How To Break XML Signature and XML Encryption

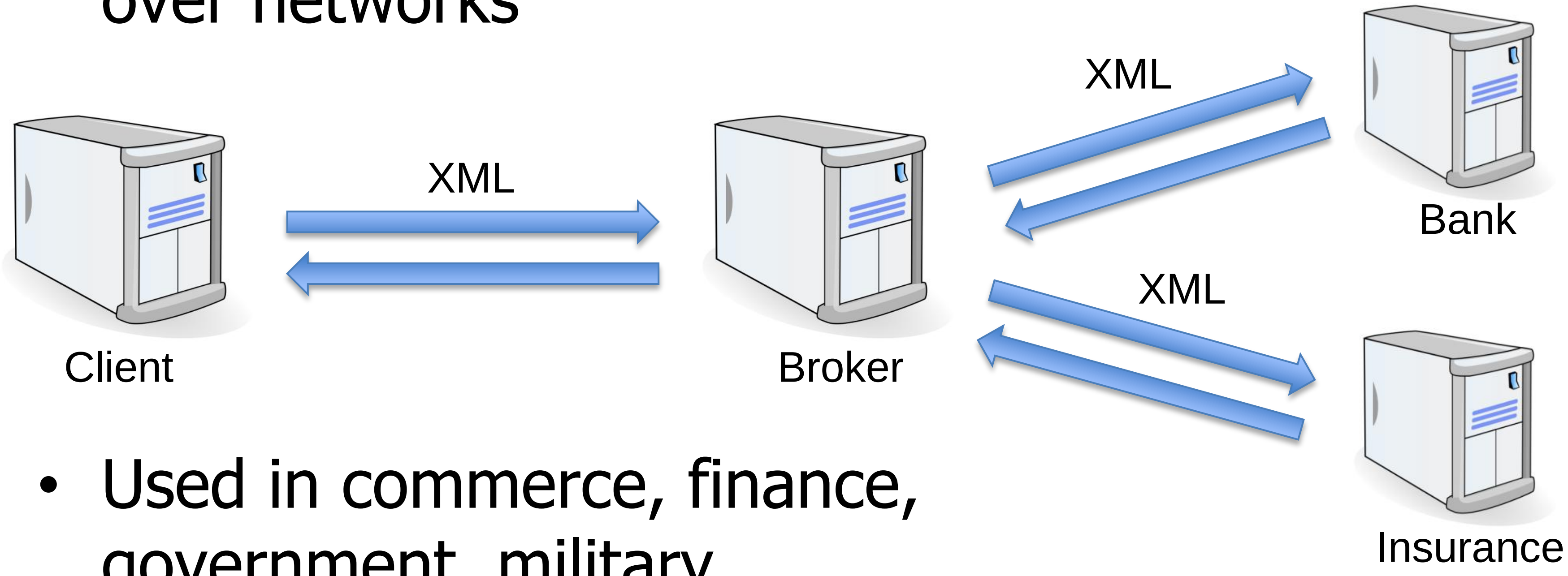
Juraj Somorovsky

Horst-Görtz Institute
Ruhr-University of Bochum
Juraj.somorovsky@rub.de

OWASP
17.11.2011

Motivation – Web Services

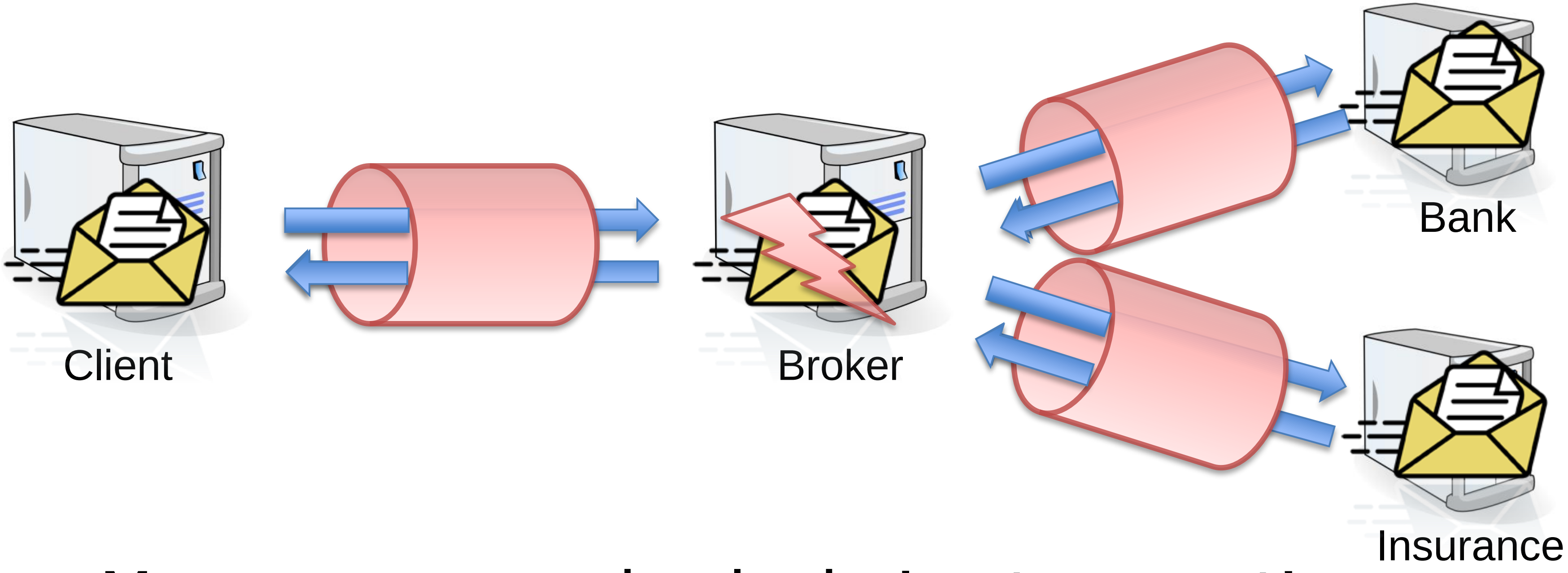
- Method for machine-to-machine communication over networks



- Used in commerce, finance, government, military, …
- XML-based message format

Motivation – XML Security

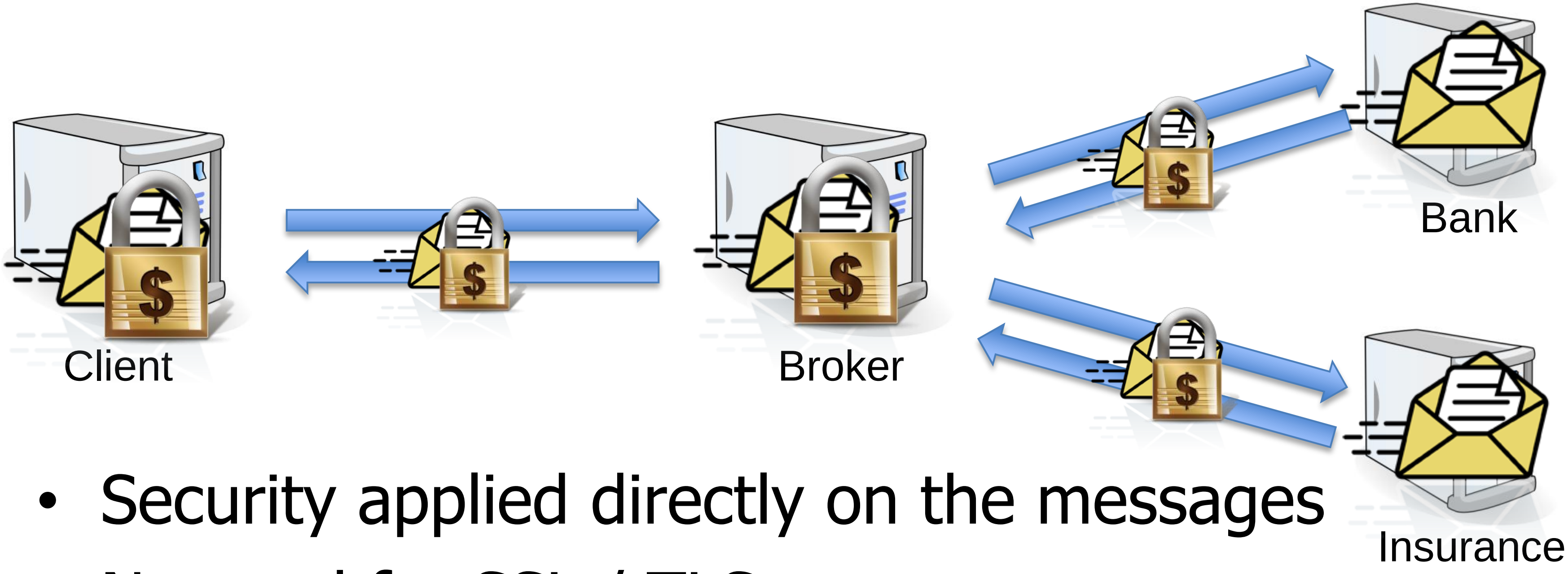
- SSL / TLS: transport-level security



- Messages secured only during transport!

Motivation – XML Security

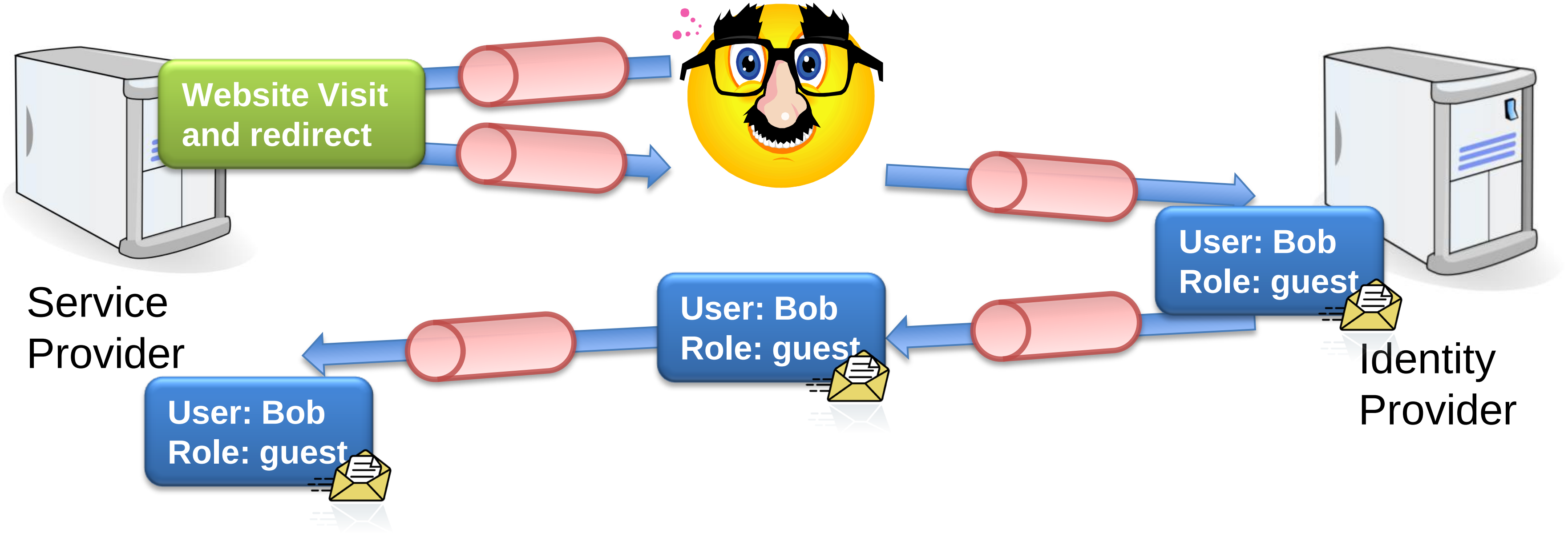
- Message level security



- Security applied directly on the messages
- No need for SSL / TLS
- Realized using XML Signature, XML Encryption

