One Bad Apple: Backwards Compatibility Attacks on State-of-the-Art Cryptography

Tibor Jager
Horst-Görtz Institute for IT-Security, Ruhr-University Bochum, Germany

Kenneth G. Paterson
Royal Holloway, University of London, United Kingdom

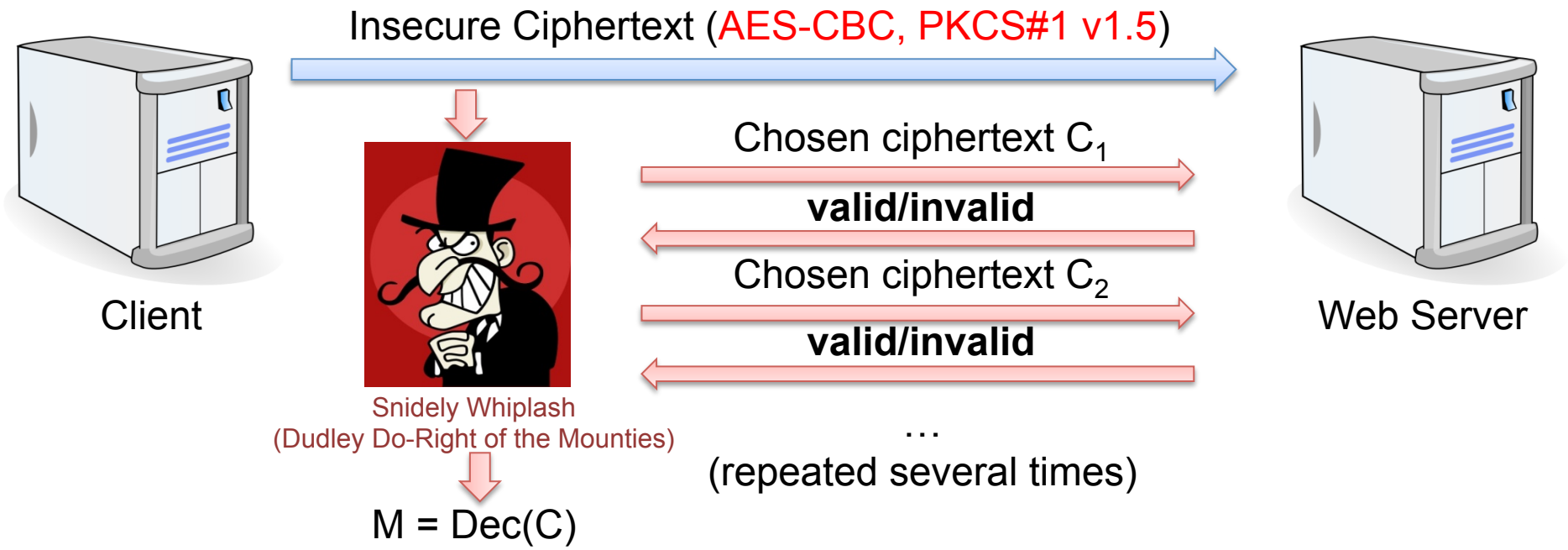
Juraj Somorovsky
Horst-Görtz Institute for IT-Security, Ruhr-University Bochum, Germany

Introduction

- Typical evolution: Crypto algorithms *designed*...crypto algorithms *broken*
- Many examples, e.g.:
  - Crypto 1998, Bleichenbacher: Chosen-Ciphertext Attack on RSA PKCS#1 v1.5
    - Applicable to SSL and other protocols
    - PKCS#1 was updated to v2.0 (RSA-OAEP)
  - Eurocrypt 2002, Vaudenay: Chosen-Ciphertext Attack on Cipher Block Chaining mode of operation
    - Applied e.g. on SSL/TLS, IPSEC ...
    - Integrity check has to be applied (HMAC, AES-GCM,...)

