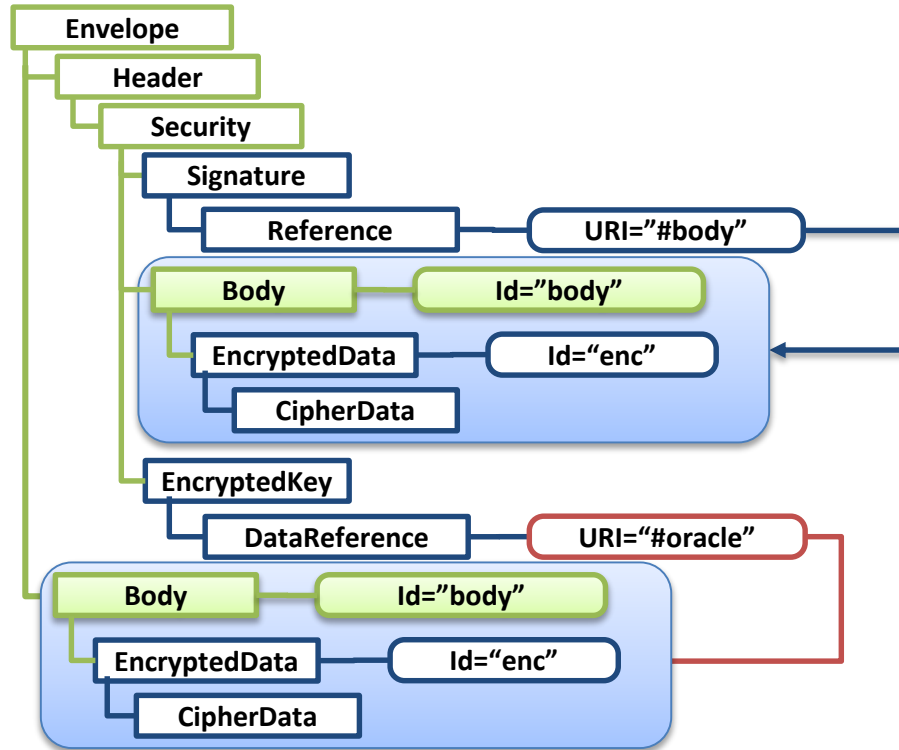
Overview

1. What is a Web Service and XML Security
2. XML Signature Wrapping
3. Attacks on XML Encryption
4. Attacks on Symmetric Encryption Scheme
 1. Attack Scenario
 2. Plaintext Validity
 3. Using Web Service for Plaintext Validation
 4. Decrypting by Checking Plaintext Validity
-  5. Countermeasures and Problems
6. WS-Attacker

Basic Idea

- Protect integrity and authenticity of ciphertexts
- Not easy...

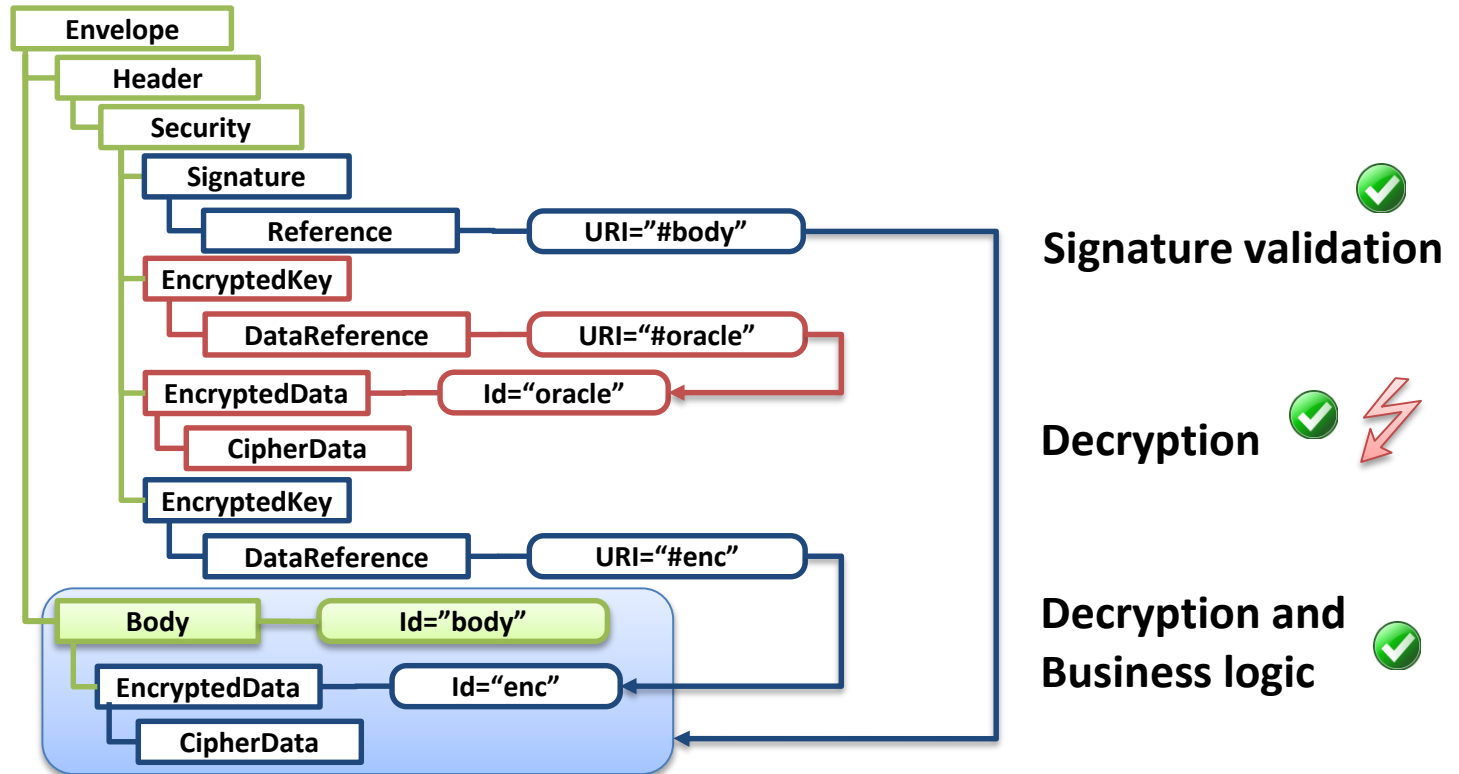
XML Signature Wrapping?