Motivation – XML Security

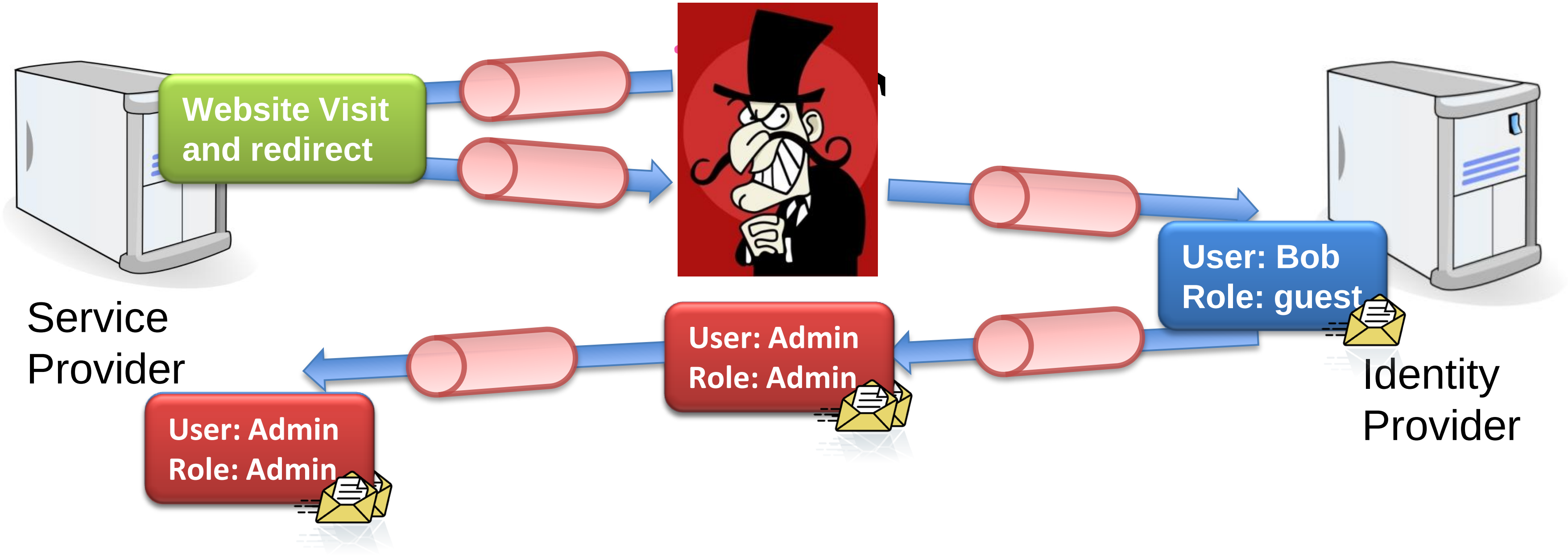
- Another example: Browser-based Single Sign-On



- Messages secured only during transport!

Motivation – XML Security

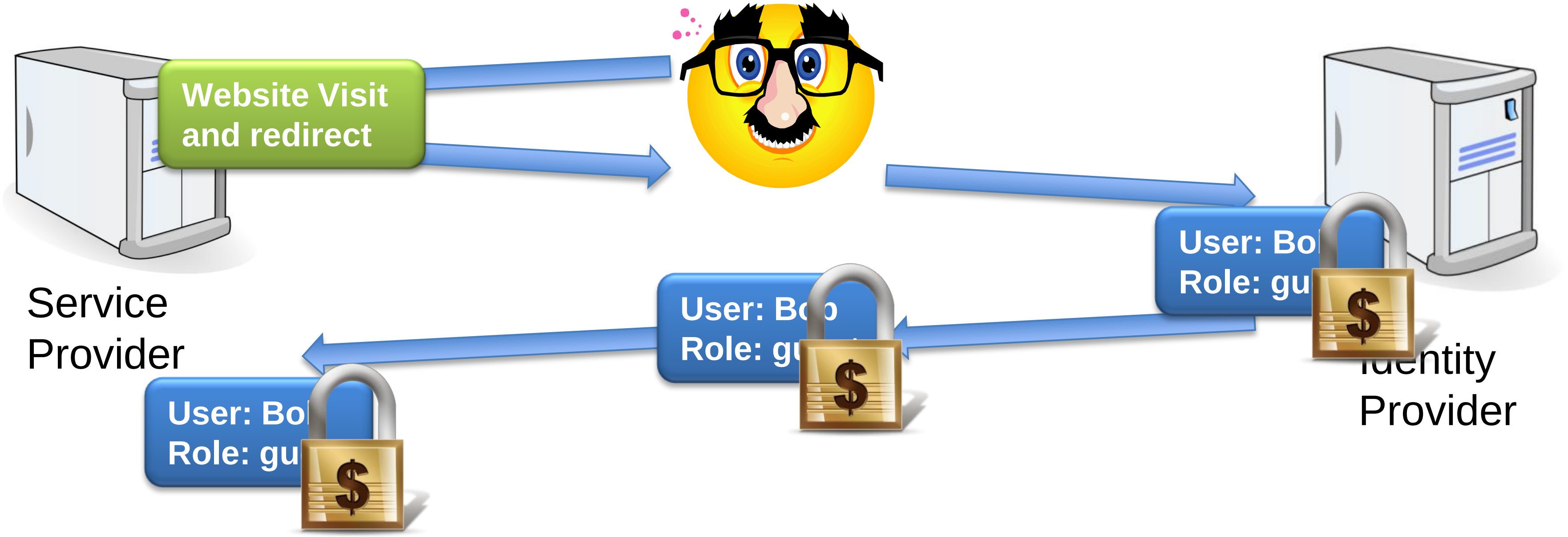
- Does SSL / TLS help?



- Need for message level security!

Motivation – XML Security

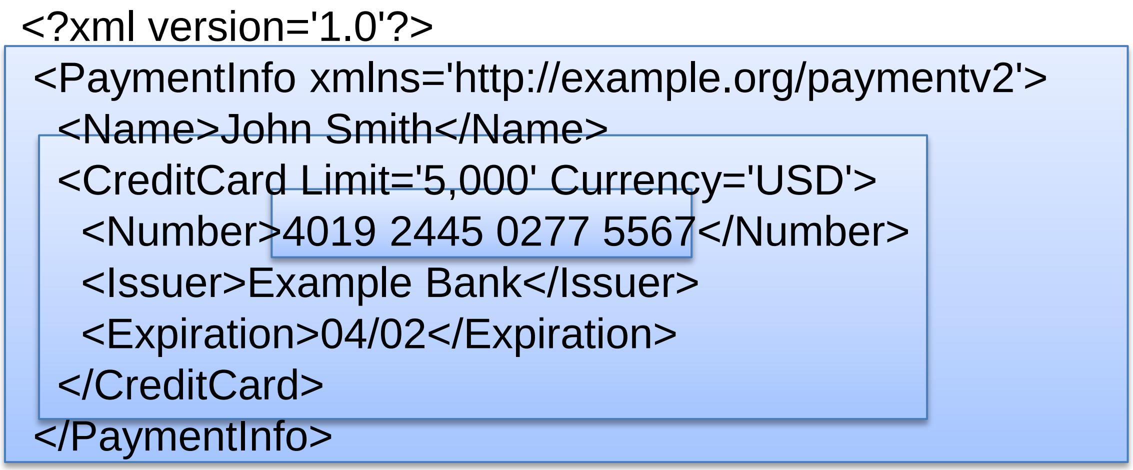
- Another example: Browser-based Single Sign-On



- Could be realized using XML Signature and XML Encryption

Motivation – XML Security

- W3C Standards: XML Signature and XML Encryption
- Describe various methods for applying cryptographic algorithms to XML documents



```
<?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>Example Bank</Issuer>
   <Expiration>04/02</Expiration>
  </CreditCard>
 </PaymentInfo>
```

Overview

1. Breaking XML Signature
    - Cloud Computing Management Interfaces
    - Amazon EC2 SOAP Interface
    - XML Signature Wrapping on Eucalyptus and Amazon
    - Countermeasures and Conclusion
2. Breaking XML Encryption
    - Attack Scenario
    - Decrypting by checking plaintext validity
    - Application to CBC mode of operation in XML Encryption
    - Countermeasures and Conclusion
3. Conclusion

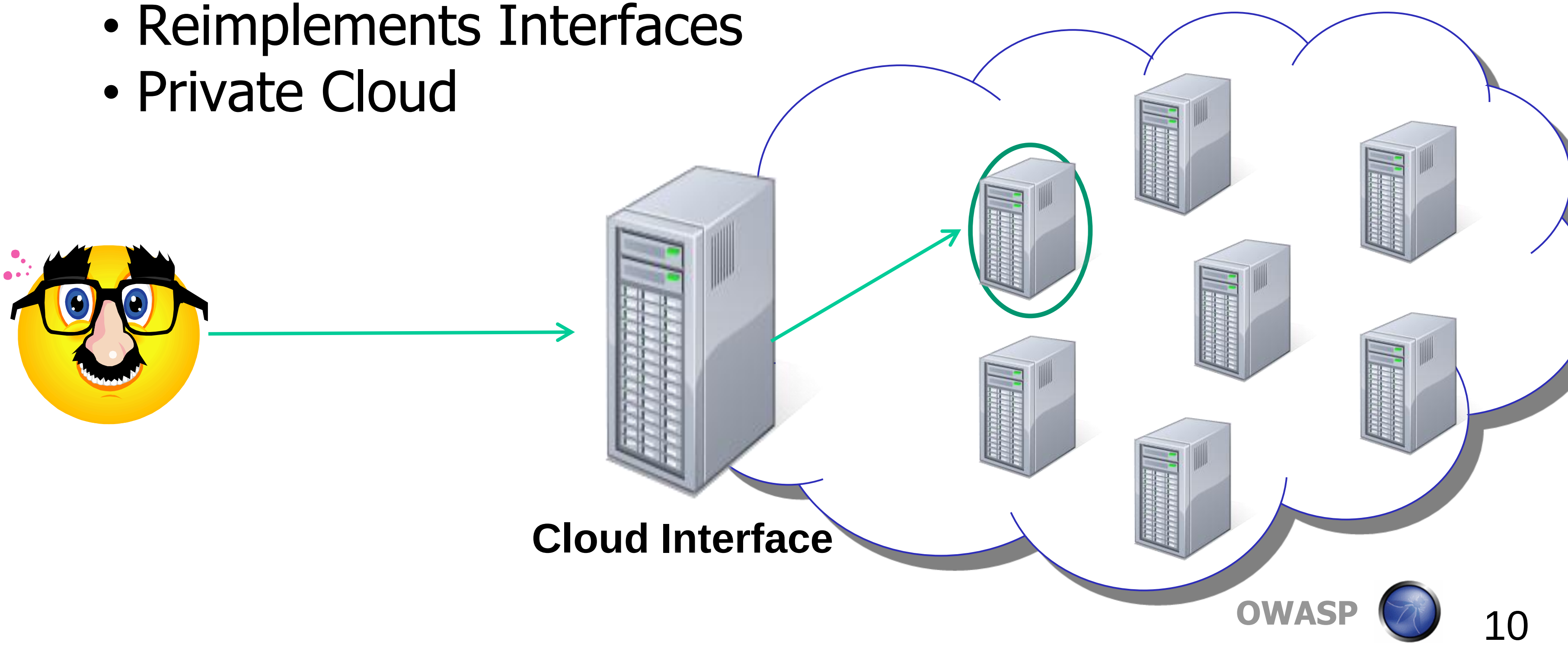
Juraj Somorovsky, Mario Heiderich, Meiko Jensen, Jörg Schwenk, Nils Gruschka, Luigi Lo Iacono: **All Your Clouds Are Belong to Us – Security Analysis of Cloud Management Interfaces** - In Proceedings of the ACM Cloud Computing Security Workshop (CCSW), 2011.