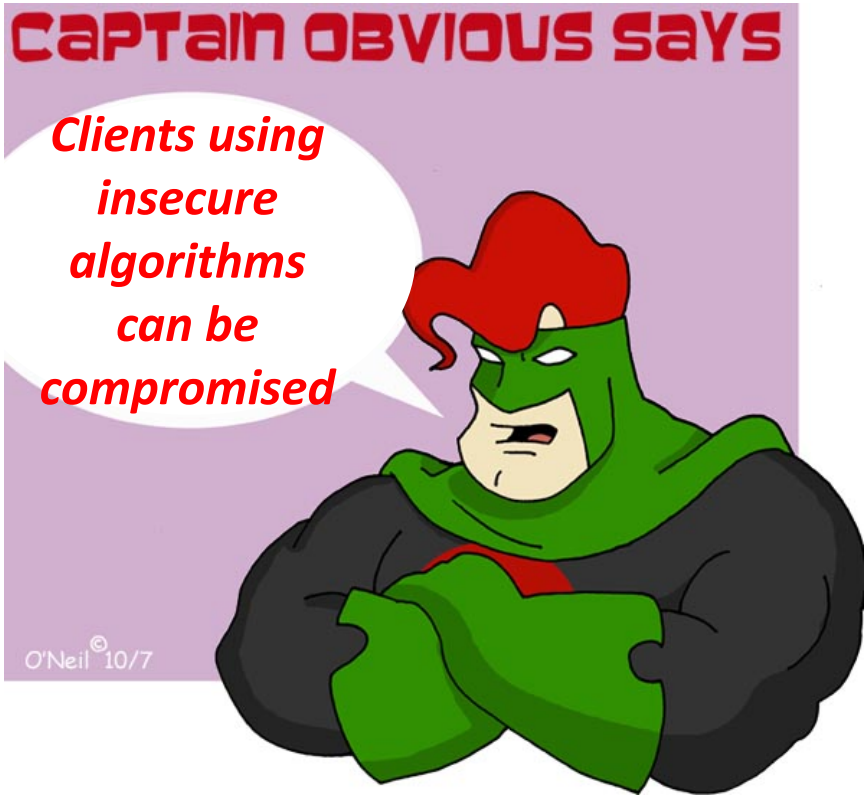
Introduction

- Adaptive chosen-ciphertext attacks:



Introduction

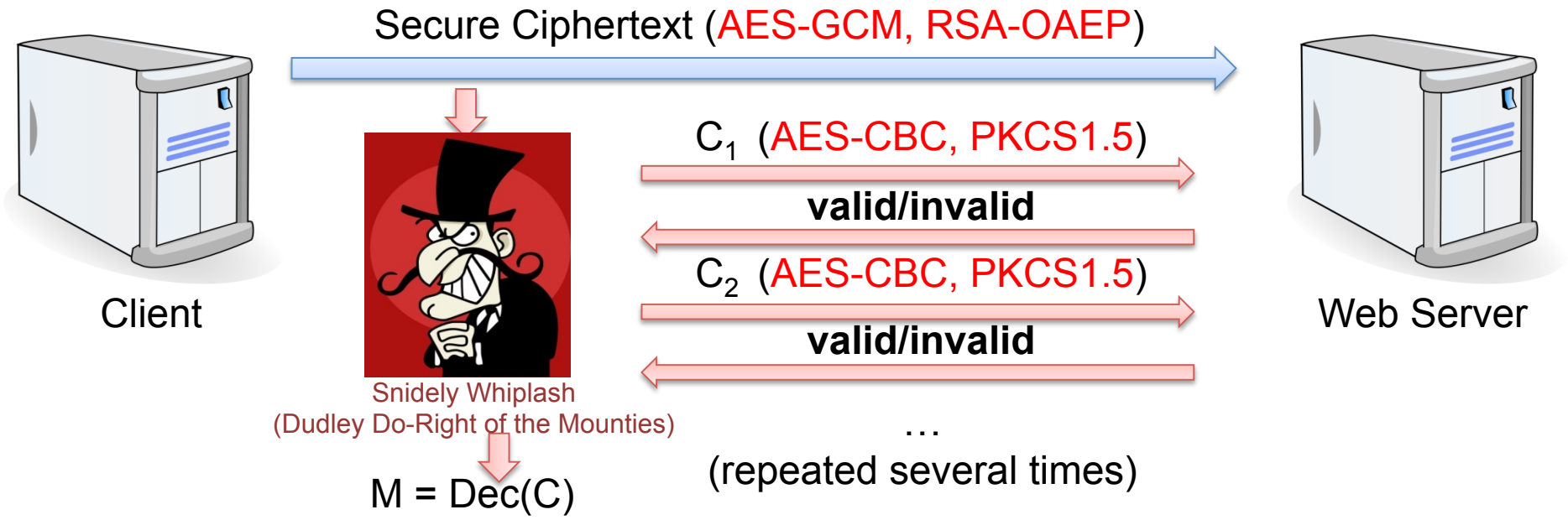
- Standards updated, but “insecure algorithms” still included for *backwards compatibility* reasons:
  - SSL / TLS servers support PKCS#1 v1.5
  - XML Encryption servers support AES-CBC and PKCS#1 v1.5
  - JSON Web Encryption servers support PKCS#1 v1.5
  - **What about using secure algorithms?**
  - **What could go wrong?**



Backwards Compatibility Attacks – Scenario

- Server supports 2 types of algorithms:
  - Secure: RSA-OAEP, AES-GCM
  - Insecure (legacy): PKCS#1 v1.5, AES-CBC
- Key reuse between secure and insecure algorithms
- Client has the best intention to use secure algorithms
- Attacker can force the server to use insecure algorithms