✔
Signature validation

✔
Decryption ⚡


XML Encryption Wrapping?



How to analyze Web Services Automatically?



Overview

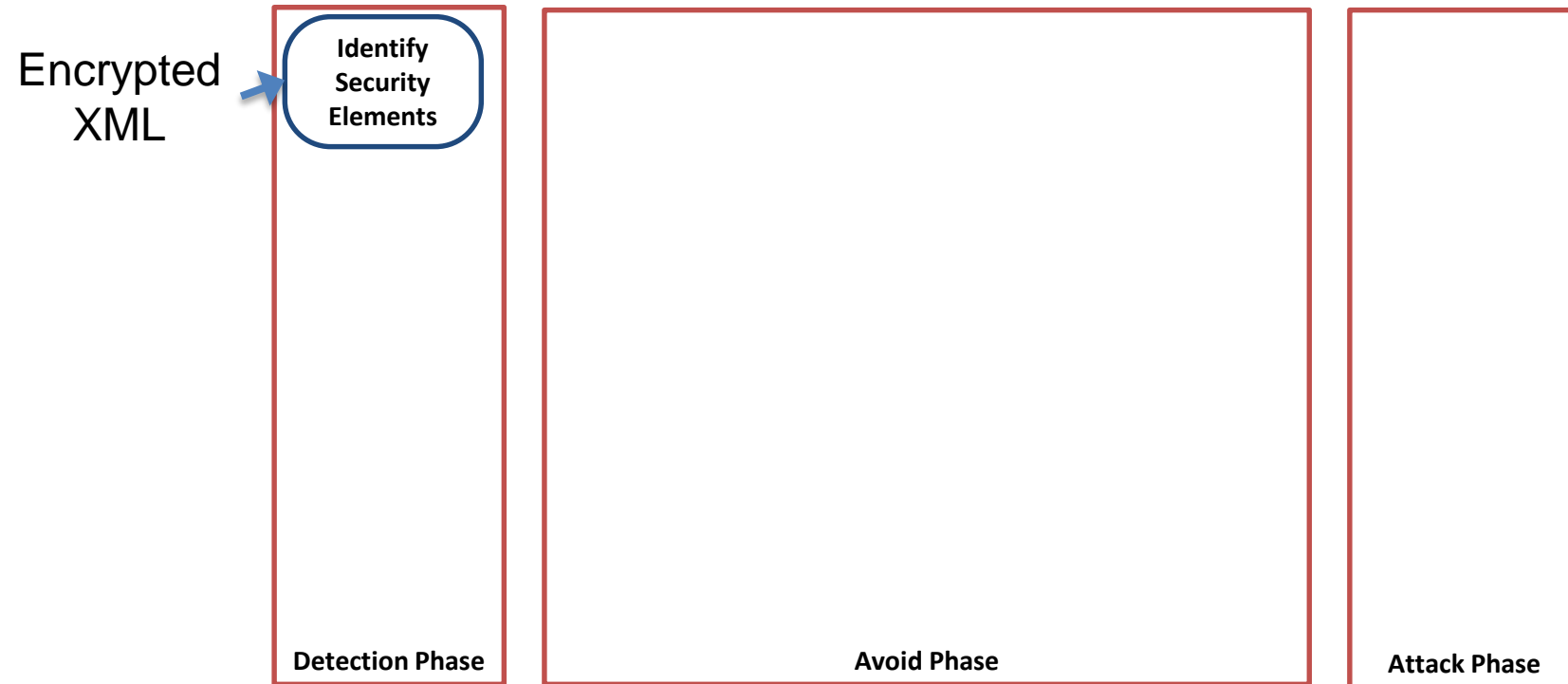
1. What is a Web Service and XML Security
 2. XML Signature Wrapping
 3. Attacks on XML Encryption
 4. Attacks on Symmetric Encryption Scheme
 1. Attack Scenario
 2. Plaintext Validity
 3. Using Web Service for Plaintext Validation
 4. Decrypting by Checking Plaintext Validity
 5. Countermeasures and Problems
 6. WS-Attacker
- 