Tibor Jager, Juraj Somorovsky: **How To Break XML Encryption** - In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), 2011.

Cloud Computing Management Interfaces

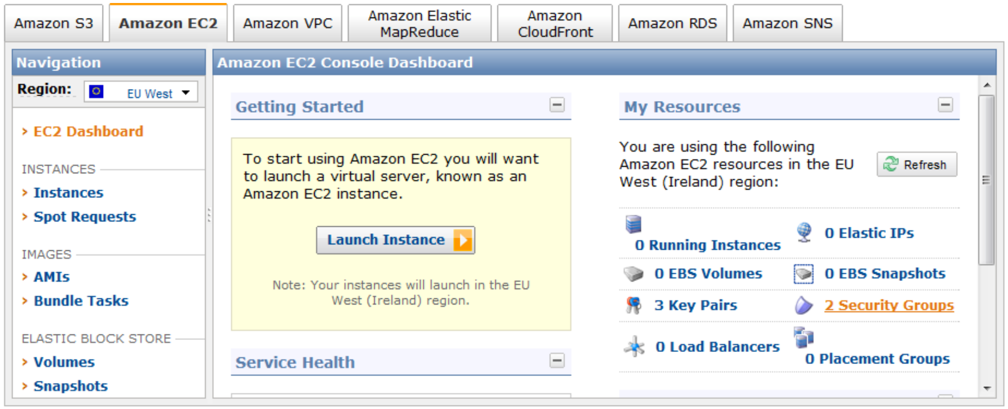
Cloud Computing:

- Amazon Web Services (2006):
  - Public Cloud
- Eucalyptus Cloud (2009):
  - Reimplements Interfaces
  - Private Cloud



Cloud Computing Management Interfaces

- Controlling of the cloud using different interfaces



To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the EU West (Ireland) region.

You are using the following Amazon EC2 resources in the EU West (Ireland) region:

SOAP

REST

Overview

1. **Breaking XML Signature**
    - **Cloud Computing Management Interfaces**
    - **Amazon EC2 SOAP Interface**
    - **XML Signature Wrapping on Eucalyptus and Amazon**
    - **Countermeasures and Conclusion**
2. **Breaking XML Encryption**
    - **Attack Scenario**
    - **Decrypting by checking plaintext validity**
    - **Application to CBC mode of operation in XML Encryption**
    - **Countermeasures and Conclusion**
3. **Conclusion**

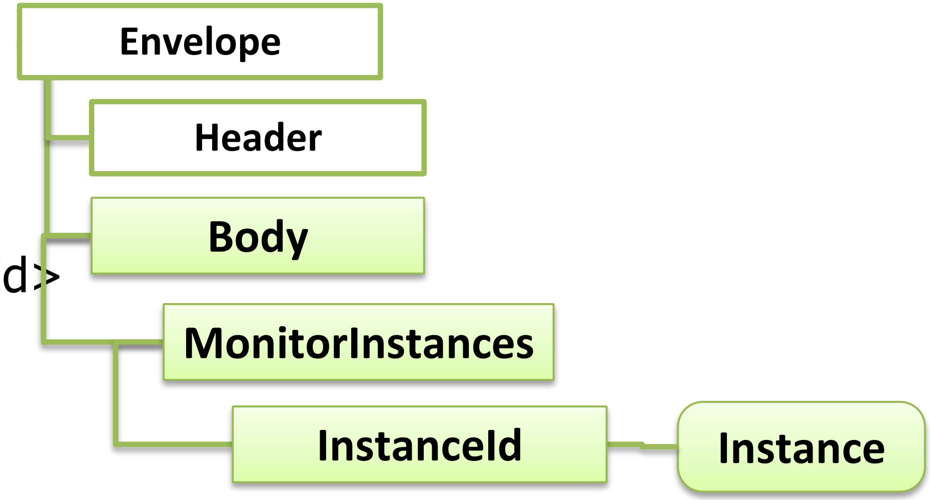
Juraj Somorovsky, Mario Heiderich, Meiko Jensen, Jörg Schwenk, Nils Gruschka, Luigi Lo Iacono: **All Your Clouds Are Belong to Us – Security Analysis of Cloud Management Interfaces** - In Proceedings of the ACM Cloud Computing Security Workshop (CCSW), 2011.

Tibor Jager, Juraj Somorovsky: **How To Break XML Encryption** - In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), 2011.

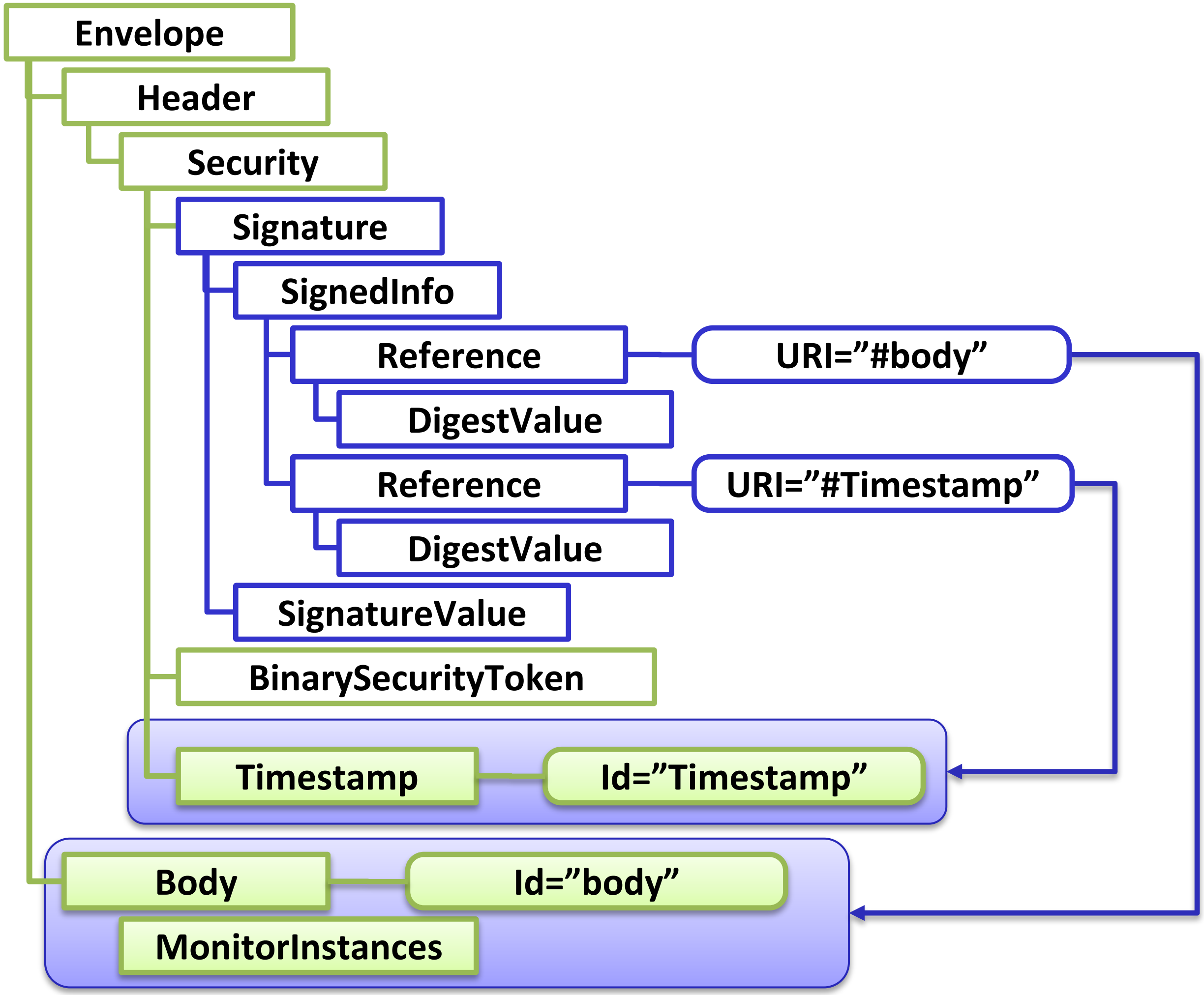
Amazon EC2 SOAP Interface

Monitoring of running instances:

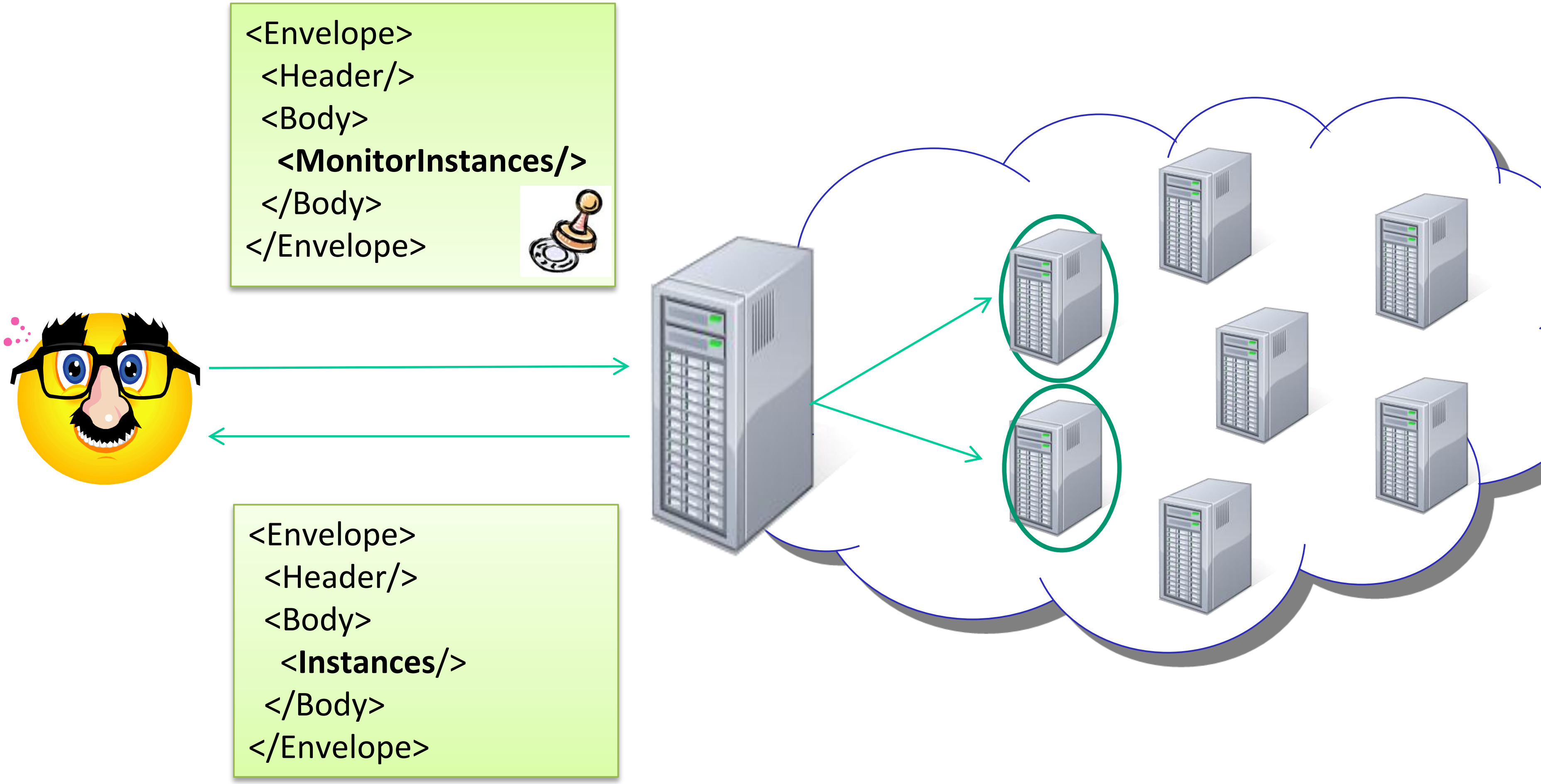
```
<Envelope>
  <Header />
  <Body>
    <MonitorInstances>
      <InstanceId> instance </InstanceId>
    </MonitorInstances>
  </Body>
</Envelope>
```