Backwards Compatibility Attacks



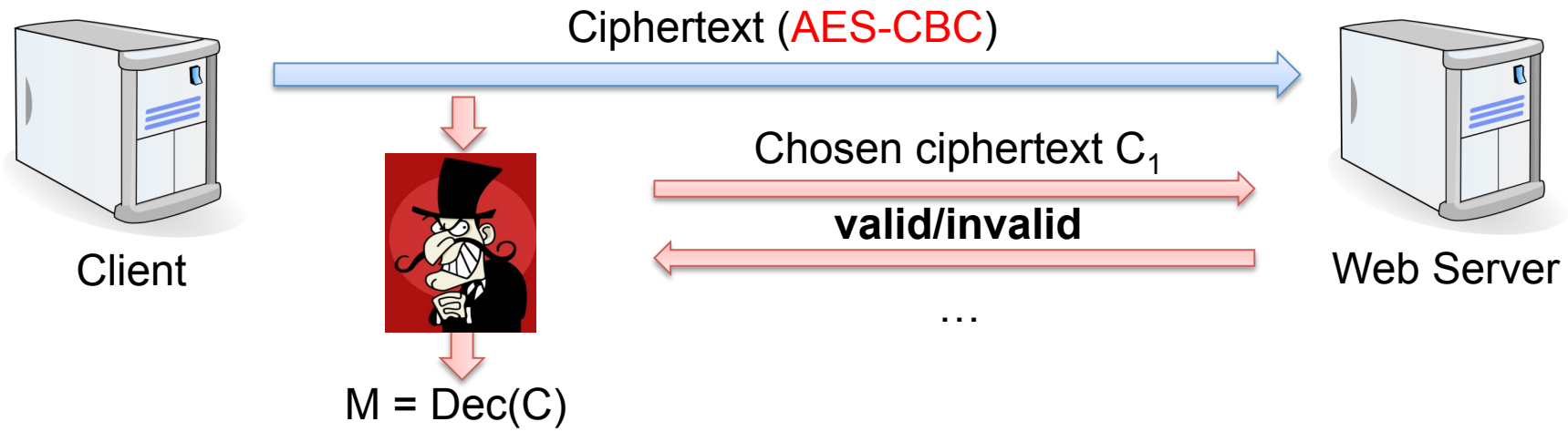
Overview of our Results

- Symmetric key crypto:
  - Break indistinguishability of AES-GCM
  - Decrypt AES-KW
- Public key crypto:
  - Decrypt RSA-OAEP ciphertexts
  - Forge server signatures
- Attacks applied to:
  - XML Encryption
  - JSON Web Encryption

Breaking Indistinguishability of AES-GCM

Attacks on CBC mode of operation

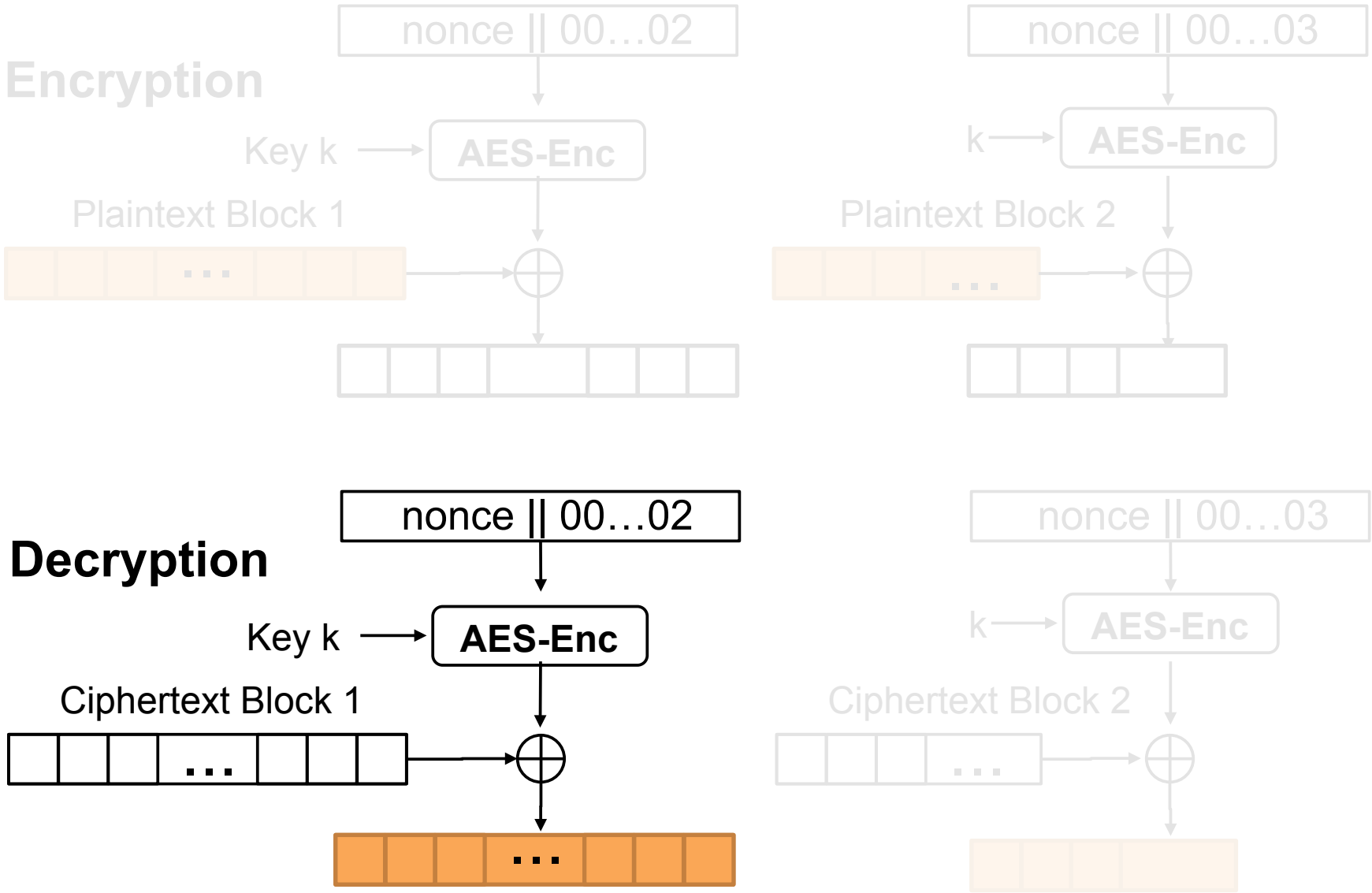
- Introduced by Vaudenay [Eurocrypt02]
- 128 queries to decrypt one byte
- Several prominent targets include ASP.Net [S&P'11], XML Encryption[CCS'11], DTLS[NDSS'12]
- The attacks provide us a decryption function: $m = Dec_{AES}(C1)$