WS-Attacker

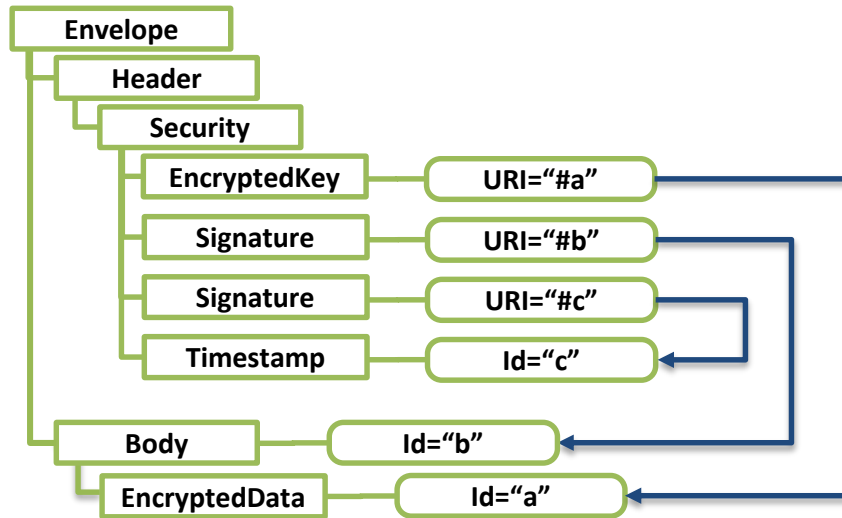
- Automatic penetration test tool for Web Services
- <https://github.com/RUB-NDS/WS-Attacker>
- Supports many attacks
 - New plugin for XML Encryption
- Approach:
 1. Analyze message security
 2. Remove signature protection
 3. Attack (symmetric / asymmetric)