Amazon EC2 SOAP Interface - XML Signature



Amazon EC2 SOAP Interface - XML Signature



```
<Envelope>
  <Header/>
  <Body>
    <MonitorInstances/>
  </Body>
</Envelope>
```

```
<Envelope>
  <Header/>
  <Body>
    <Instances/>
  </Body>
</Envelope>
```

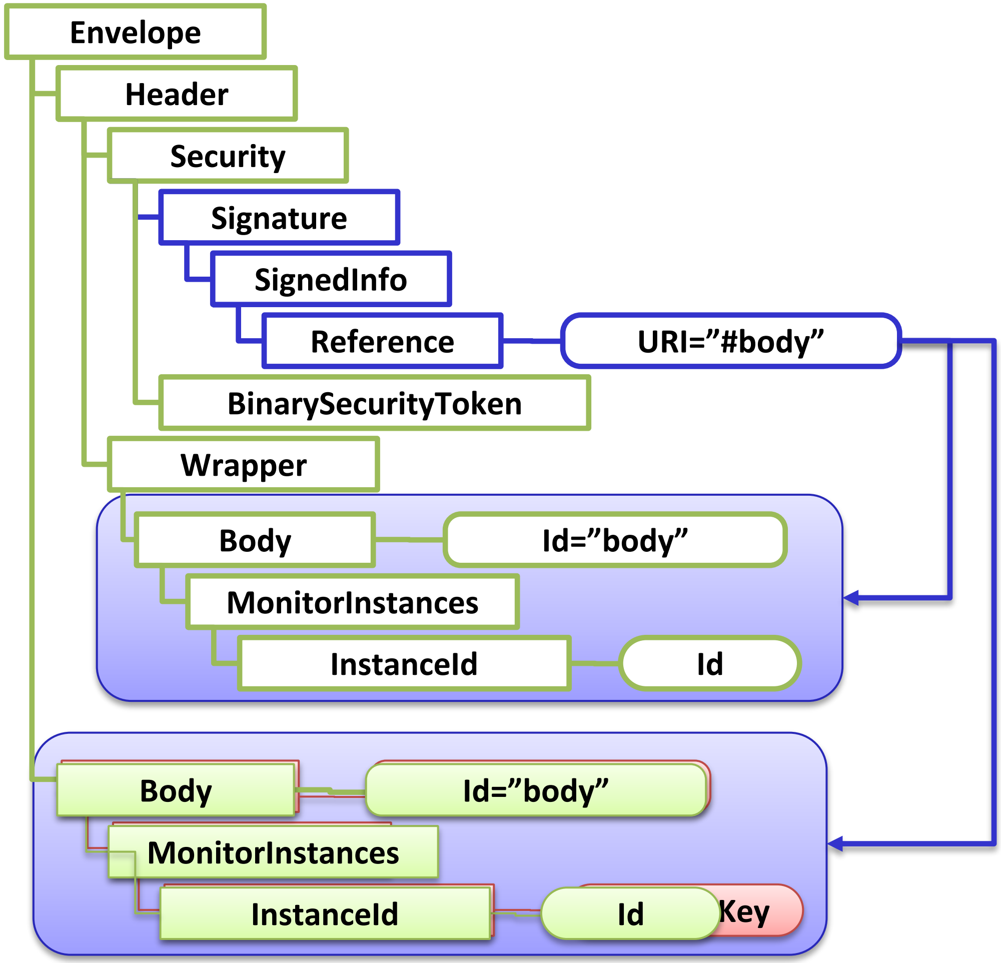
Overview

1. **Breaking XML Signature**
   - **Cloud Computing Management Interfaces**
   - **Amazon EC2 SOAP Interface**
   - **XML Signature Wrapping on Eucalyptus and Amazon**
   - **Countermeasures and Conclusion**
2. **Breaking XML Encryption**
   - **Attack Scenario**
   - **Decrypting by checking plaintext validity**
   - **Application to CBC mode of operation in XML Encryption**
   - **Countermeasures and Conclusion**
3. **Conclusion**

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** - In Proceedings of the ACM Cloud Computing Security Workshop (CCSW), 2011.

Tibor Jager, Juraj Somorovsky: **How To Break XML Encryption** - In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), 2011.

XML Signature Wrapping



The same attack on the Timestamp!

McIntosh and Austel. XML Signature Element Wrapping attacks, 2005

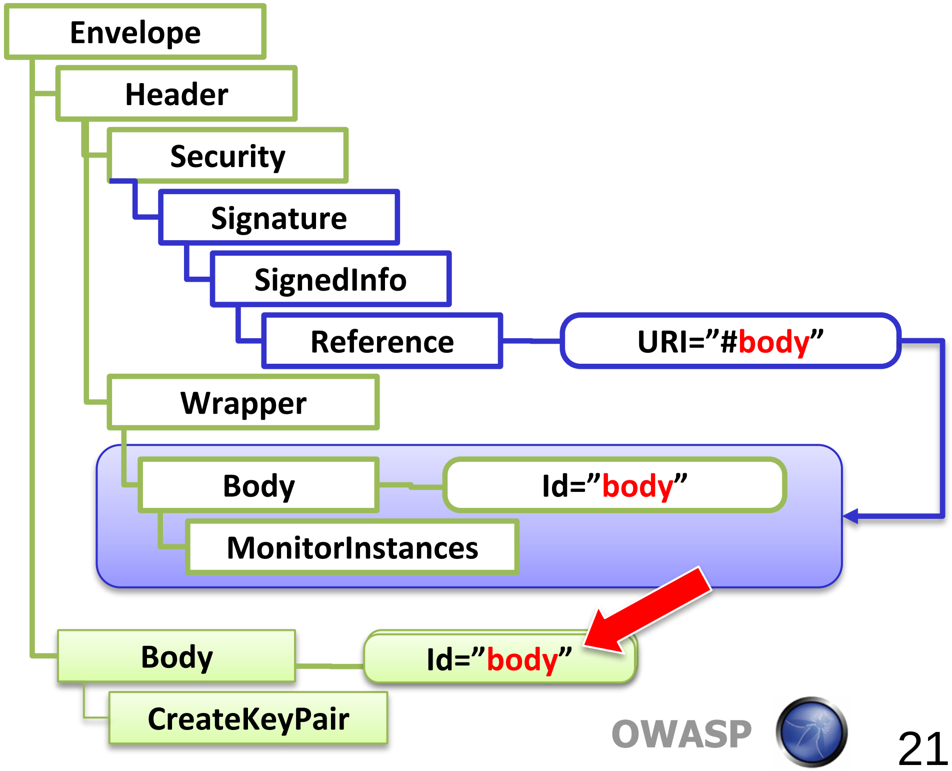
XML Signature Wrapping on Eucalyptus

- Attack by McIntosh and Austel directly applicable
- Eucalyptus: Open Source
- Reason: Problem in Apache Axis Web Services Framework

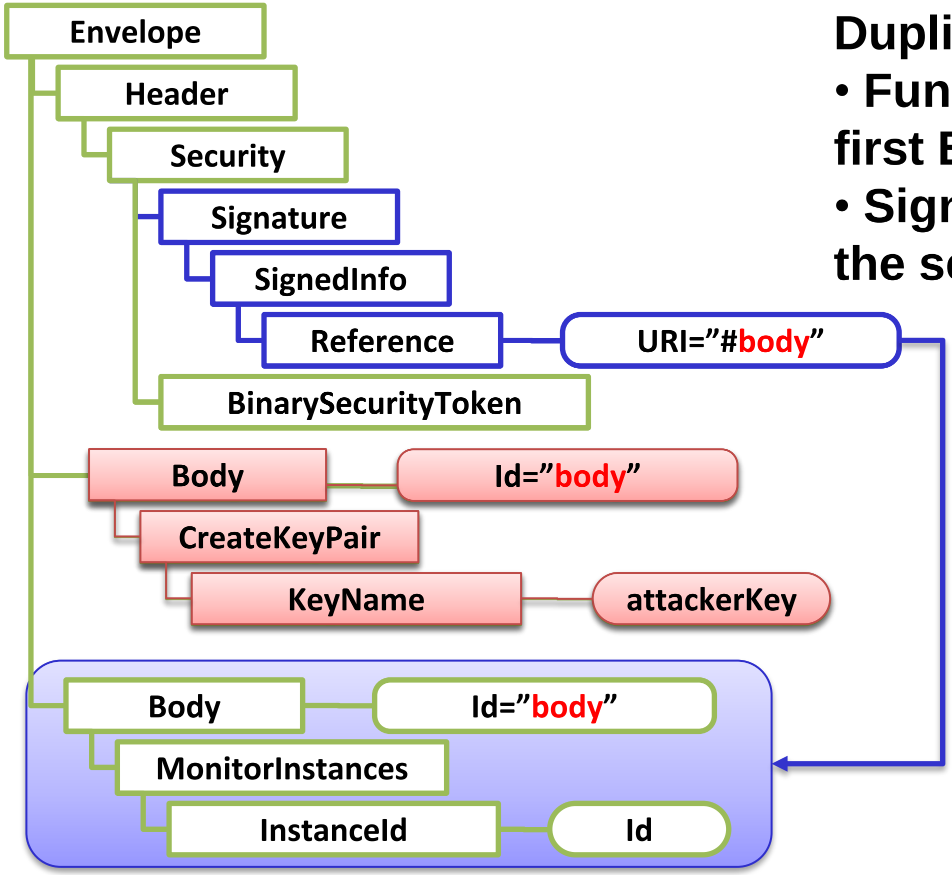
- CVE-2011-0730

XML Signature Wrapping on Amazon

- Attack from McIntosh & Austel does not work
- Amazon checks, if the Id of the signed element equals to the Id of the processed element
- But what happens if we use two elements with the same Id?
- Which element is used for signature validation?
- Which for function execution?