AES-Galois Counter Mode (AES-GCM)

- Authenticated encryption scheme
- No chosen-ciphertext attacks possible
- Counter mode
- Authentication tag computed over Galois field (not relevant for us)

Counter Mode

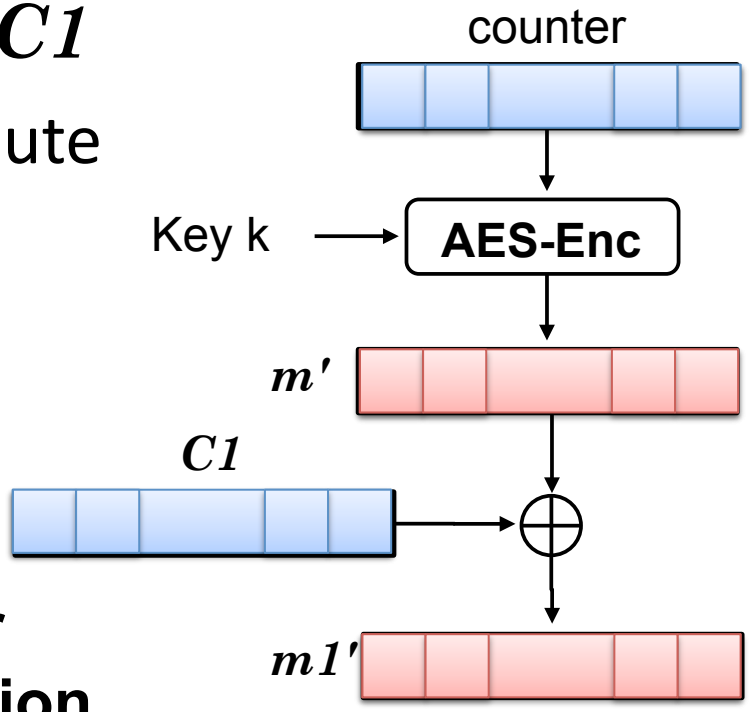


Breaking indistinguishability of AES-GCM

- We have $m = Dec_{AES}(C)$
  - 2000 queries for one AES block

1. Guess $m1'$
2. Compute $m' = m1' \; XOR \; C1$
3. Use the CBC attack to compute
   $x = Dec_{AES}(m')$
4. If $x == counter$
   Correct guess

known



Counter Decryption

Breaking indistinguishability of AES-GCM

- What does this mean?
- We can decrypt ciphertexts with a low entropy (credit card numbers, blocks with known plaintexts)

- Extended attack version:
  - One AES-GCM plaintext guess == one CBC query

Details boring,
See our paper or:

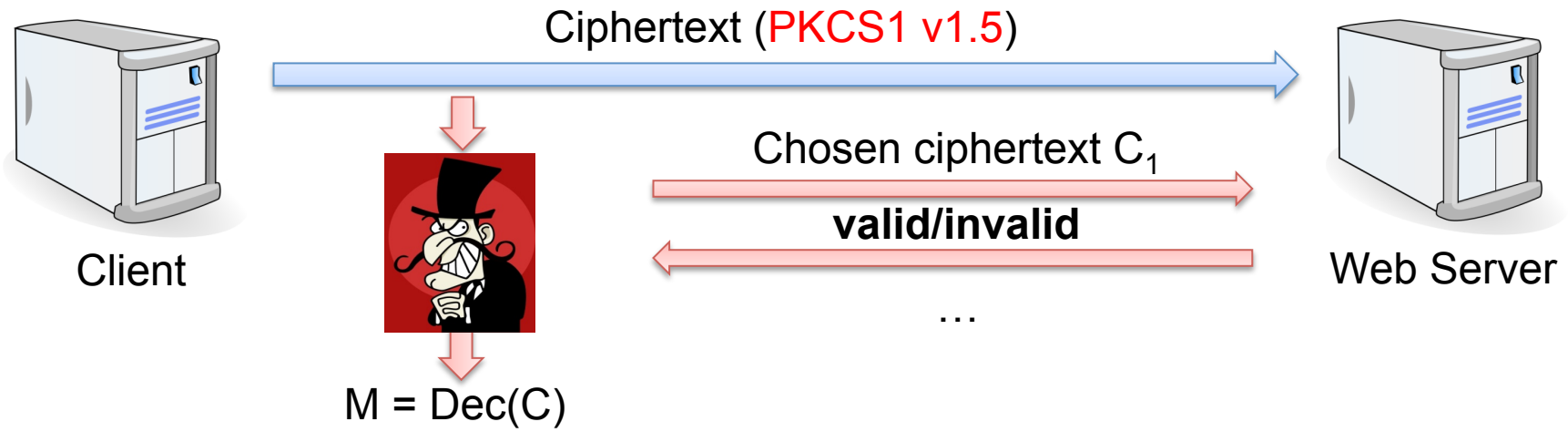
or

- Similar attacks applicable to different combinations of modes of operation…and to the asymmetric crypto

Asymmetric Crypto Results

Bleichenbacher's Attack on PKCS1 v1.5

- Introduced by Bleichenbacher [Crypto 1998]
- Targets include Hardware Security Modules [Crypto'12], XML Encryption[ESORICS'12]



The Power of Bleichenbacher's Attack

- We get an RSA decryption function:

$$x = Dec_{RSA}(c)$$

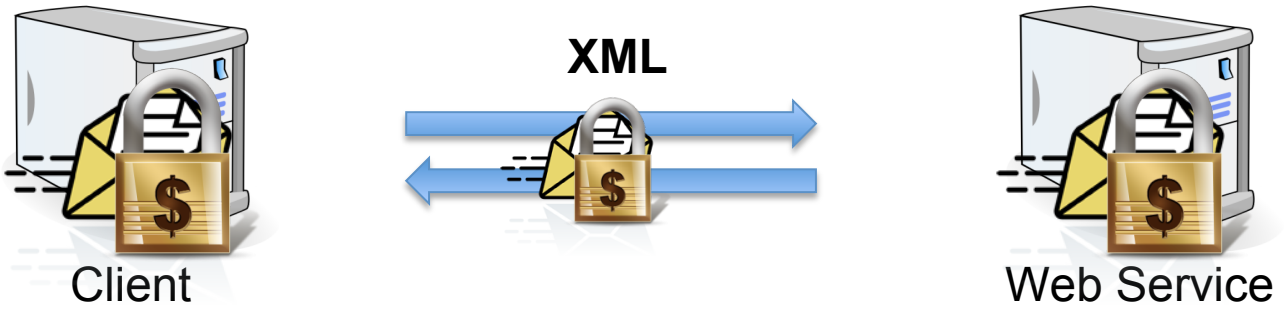
- It is thus possible to:
  - Decrypt RSA-OAEP ciphertexts
  - Forge server signatures
- Prerequisite:
  - Server uses the same key pair for more algorithms

Application to XML Encryption and JSON Web Encryption

XML Encryption

- W3C standard for encrypting XML data
- Published in 2002
- Describes various methods for applying
  - Symmetric ciphers (AES-CBC, AES-GCM, 3DES-CBC)
  - Public-key encryption (RSA-PKCS#1 v1.5, v2.0)

  to XML documents
- Applied e.g. in Web Services



JSON Web Encryption

- IETF Standard
- Still a draft
- Describes methods for applying crypto to JSON documents
- Motivation: JSON Web Services, browser applications

Why is it possible to force the server to use a different algorithm?

- XML Encryption and JSON Web Encryption provide the algorithm data directly in the message, e.g.:

```
{ "alg":" RSA1_5 ",
  "enc":"A256GCM",
  "iv":"__79_Pv6-fg",
  "jku":"example.com/pk.jwk" }
```

Countermeasures

Countermeasures

- Disable Legacy Cryptosystems
  - Shibboleth (Single Sign-On solution) blacklisted PKCS1 v1.5
  - Not possible in many scenarios
  - Platforms often do not support newer cryptographic standards
  - What if a secure algorithm gets broken???