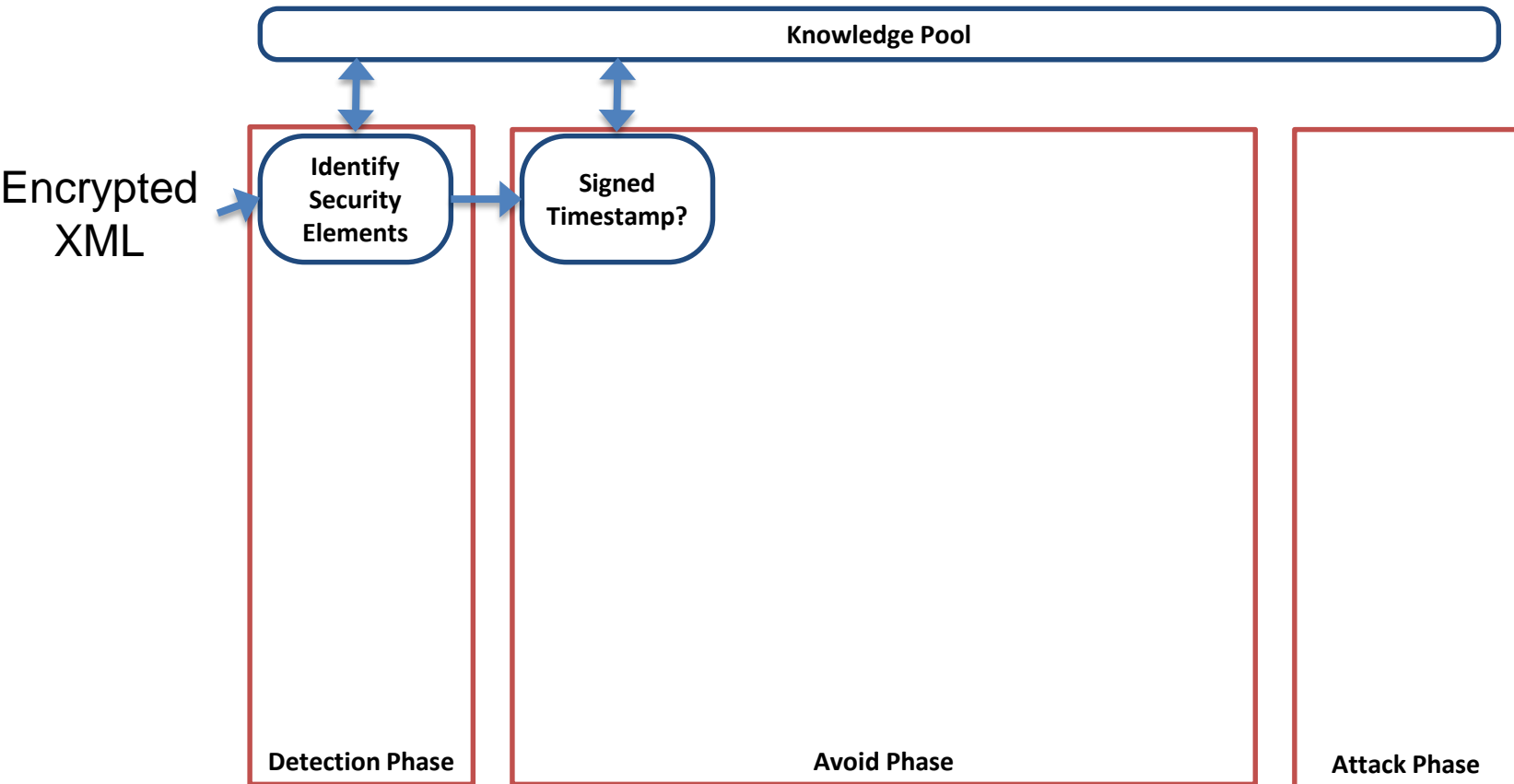
Automated Attack Workflow



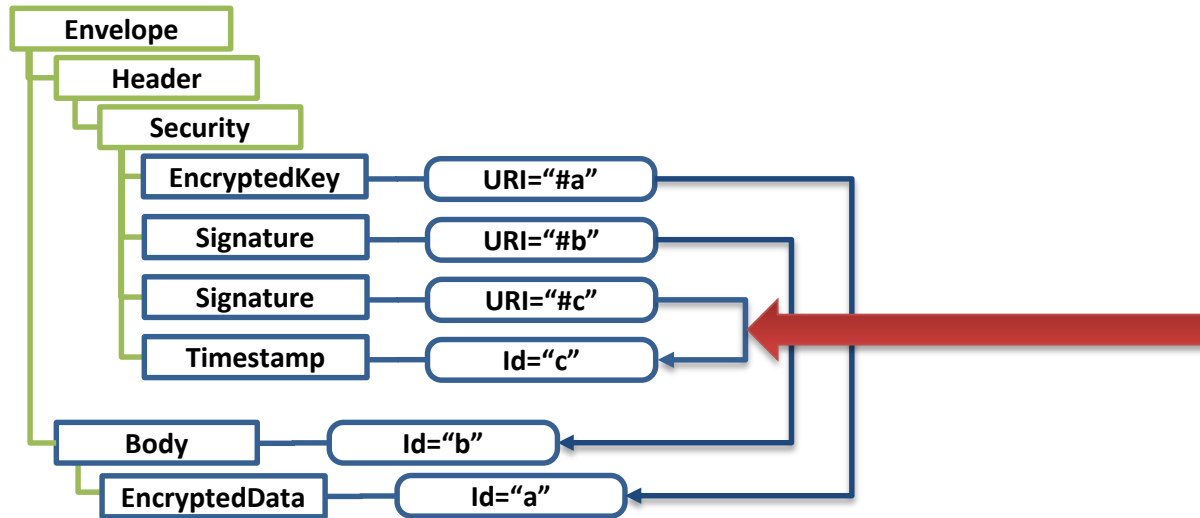
Detection Phase (Offline)