XML Signature Wrapping on Amazon



Duplicate the Body element:

- Function invocation from the first Body element
- Signature verification over the second Body element

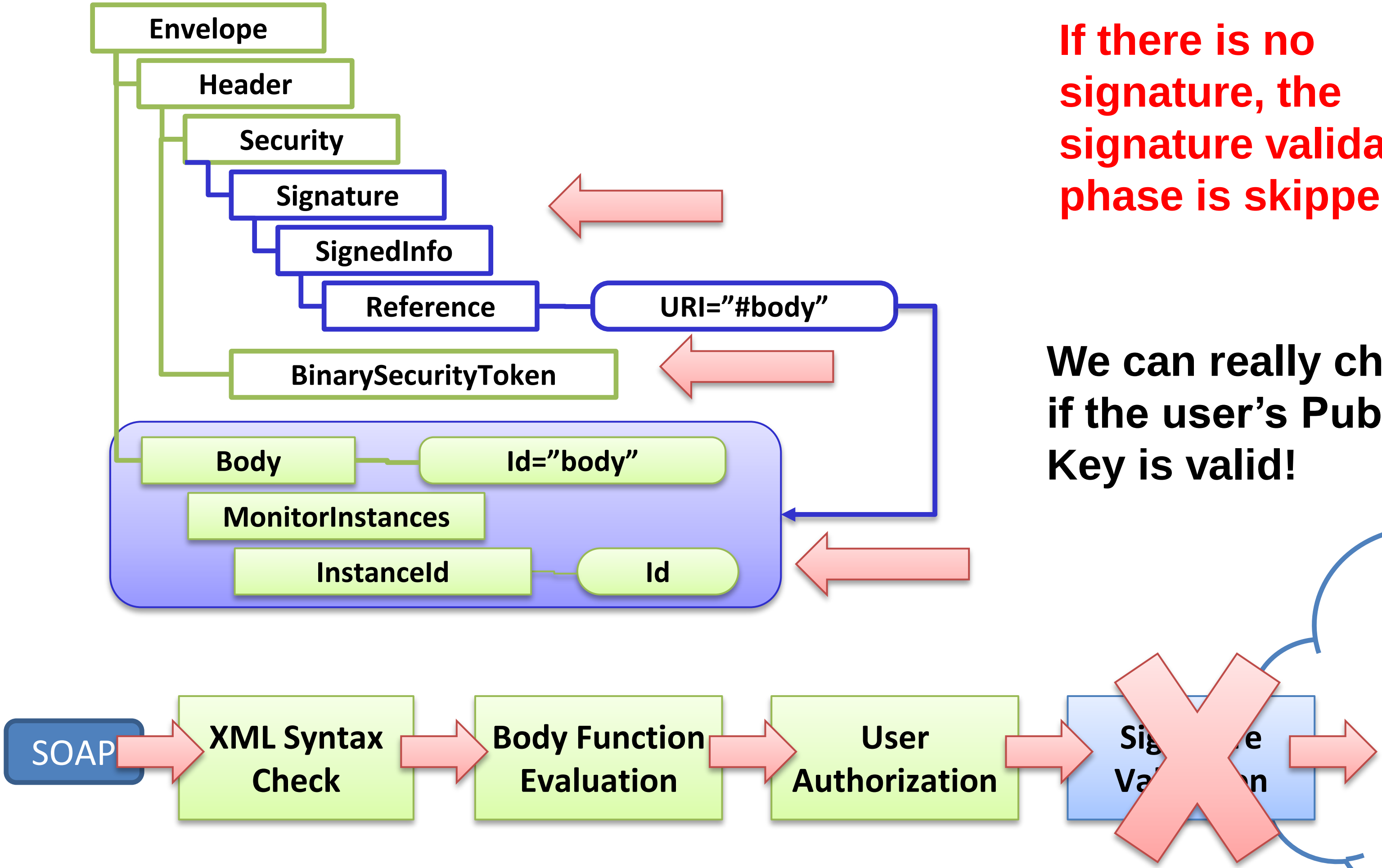
The same attack on the Timestamp!

XML Signature Wrapping on Amazon - Analysis

- Amazon: No Open Source
- Analysis using the SOAP error messages
  - The timestamp has expired
  - The timestamp or body was not signed
  - The certificate holder could not be authorized
  - The signature was invalid
- SOAP error messages = really good source of information

- We sent different hand-crafted messages to the Amazon EC2 interface

XML Signature Wrapping on Amazon - Analysis

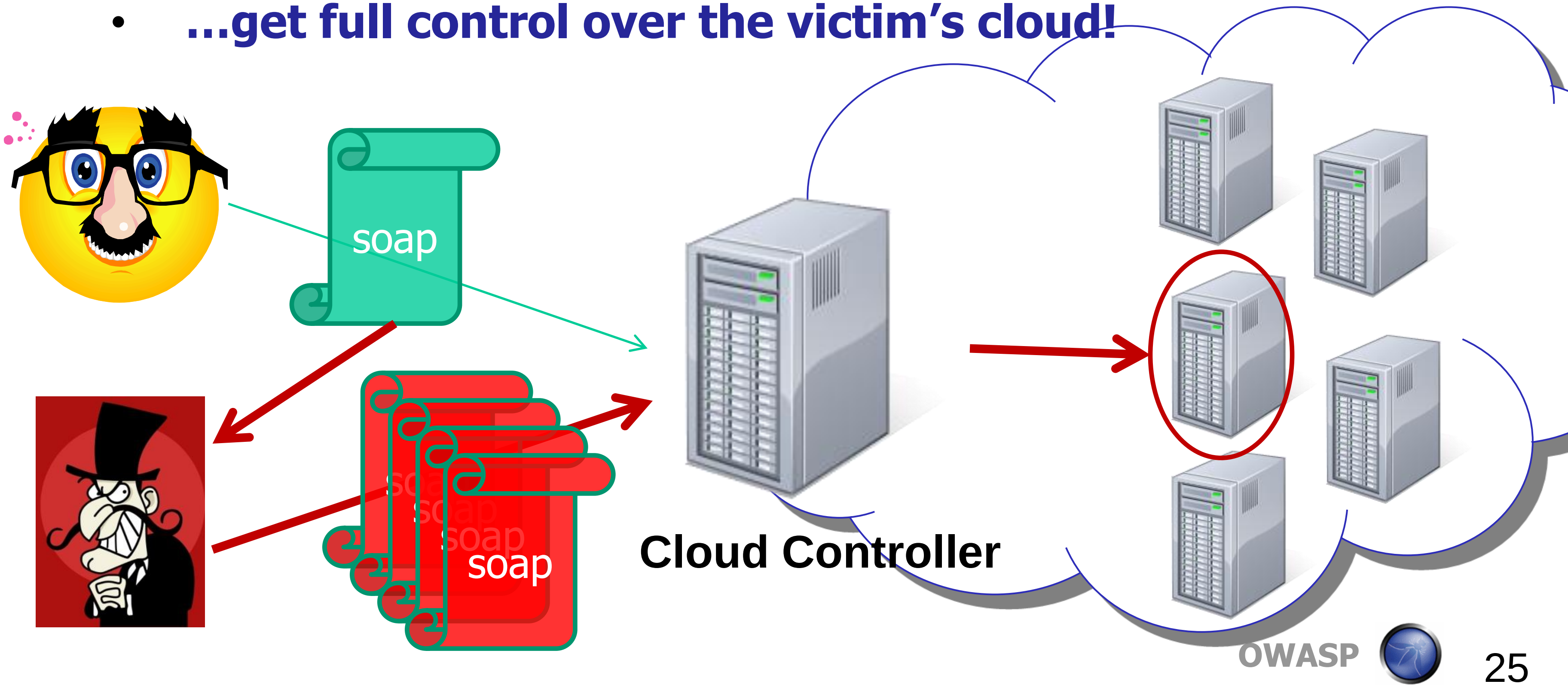


If there is no signature, the signature validation phase is skipped!

We can really check if the user’s Public Key is valid!

XML Signature Wrapping on Amazon - Analysis

- Works only for the Amazon EC2 / Eucalyptus SOAP interface
- One valid SOAP message is enough to:
  - Start and stop cloud instances
  - Download and upload virtual images
  - **...get full control over the victim's cloud!**



Overview

1. **Breaking XML Signature**
    - **Cloud Computing Management Interfaces**
    - **Amazon EC2 SOAP Interface**
    - **XML Signature Wrapping on Eucalyptus and Amazon**
    - **Countermeasures and Conclusion**
2. **Breaking XML Encryption**
    - **Attack Scenario**
    - **Decrypting by checking plaintext validity**
    - **Application to CBC mode of operation in XML Encryption**
    - **Countermeasures and Conclusion**
3. **Conclusion**

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** - In Proceedings of the ACM Cloud Computing Security Workshop (CCSW), 2011.

Tibor Jager, Juraj Somorovsky: **How To Break XML Encryption** - In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), 2011.

XML Signature Wrapping - Conclusion

- We showed practical critical Signature Wrapping attacks on Amazon and Eucalyptus Cloud Interfaces
- All the vulnerabilities have been fixed
- XML Signature Wrapping attacks are known since 2005, but:
  - Are not in focus of research community
  - Nearly all implementations are vulnerable
- Please be aware of Signature Wrapping when applying XML Signatures
  - In Web Services
  - SAML (Single Sign-On)
  - Custom applications
- There are more attacks coming soon

Overview

1. **Breaking XML Signature**
   - **Cloud Computing Management Interfaces**
   - **Amazon EC2 SOAP Interface**
   - **XML Signature Wrapping on Eucalyptus and Amazon**
   - **Countermeasures and Conclusion**
2. **Breaking XML Encryption**
   - **Attack Scenario**
   - **Decrypting by checking plaintext validity**
   - **Application to CBC mode of operation in XML Encryption**
   - **Countermeasures and Conclusion**
3. **Conclusion**

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** - In Proceedings of the ACM Cloud Computing Security Workshop (CCSW), 2011.

Tibor Jager, Juraj Somorovsky: **How To Break XML Encryption** - In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), 2011.

XML Encryption

- W3C standard for encrypting XML data (published in 2002)
- Describes various methods for applying
  - Symmetric ciphers (AES-CBC, 3DES-CBC)
  - Public-key encryption (e.g. RSA-PKCS#1)

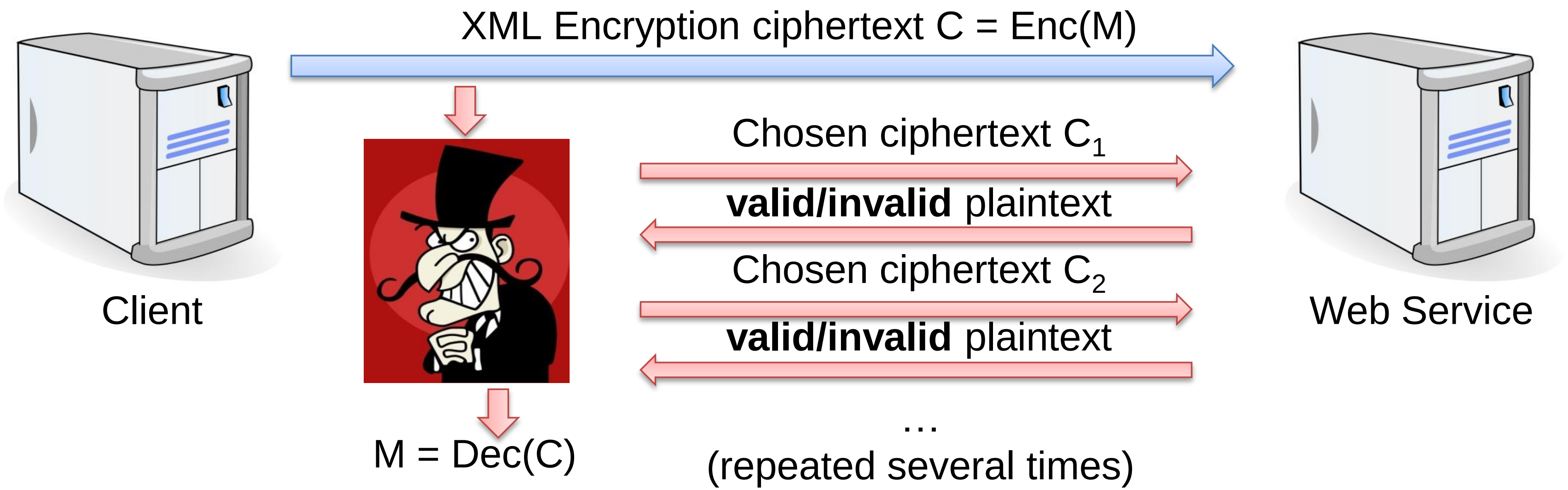
```
<PaymentInfo>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000'>
    <Number>4019 ...5567</Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

```
<PaymentInfo>

  <EncryptedData Id="EncData">
    <EncryptionMethod Algorithm="...xmlenc#aes128-cbc"/>
    …<CipherValue>3bP...Zx0=</CipherValue>…
  </EncryptedData>