Countermeasures

- Apply key separation principle
  - Different keys for different algorithms
  - Symmetric:
    - k' = PRF(k, ”Algorithm Identifier”)
  - Asymmetric:
    - Use different key pairs
    - X.509 certificates do not enforce by default
    - WS-Security Policy (Section 7.5): “Commonly used asymmetric algorithms, such as RSA, allow the same key pair to be used for both encryption and signature”
- Security Considerations added to XML Encryption 1.1

Conclusions

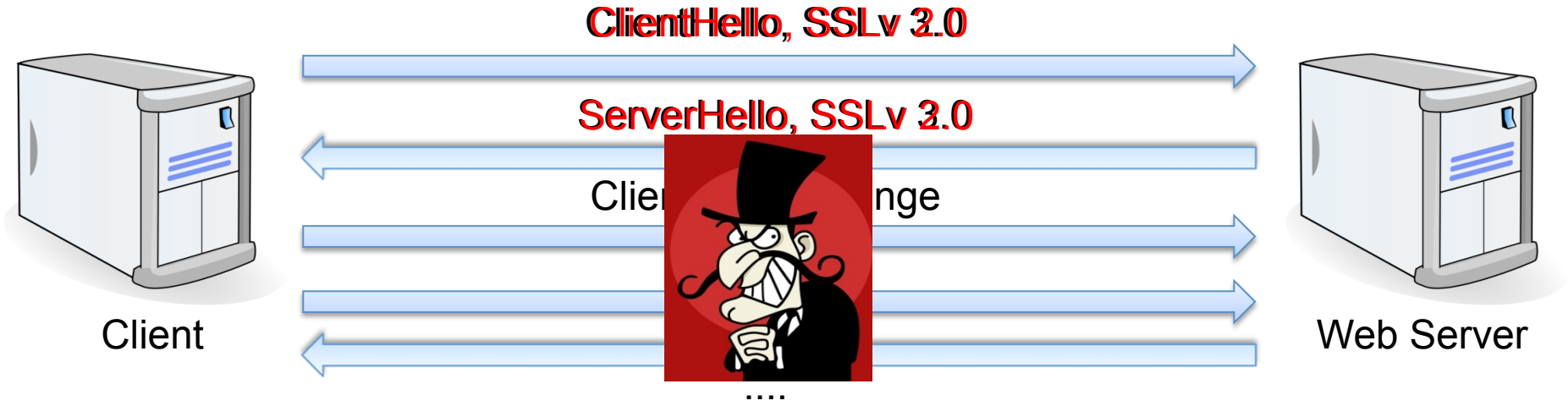
- Introduced Backwards Compatibility attacks
- Presence of an insecure algorithm can spoil the security of state-of-the-art algorithms:
  - Break indistinguishability of AES-GCM
  - Break RSA-OAEP
  - Forge server signatures
- Insecure algorithms should be omitted
- Key separation important

BACKUP Slides

Related Work

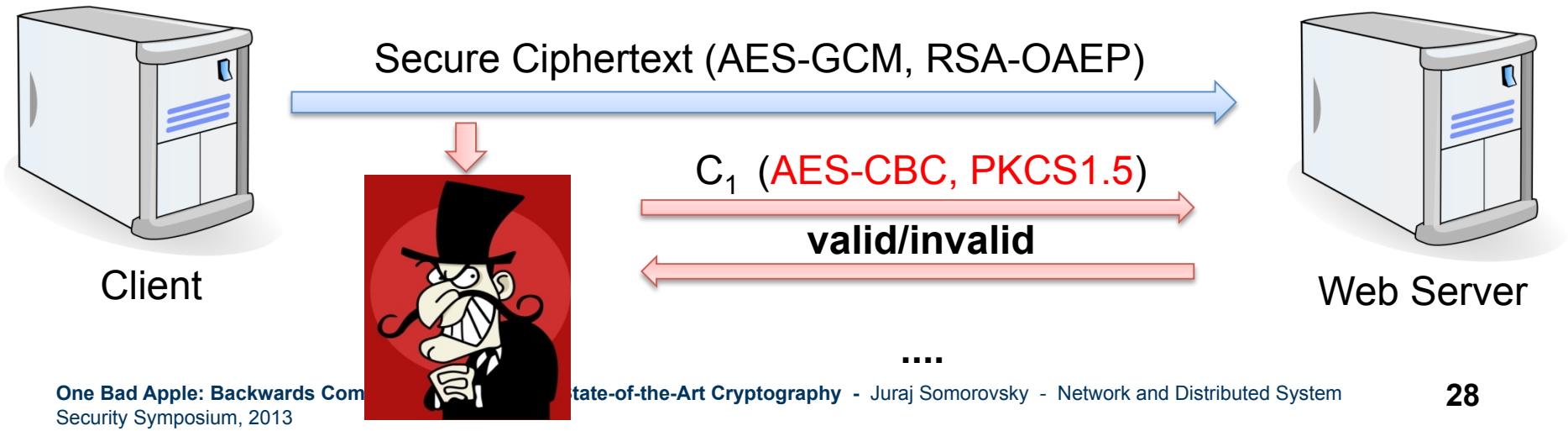
Differences to Version Rollback Attacks

- Version Rollback Attacks:
  - Wagner and Schneier
  - Interactive protocols (e.g. SSL/TLS)
  - Both partners are lured into using weak cryptography