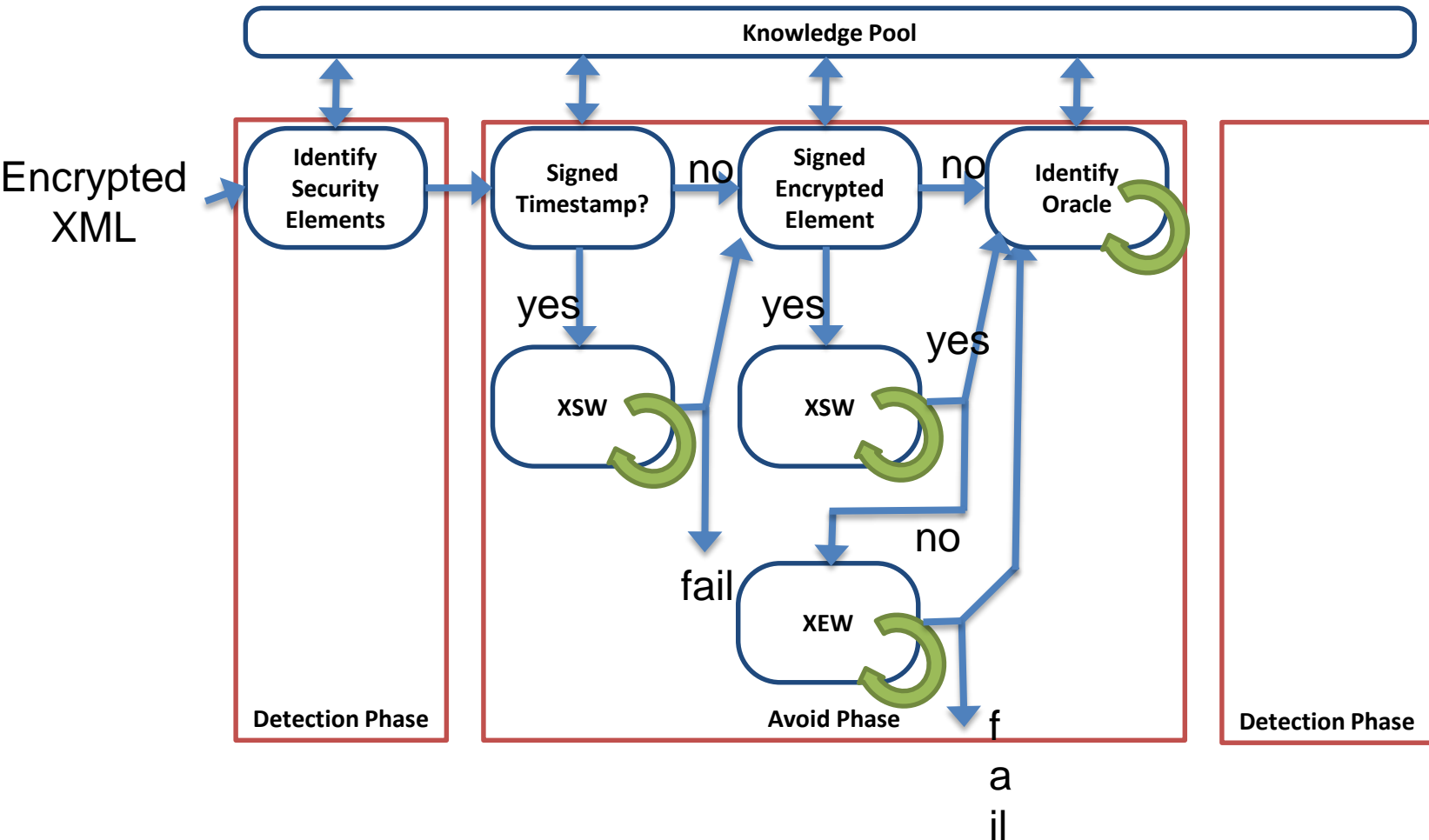
Automated Attack Workflow



Detection Phase (Offline)