</PaymentInfo>
```

XML Encryption

- Attack on XML Encryption
- All major Web Services frameworks vulnerable
    - Apache Axis 2
    - RedHat JBoss
    - IBM WebSphere
    - Microsoft .NET
    - And more (recently discovered)
- Also applicable to XML-based Single Sign-On (recently discovered)

XML Encryption – Attack Scenario



What is a “valid” plaintext?

How to use Web Service as “plaintext validity oracle”?

How to use this oracle to decrypt C?

- How to create useful chosen ciphertexts C_1, C_2, …?

Overview

1. **Breaking XML Signature**
   - **Cloud Computing Management Interfaces**
   - **Amazon EC2 SOAP Interface**
   - **XML Signature Wrapping on Eucalyptus and Amazon**
   - **Countermeasures and Conclusion**
2. **Breaking XML Encryption**
   - **Attack Scenario**
   - **Decrypting by checking plaintext validity**
   - **Application to CBC mode of operation in XML Encryption**
   - **Countermeasures and Conclusion**
3. **Conclusion**

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** - In Proceedings of the ACM Cloud Computing Security Workshop (CCSW), 2011.

Tibor Jager, Juraj Somorovsky: **How To Break XML Encryption** - In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), 2011.

Decrypting by checking plaintext validity

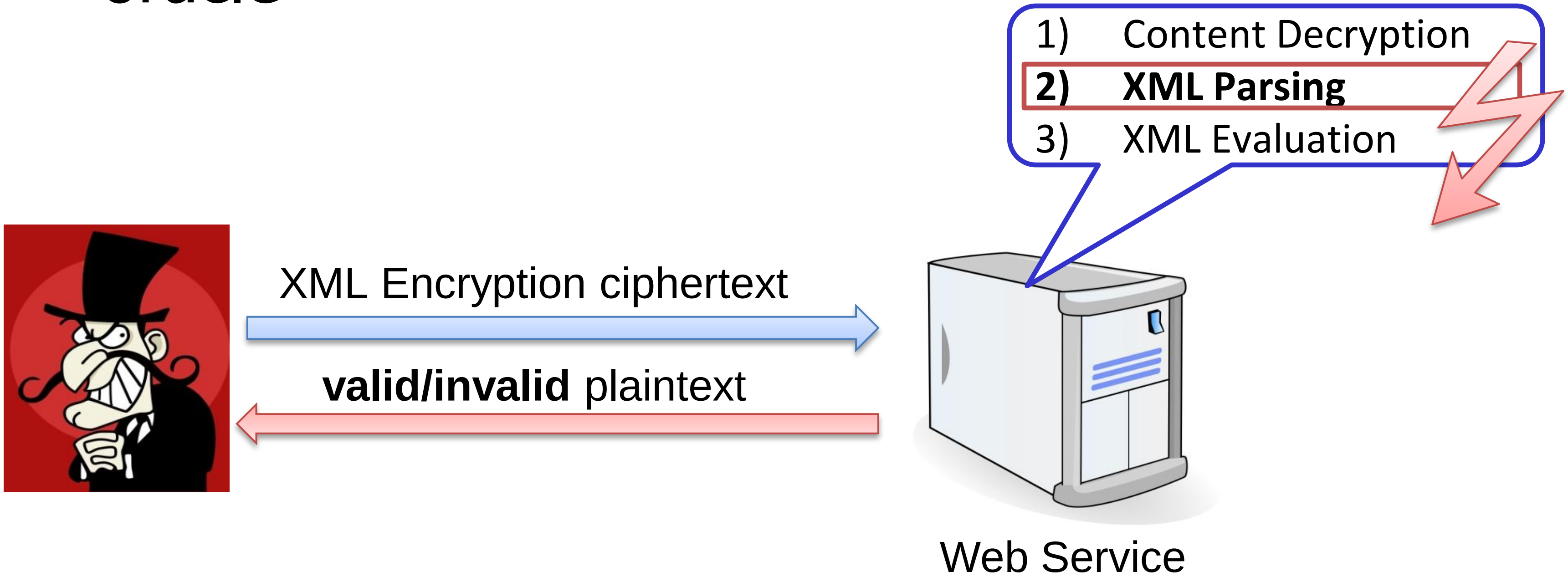
- XML is a text-based data format
- Characters (usually) encoded in ASCII
  - Type A: “special” characters

    EOF, BEL, ACK, …, <, &, …
  - Type B: other

    A,B,C, …, a,b,c, …, 1,2,3, …, !, %, …
- This talk:
  “Valid” plaintext contains **no Type-A** character

Decrypting by checking plaintext validity

- Using Web Services Server as plaintext validity oracle



1) Content Decryption

2) XML Parsing

3) XML Evaluation

XML Encryption ciphertext

valid/invalid plaintext

Web Service

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

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

0x00	(Type A)	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	(Type A)	0x27	'	0x47	G	0x67	g
0x08	(Type A)	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	(Type A)	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

Overview

1. **Breaking XML Signature**
   - **Cloud Computing Management Interfaces**
   - **Amazon EC2 SOAP Interface**
   - **XML Signature Wrapping on Eucalyptus and Amazon**
   - **Countermeasures and Conclusion**
2. **Breaking XML Encryption**
   - **Attack Scenario**
   - **Decrypting by checking plaintext validity**
   - **Application to CBC mode of operation in XML Encryption**
   - **Countermeasures and Conclusion**
3. **Conclusion**

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** - In Proceedings of the ACM Cloud Computing Security Workshop (CCSW), 2011.

Tibor Jager, Juraj Somorovsky: **How To Break XML Encryption** - In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), 2011.

Decrypting by checking plaintext validity

- ASCII exhibits nice pattern of Type A/B characters
- Suppose we can transform Enc(M) into Enc(M ⊕ msk) for any msk
  - **We can flip arbitrary plaintext bits, given only the ciphertext**
- Approach: Given C = Enc(M),
  1. Modify plaintext character-wise
  2. Query the “oracle” with each modified ciphertext
  3. Observe whether plaintext remains “valid” or not

Decrypting by checking plaintext validity

- **Example**
- We have eavesdropped a ciphertext
  C = Enc(“ACMCCS11”)
- We recover M = “ACMCCS11” character-wise
- How to determine (b_1,b_2) of M_1 = “A”?

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

0x00	(Type A)	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	(Type A)	0x27	'	0x47	G	0x67	g
0x08	(Type A)	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	(Type A)	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