Differences to Version Rollback Attacks

- Backwards Compatibility Attacks:
  - Non-Interactive protocols (e.g. in Web Services)
  - Only one communication partner (server) is lured into using weak cryptography
  - Weaker attacker



Communication with Developers

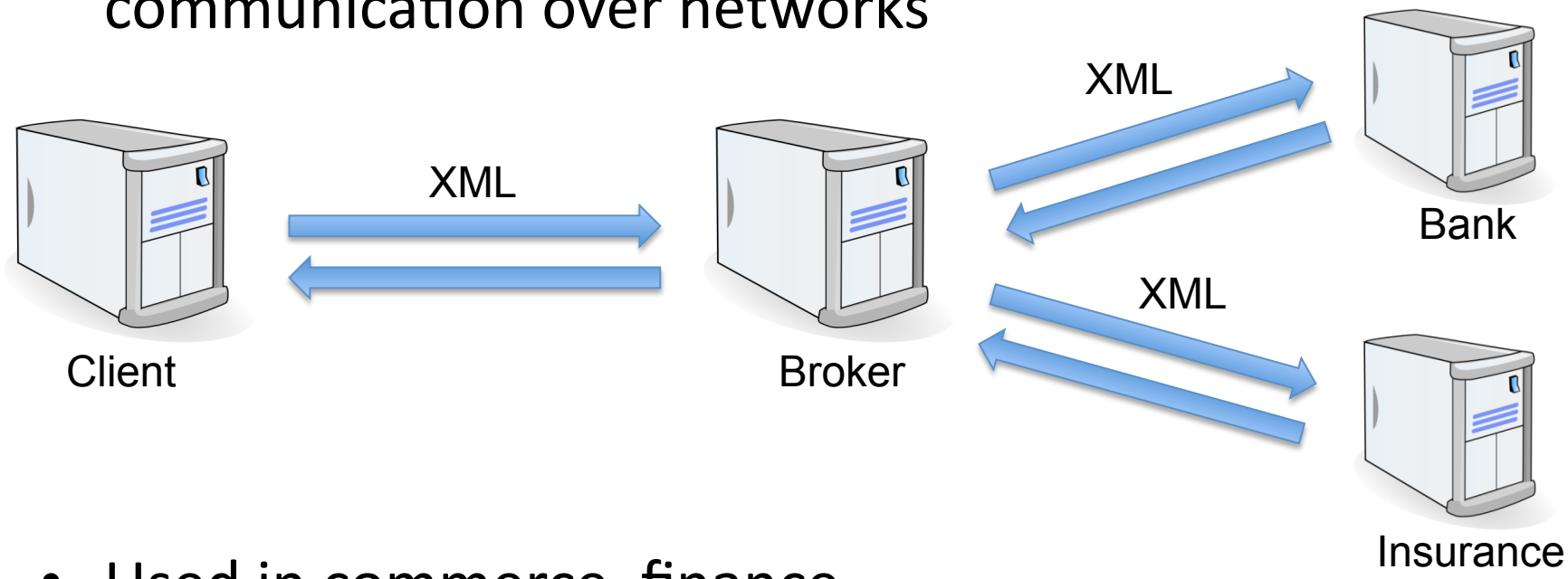
- Web Services Policy allows you to restrict algorithms (e.g. RSA-OAEP) used on the server
- Apache CXF processing:
  1. Decrypt everything in the message
  2. Check if the encryption algorithm could be used
- Attacker can force Apache CXF to process any algorithm (also PKCS 1 v1.5)
- Fix: Algorithm check is now performed before decryption

Communication with Developers

- PingIdentity
  - Single Sign-On solution
  - Suggested their users to use the same RSA key pair for encryption and signature
  - Appropriate key separation now
- Shibboleth
  - Single Sign-On software
  - *Blacklisted* RSA PKCS#1 v1.5 by default in the Service Provider v2.5