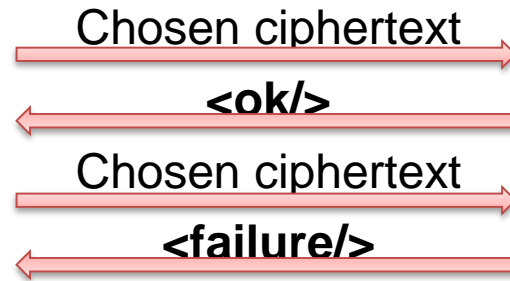


Automated Attack Workflow



Identify Oracle

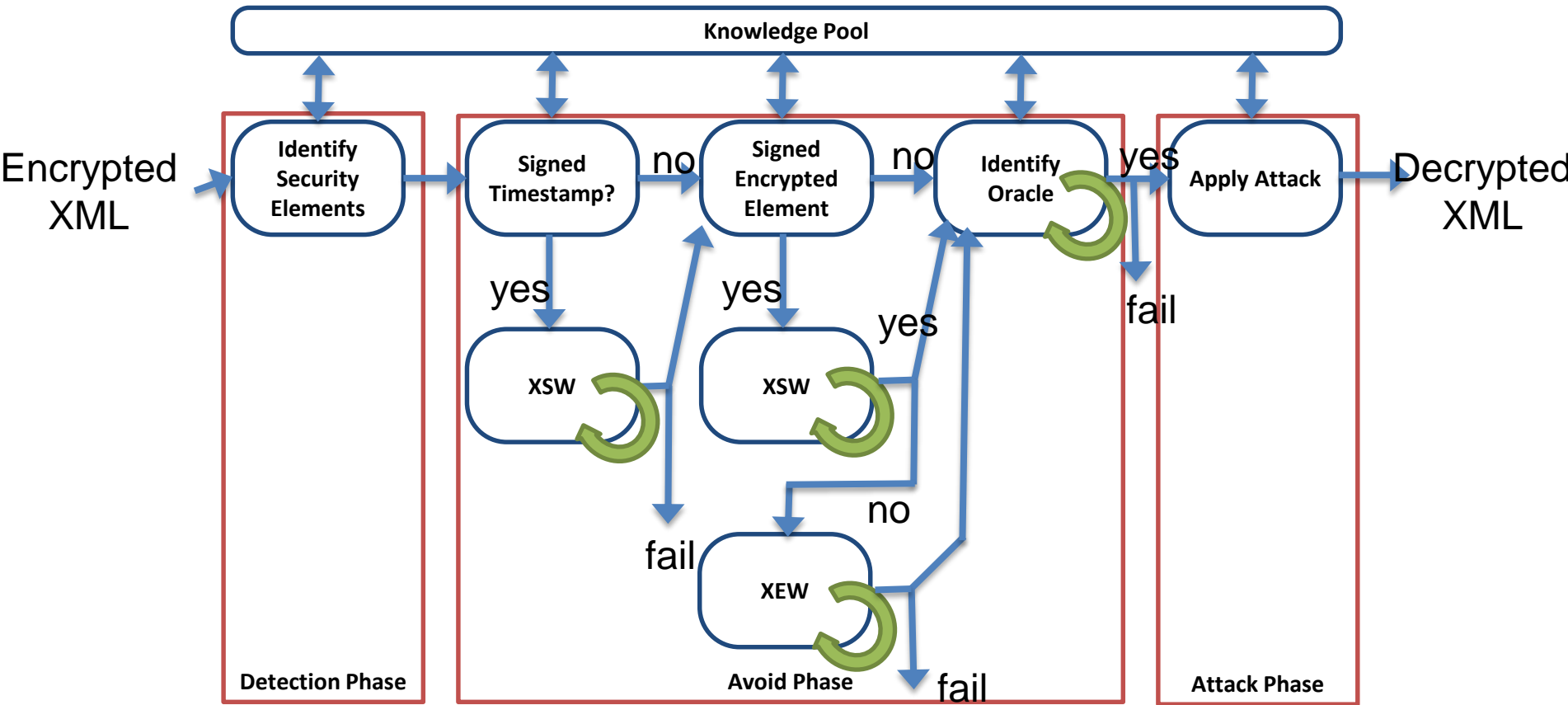
- Map Server Responses to „valid“ or „invalid“



Web Service

- Implementation dependent!