OWASP

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

0x00	(Type A)	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	(Type A)	0x27	'	0x47	G	0x67	g
0x08	(Type A)	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	(Type A)	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

Overview

1. **Breaking XML Signature**
   - **Cloud Computing Management Interfaces**
   - **Amazon EC2 SOAP Interface**
   - **XML Signature Wrapping on Eucalyptus and Amazon**
   - **Countermeasures and Conclusion**
2. **Breaking XML Encryption**
   - **Attack Scenario**
   - **Decrypting by checking plaintext validity**
   - **Application to CBC mode of operation in XML Encryption**
   - **Countermeasures and Conclusion**
3. **Conclusion**

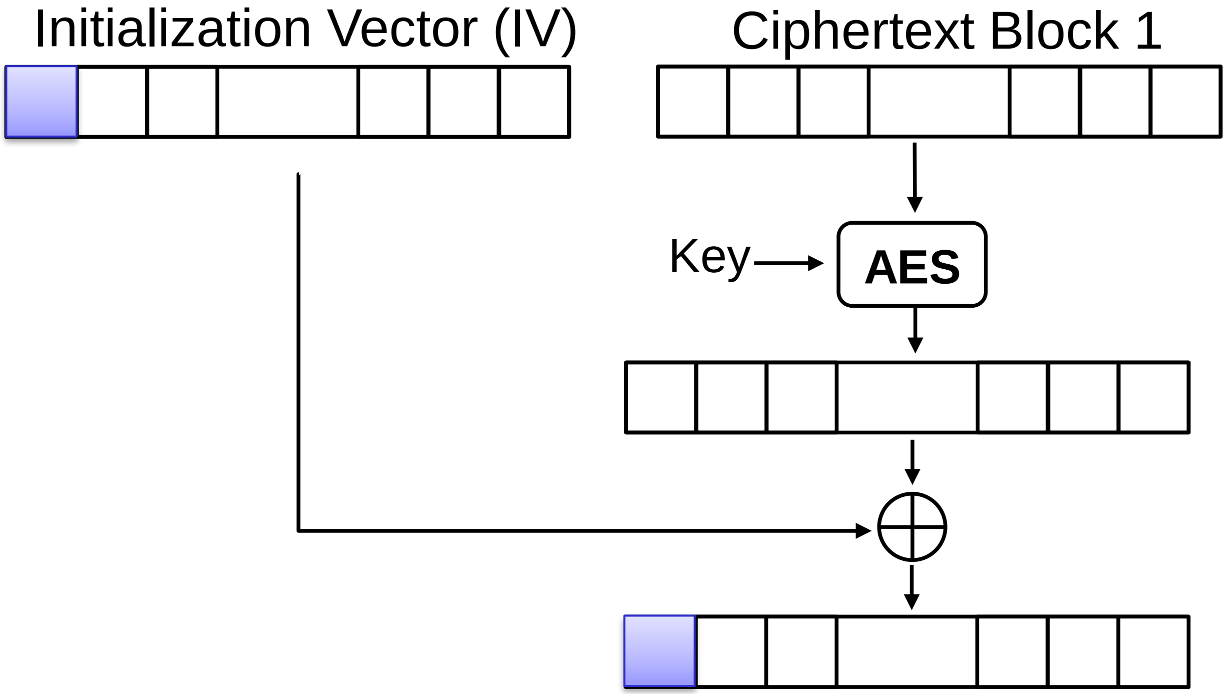
Juraj Somorovsky, Mario Heiderich, Meiko Jensen, Jörg Schwenk, Nils Gruschka, Luigi Lo Iacono: **All Your Clouds Are Belong to Us – Security Analysis of Cloud Management Interfaces** - In Proceedings of the ACM Cloud Computing Security Workshop (CCSW), 2011.

Tibor Jager, Juraj Somorovsky: **How To Break XML Encryption** - In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), 2011.

Computing Enc(M $\oplus$ msk) from Enc(M)

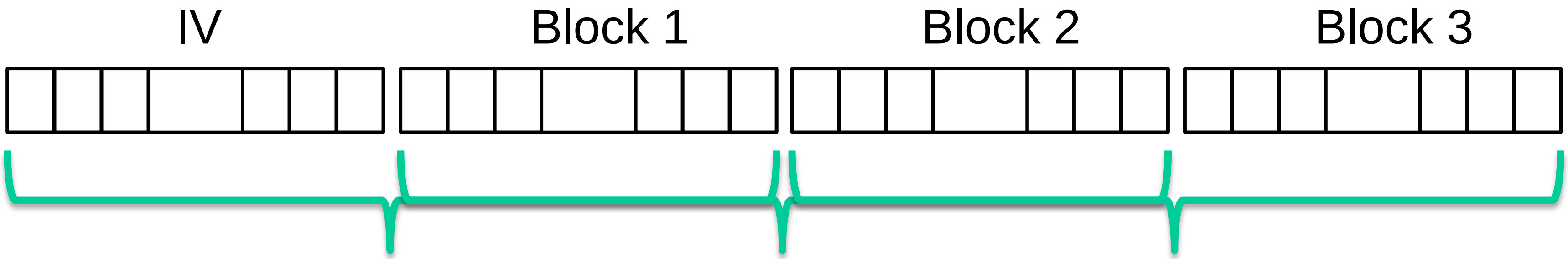
- XML Encryption uses block ciphers in *cipher-block chaining* (CBC) mode
- Known weakness of CBC
  - Padding oracle attacks
    (Vaudenay Eurocrypt 2002, and many more)
  - Error oracle attacks
    (Mitchell ISC 2005)
  - Chosen-plaintext attacks on SSL
    (Bard Cryptology ePrint 2004, Duong and Rizzo Ekoparty 2011)

Computing Enc(M ⊕ msk) from Enc(M)



- Transform encryption of M into encryption of M ⊕ msk for arbitrary msk!
- Applicable only to single-block ciphertexts

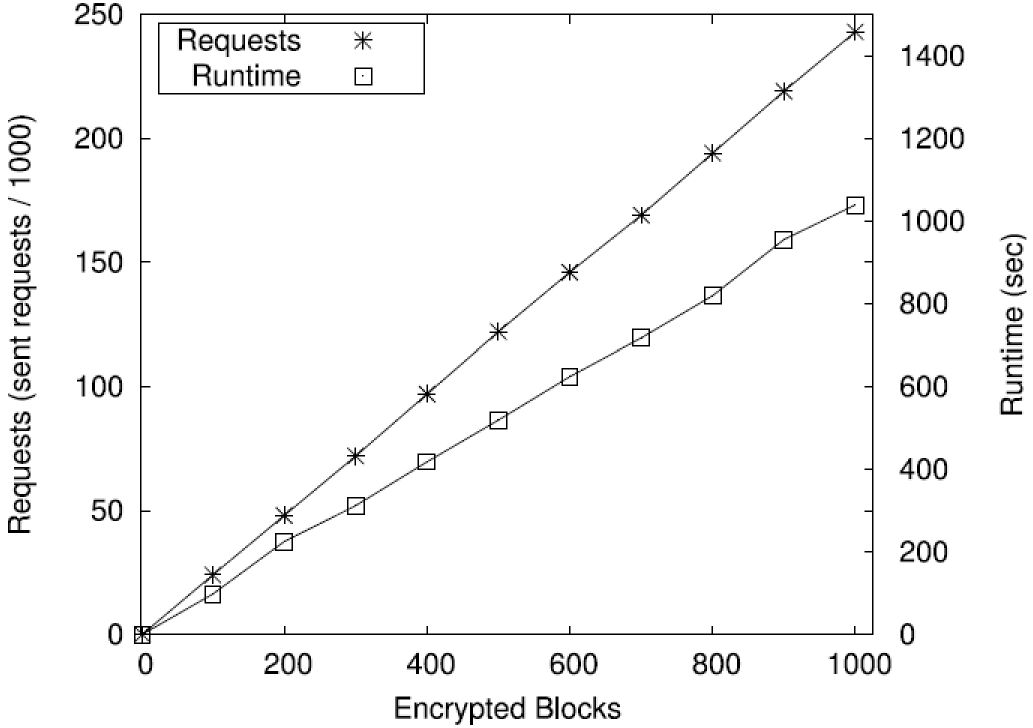
Multi-block ciphertexts



- CBC: Each block serves as IV for next block
  - Long ciphertexts “consist of many single-block ciphertexts”
- Apply single-block attack to decrypt longer ciphertexts block-wise
  - Decrypt Block 1
  - Decrypt Block 2 with Block 1 as IV
  - Decrypt Block *i+1* with Block *i* as IV

Experimental Results

Apache Axis 2, localhost, random plaintexts:



- Timing depends on system performance, network latency, ...
- Approx. 14 server requests/plaintext byte
  - Padding oracle attacks: ca. 128 requests/byte

Improvements and Variations

- XML schema is often public
  - Known structure of XML document
  - Skip blocks containing known plaintext
- Reduced plaintext set
  - Numbers, Base64, “Yes”/”No”, etc.
  - Less plaintext validity checks
- *Encryption* is possible, too
  (Following Rizzo and Duong, Usenix WOOT 2010)

Overview

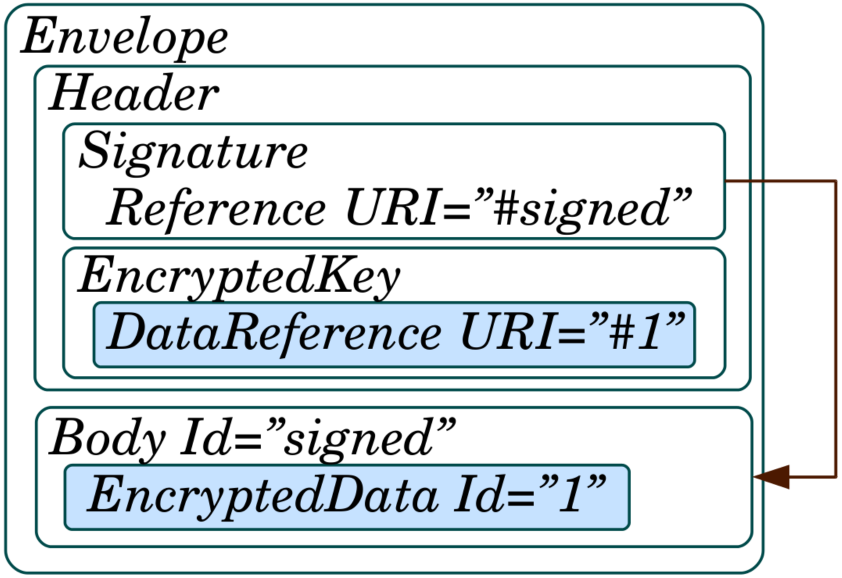
1. **Breaking XML Signature**
    - **Cloud Computing Management Interfaces**
    - **Amazon EC2 SOAP Interface**
    - **XML Signature Wrapping on Eucalyptus and Amazon**
    - **Countermeasures and Conclusion**
2. **Breaking XML Encryption**
    - **Attack Scenario**
    - **Decrypting by checking plaintext validity**
    - **Application to CBC mode of operation in XML Encryption**
    - **Countermeasures and Conclusion**
3. **Conclusion**

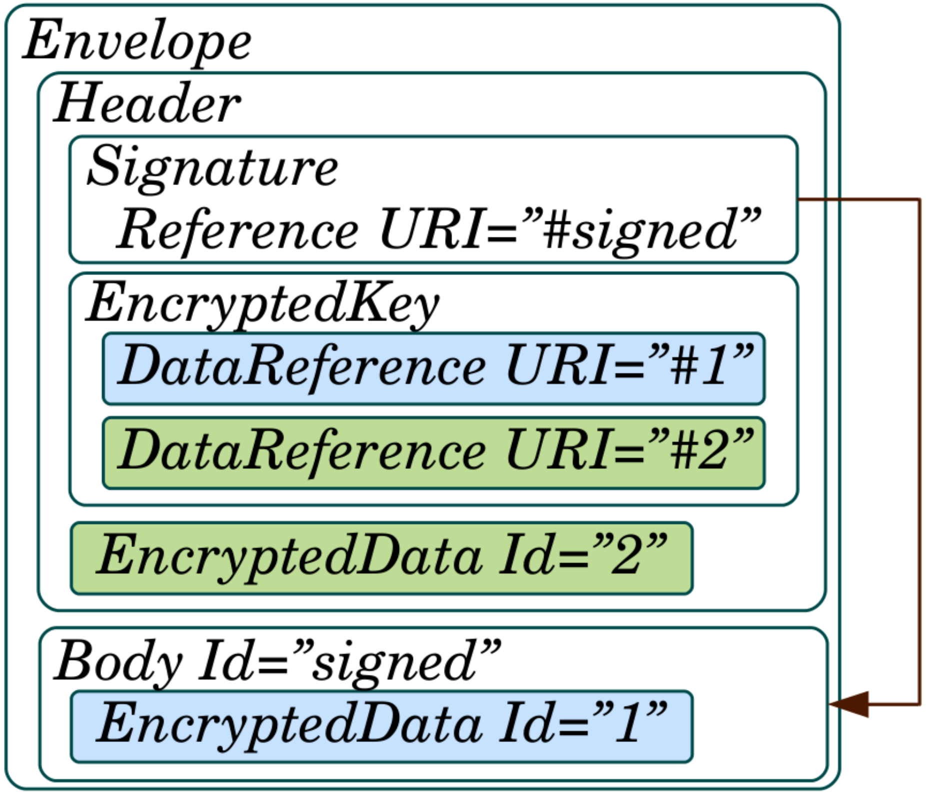
Juraj Somorovsky, Mario Heiderich, Meiko Jensen, Jörg Schwenk, Nils Gruschka, Luigi Lo Iacono: **All Your Clouds Are Belong to Us – Security Analysis of Cloud Management Interfaces** - In Proceedings of the ACM Cloud Computing Security Workshop (CCSW), 2011.

Tibor Jager, Juraj Somorovsky: **How To Break XML Encryption** - In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), 2011.

Countermeasures

- XML Signature
  - Signature Wrapping attacks