Automated Attack Workflow

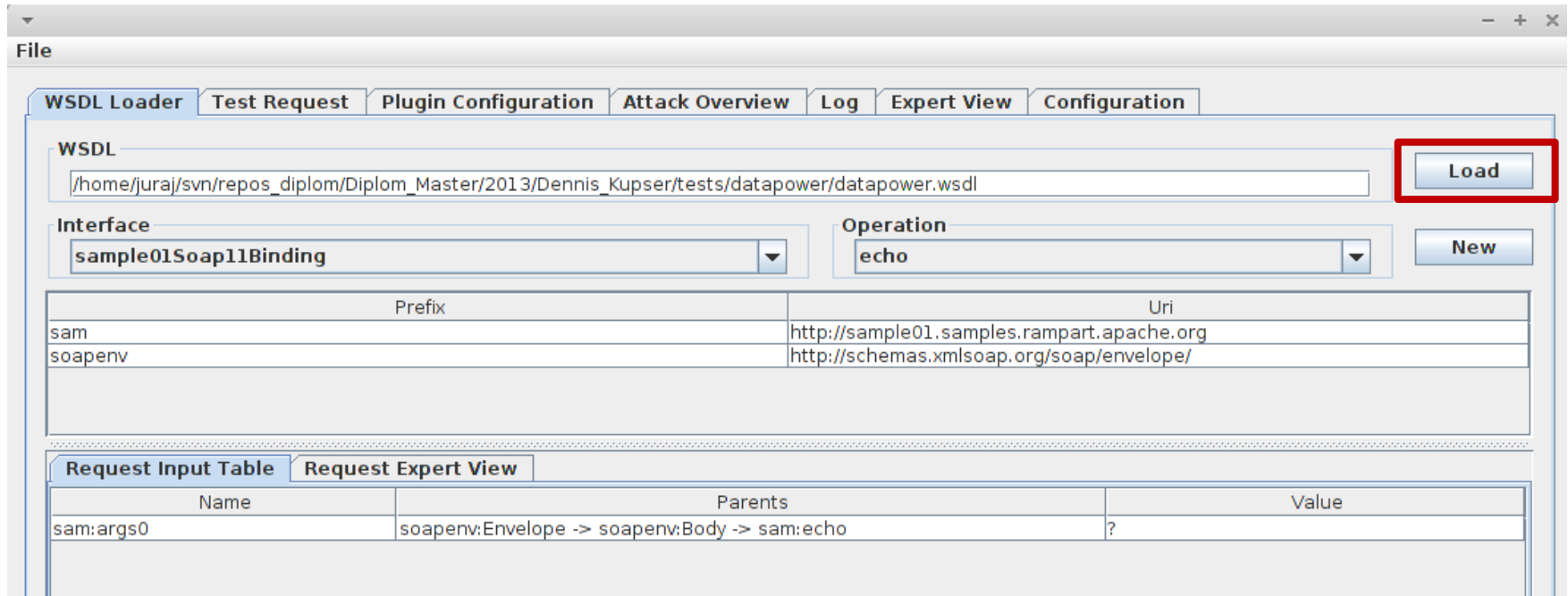


Results

System	Asymmetric Attack	Symmetric Attack	Countermeasures applicable?
Apache Axis2 1.6.2		✓	
Apache CXF 2.7.10	✓	✓	yes
Axway Gateway 7.3.1	✓	✓	yes
IBM Datapower XI50		✓	yes
Microsoft WCF			yes



Load WSDL



File

WSDL Loader | Test Request | Plugin Configuration | Attack Overview | Log | Expert View | Configuration

WSDL

/home/juraj/svn/repos_diplom/Diplom_Master/2013/Dennis_Kupser/tests/datapower/datapower.wsdl **Load**

Interface: sample01Soap11Binding | Operation: echo **New**

Prefix	Uri
sam	http://sample01.samples.rampart.apache.org
soapenv	http://schemas.xmlsoap.org/soap/envelope/

Request Input Table | Request Expert View

Name	Parents	Value
sam:args0	soapenv:Envelope -> soapenv:Body -> sam:echo	?

Send Test Request

The screenshot shows a web security tool interface with a 'File' menu and several tabs: 'WSDL Loader', 'Test Request', 'Plugin Configuration', 'Attack Overview', 'Log', 'Expert View', and 'Configuration'. The 'Test Request' tab is active, showing a 'URL Endpoint' of 'http://' and a path of '/decrypt'. Below this, there are two sub-tabs: 'XML Request' and 'Additional HTTP Request Headers'. The 'XML Request' sub-tab is active, displaying a SOAP request XML document. The response area below shows 'XML Response' and 'HTTP Response Headers' tabs, with the 'XML Response' sub-tab active, displaying a partial SOAP response XML document.

XML Request

```

1 | <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sam="http://ru
2 |   <soapenv:Header>
3 |     <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/
4 |       <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
5 |         <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" xmlns:ds
6 |         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
7 |           <wsse:SecurityTokenReference>
8 |             <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-20
9 |           </wsse:SecurityTokenReference>
10 |         </dsig:KeyInfo>
11 |         <xenc:CipherData xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
12 |           <xenc:CipherValue>pTm4u6vay+exrUdf33Z59UZpnqyR0tgYnTqRrffOGTkcahdbaJVliiliOmFKlk6c
13 |         </xenc:CipherData>
14 |         <xenc:ReferenceList>
15 |           <xenc:DataReference URI="#body" />
16 |         </xenc:ReferenceList>
17 |       </xenc:EncryptedKey>
18 |     </wsse:Security>

```

XML Response

```

1 | <soapenv:Envelope xmlns:sam="http://rub.de/hgi-nds" xmlns:soapenv="http://schemas.xmlsoap.org/sc
2 |   <soapenv:Header>
3 |     <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/
4 |   </soapenv:Header>

```

Choose Attacks

The screenshot shows a web security tool interface with the following components:

- File Menu:** Located at the top left.
- Navigation Tabs:** WSDL Loader, Test Request, **Plugin Config**, Attack Overview, Log, Expert View.
- Left Sidebar (Tree View):**
 - Active Plugins (1)
 - All Plugins (10)
 - Denial of Service (6)
 - Coercive Parsing (Ready)
 - Hash DOS Attack (Ready)
 - Oversized XML (Ready)
 - SOAP array attack (Ready)
 - Test DOS Attack (Ready)
 - Xml Entity Expansion (recursive) (Ready)
 - Security (1)
 - Signature (1)
 - Signature Wrapping (Ready)**
 - Spoofing Attacks (2)
 - SOAPAction Spoofing (Ready)
 - WS-Addressing Spoofing (Ready)
 - Test (1)
 - Alphabetical Sorted (10)