  - **Encryption Wrapping attack:** WS-Security Policy says, what must be encrypted…but it says not, what must not be encrypted





Countermeasures

- Authenticated encryption!
  - Not a standard-conformant option

Breaking XML Encryption – Conclusion

- Attack on XML Encryption
  - Applicable in particular to Web Services
  - All major WS frameworks are vulnerable
- No generic *ad-hoc* countermeasure
- W3C plans update of XML Encryption standard

Overview

1. **Breaking XML Signature**
   - **Cloud Computing Management Interfaces**
   - **Amazon EC2 SOAP Interface**
   - **XML Signature Wrapping on Eucalyptus and Amazon**
   - **Countermeasures and Conclusion**
2. **Breaking XML Encryption**
   - **Attack Scenario**
   - **Decrypting by checking plaintext validity**
   - **Application to CBC mode of operation in XML Encryption**
   - **Countermeasures and Conclusion**
3. **Conclusion**

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** - In Proceedings of the ACM Cloud Computing Security Workshop (CCSW), 2011.

Tibor Jager, Juraj Somorovsky: **How To Break XML Encryption** - In Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), 2011.

Conclusion

- XML Security applies cryptographic primitives on the message level
- It brings advantages in many applications: business process scenarios, Single Sign-On …
- However, the attacks exist…
  1. XML Signature Wrapping: pay attention when applying XML Signatures in your applications
  2. XML Encryption is broken:
     - You can use XML Signatures to ensure authenticity: XML Signature and XML Encryption Wrapping?
     - There are different countermeasures, but they are application and scenario specific
     - New standard with another chaining mode coming soon

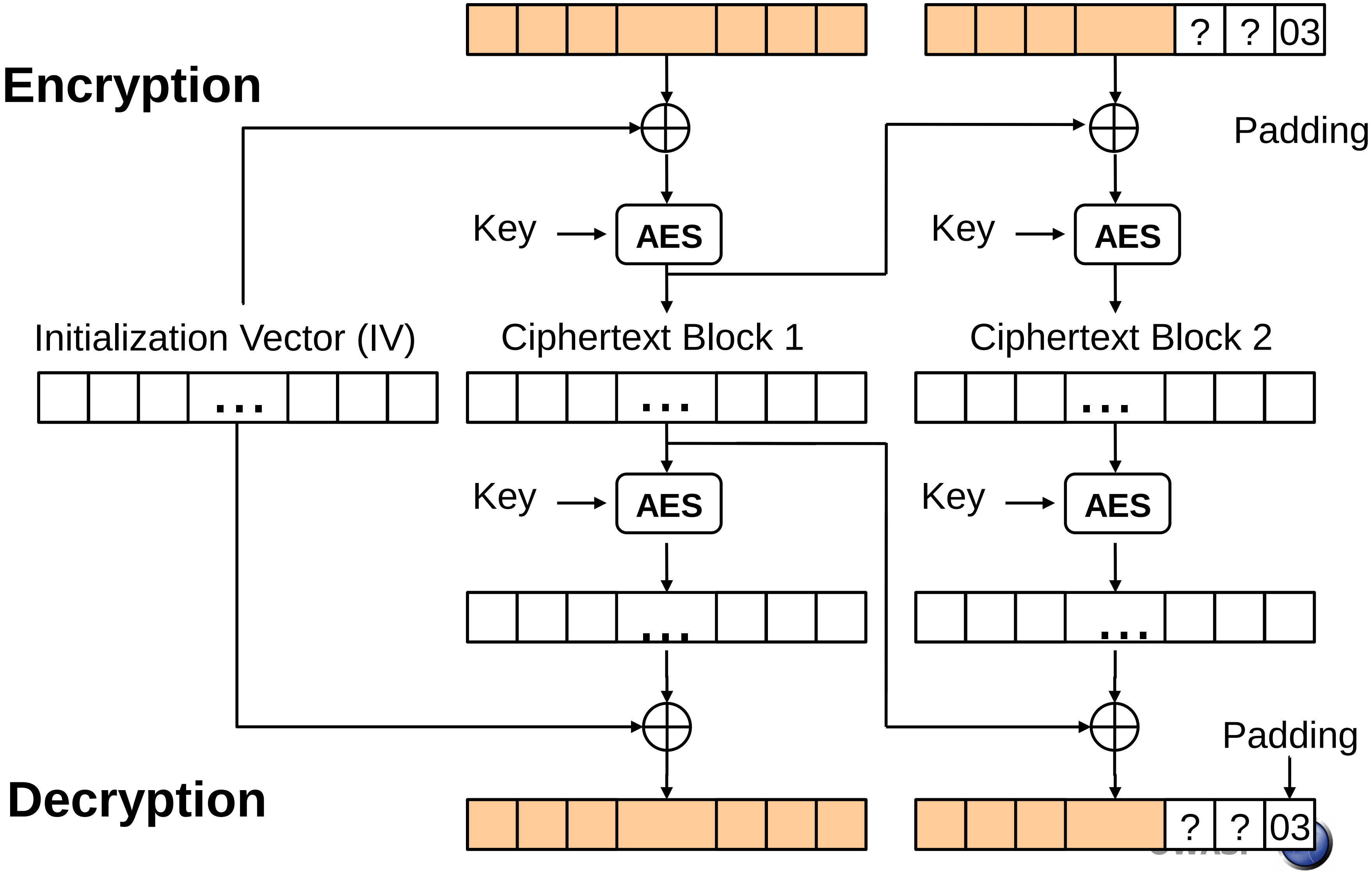
Responsible disclosure

- Attack disclosed in Feb 2011 to:
  - W3C, Apache, IBM, RedHat JBoss, Microsoft, governmental CERTs, vendor-sec mailing list, …
  - All have confirmed that attack is applicable
- Intensive cooperation with some developers
  - More than 100 e-mails since Feb 2011
- In contact with W3C working group
  - Authenticated encryption planned for v2.0

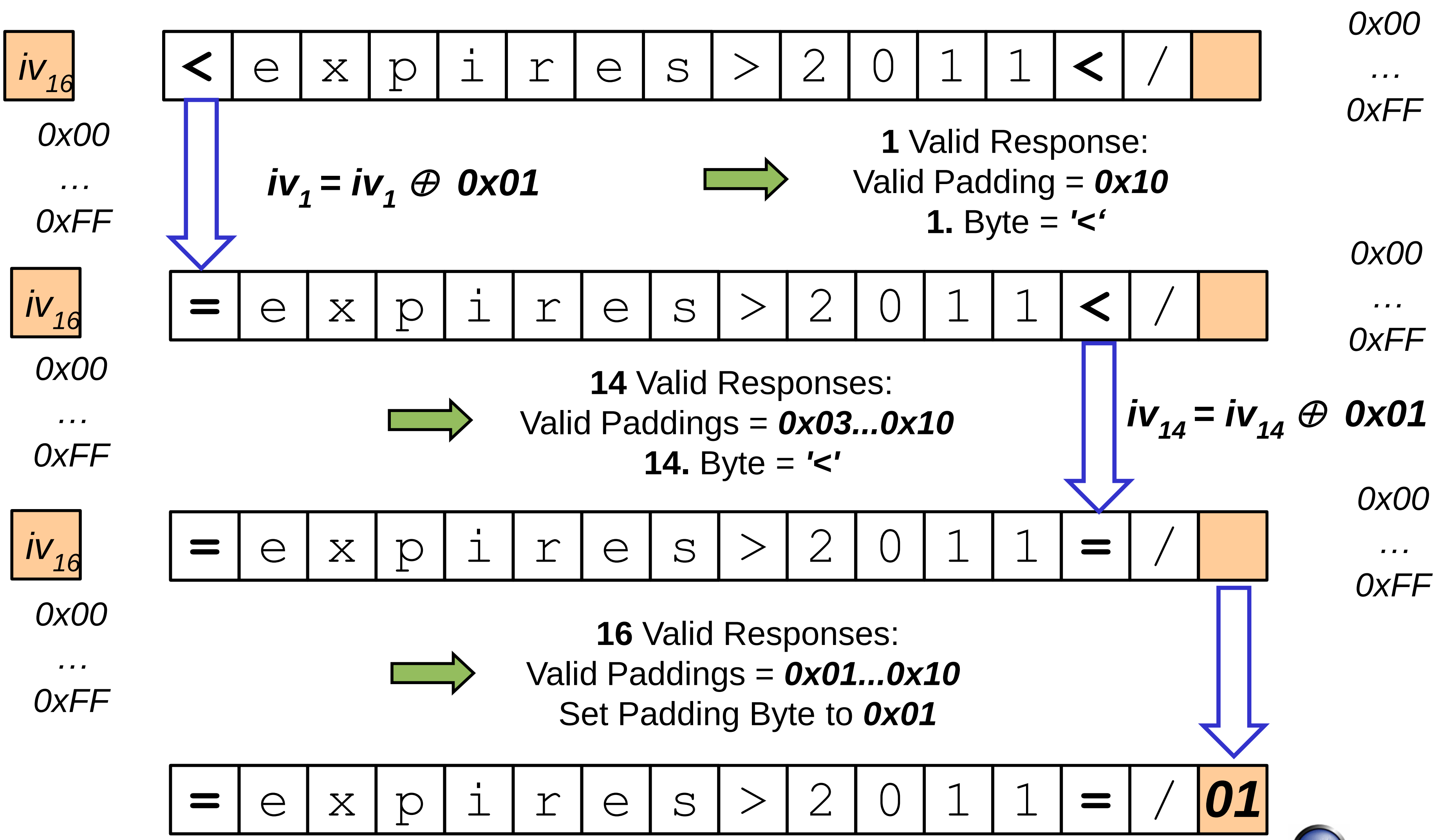
XML Signature Wrapping - Countermeasures

- Amazon Countermeasure + checking for duplicate Ids:
  - Not as easy as it seems to be
  - Id vs ID vs wsu:Id ….?
  - XML Entities?
- See what is signed:
  - Validate signature first
  - Forward only validated document parts
  - XML message becomes not well-formed, could lead to problems e.g. in XML Security Gateways
- Usage of XPath for position fixation
  - Another attacks [Jensen et al.: The curse of namespaces in the domain of xml signature]

CBC and padding in XML Encryption



Extracting ‘<‘



iv_{16}

0x00 … 0xFF

<	e	x	p	i	r	e	s	>	2	0	1	1	<	/	

0x00 … 0xFF

$iv_1 = iv_1 \oplus 0x01$

1 Valid Response:
Valid Padding = ***0x10***
1. Byte = ***'<‘***

iv_{16}

0x00 … 0xFF

=	e	x	p	i	r	e	s	>	2	0	1	1	<	/	

0x00 … 0xFF

14 Valid Responses:
Valid Paddings = ***0x03…0x10***
14. Byte = ***'<'***

$iv_{14} = iv_{14} \oplus 0x01$

iv_{16}

0x00 … 0xFF

=	e	x	p	i	r	e	s	>	2	0	1	1	=	/	

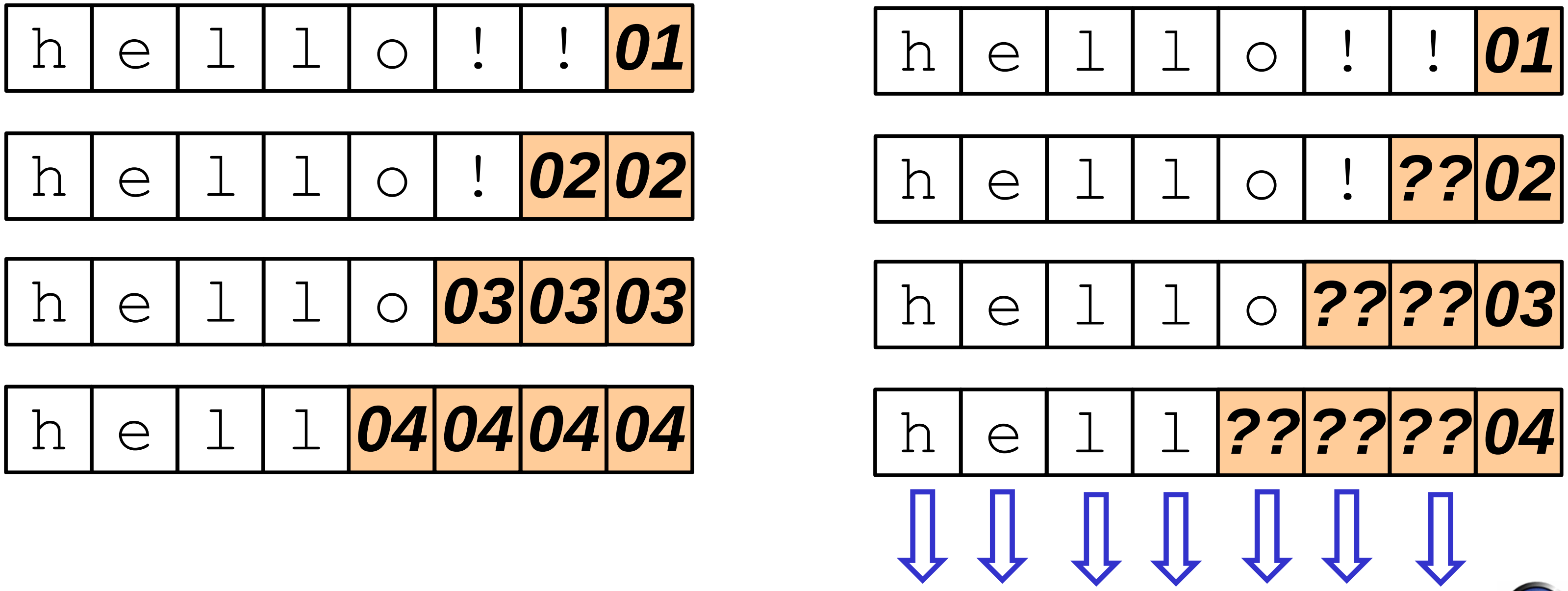
0x00 … 0xFF

16 Valid Responses:
Valid Paddings = ***0x01…0x10***
Set Padding Byte to ***0x01***

=	e	x	p	i	r	e	s	>	2	0	1	1	=	/	***01***

Difference to Vaudenay’s attack

- Misused security responses caused by incorrect **PKCS** padding
- Vaudenay: IPSEC, SSL, WTLS
- Rizzo and Duong: .NET Framework, JSF View States, Captchas



h	e	l	l	o	!	!	**01**
h	e	l	l	o	!	**02**	**02**
h	e	l	l	o	**03**	**03**	**03**
h	e	l	l	**04**	**04**	**04**	**04**

h	e	l	l	o	!	!	**01**
h	e	l	l	o	!	**??**	**02**
h	e	l	l	o	**??**	**??**	**03**
h	e	l	l	**??**	**??**	**??**	**04**