- Main Panel: Signature Wrapping**
- Schema?** Turn on, to not use any XML Schema. Browse...
- Used Schema files** *Set the Schema Files. Soap11, Soap12, WSA, WSSE, WSU, DS and XPathFilter2 are included by default.*
- Search?** SOAP Response must contain a specific String. View
- View** *Display the wrapping messages.*
- Show:** Payload #1
- Reference Element:**

```
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" xmlns:wsa="http://www.w3.org/2001/10/xml-exc-c14n#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <ds:Reference URI="#_13c5cf5aaaa9075df9872ed43d69f936" />
    <ds:Transforms>
      <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
      <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" /></ds:Transforms>
  </ds:SignedInfo>
  <ds:Signature Value="..." />
</Assertion>
```
- Timestamp?**
 - URI="#_13c5cf5aaaa9075df9872ed43d69f936"
- Analyzing:** `//*[@AssertionID='_13c5cf5aaaa9075df9872ed43d69f936']`
- Buttons:** All, None, Save, Load
- Status Bar:** [INFO] Has payload? false

Configure XML Encryption

Detected Encrypted Elements:

Elements: EncryptedKey TimeStamp

Configuration:

Attack: Oracle Type:

Wrapping Attack: StringCompare:

Threshold Wrap Error: Threshold General Error:

EncryptedKey:

isAttackPayload
 isSigned
 isAddWrap

PKCS1 Strategy:

```

"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <xenc:EncryptionMethod xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Algorithm=
"http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
  <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier EncodingType=

```

EncryptedData:

1/1

isAttackPayload
 isSigned
 isAddWrap
 useTypeWeakness

```

  <xenc:CipherData>
    <xenc:CipherValue>
jXvVRqnTrgKaBip88NTWuyqUkupHAhy6vkdwCDgzQ1+Bwf5gT72pF1WZ2s39aBdjQoWkLukjkSL/341yLxwWk2JqTZhJ3cZH4lou9XwnauNT1caIm4wriCoLcGQySfAFcfC6QrNQ29cJiFxpTJTtLogTJxJz6tWz+/hPnym8vg/2LwzK607dJj3AmU1tq43Nfg3mW6y2o5/1SEhvoXpRXbw==</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>

```

Start Attack

File

WSDL Loader | Test Request | Plugin Configuration | **Attack Overview** | Log | Expert View | Configuration

Start | Stop | Clean | Save

Critical | Important | Info | Trace

Name	Status	Rating	Vulnerable?
XML-Encryption Attack	Finished	100%	YES

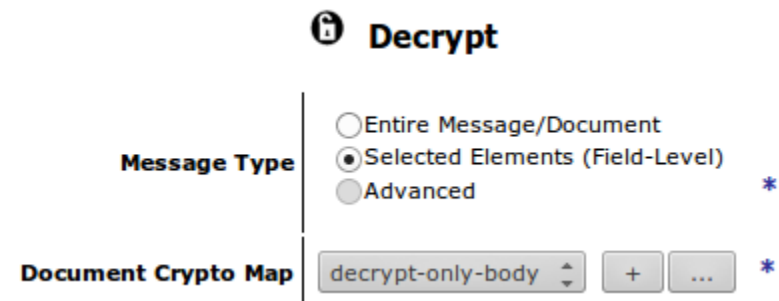
Time	Level	Source	Content
13:39:16.775	Important	XML-Encryption Attack	Setting avoided Document
13:42:58.252	Info	XML-Encryption Attack	Starting CBC_ATTACK
13:51:37.529	Critical	XML-Encryption Attack	Plaintext of encrypted data: <sam:echo xmlns="" xmlns:sam="http://sample01.samples.rampart.apache.org" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"> <!--Optional:--> <sam:args0?</sam:args0> </sam:echo>
13:51:37.529	Info	XML-Encryption Attack	Number of Oracle Queries: 4855
			Bytes decrypted: 221

Playing with WS-Attacker

- <https://github.com/RUB-NDS/WS-Attacker>
- Use Apache Axis2 and Apache Rampart
- Examples:
 - <http://web-in-security.blogspot.de/>

Countermeasures

- AES-CBC, RSA-PKCS#1 v1.5 insecure!
- XML Encryption updated (Version 1.1)
 - AES-GCM added
- Use of secure algorithms: RSA-OAEP, AES-GCM
- If secure algorithms not available, only decrypt signed XML ciphertexts
 - Example: IBM Datapower



Conclusion

- XML – especially XML Security – is complex
- WS-Attacker for evaluation:
 - <https://github.com/RUB-NDS/WS-Attacker>
- Our approach applicable to other scenarios
 - SAML, JSON, Web Crypto...
- Prefer authenticated encryption (AES-GCM instead of AES-CBC)