Motivation – XML Security

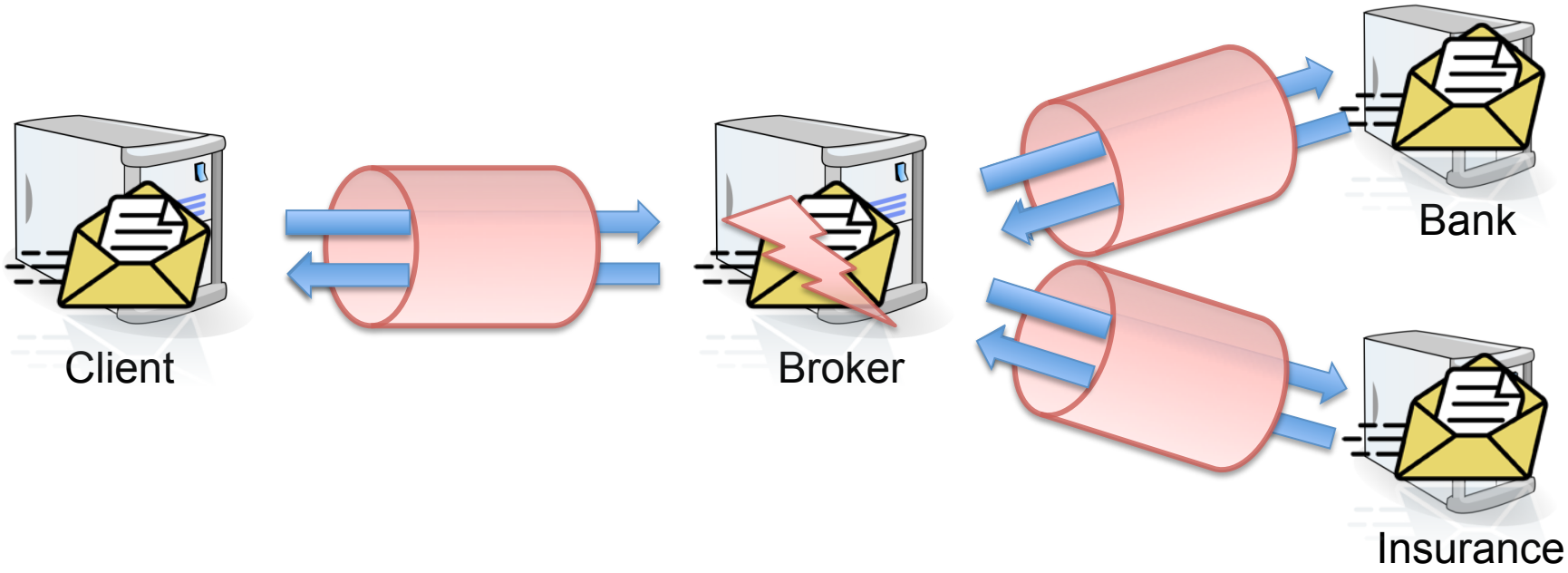
- Web Services: Method for machine-to-machine communication over networks



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

Motivation – XML Security

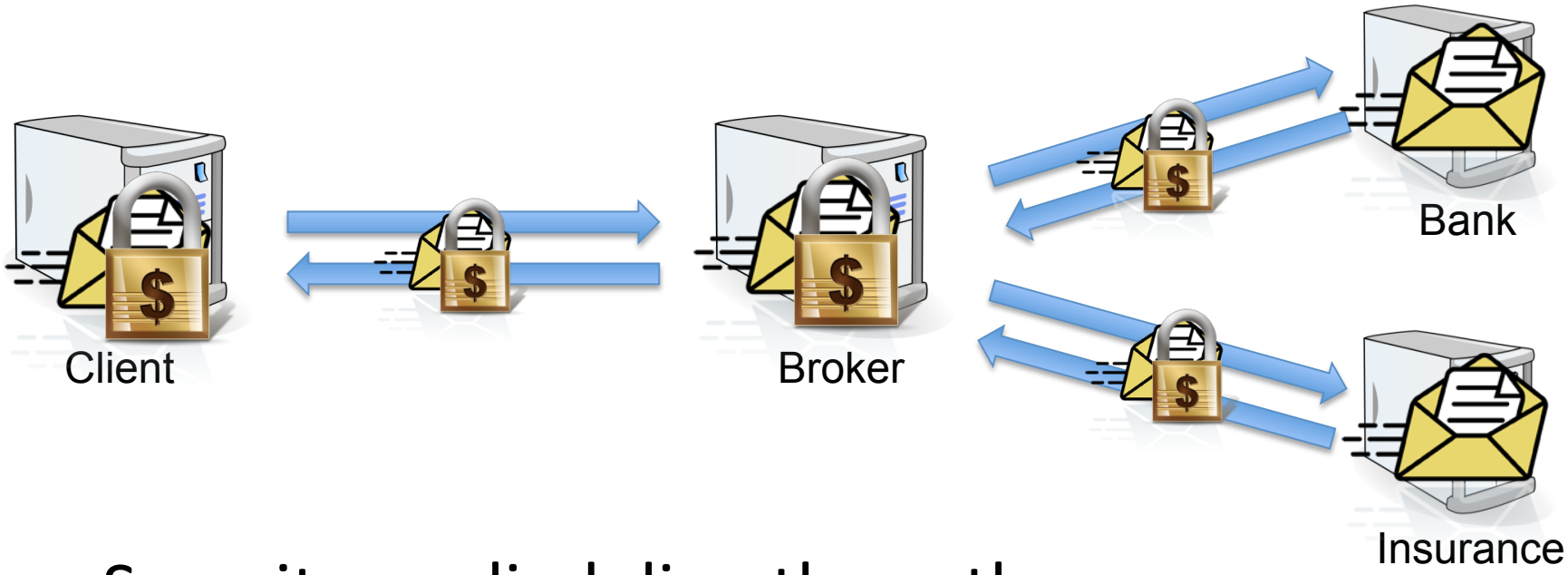
- SSL / TLS: transport-level security



- Messages secured only during transport!

Motivation – XML Security

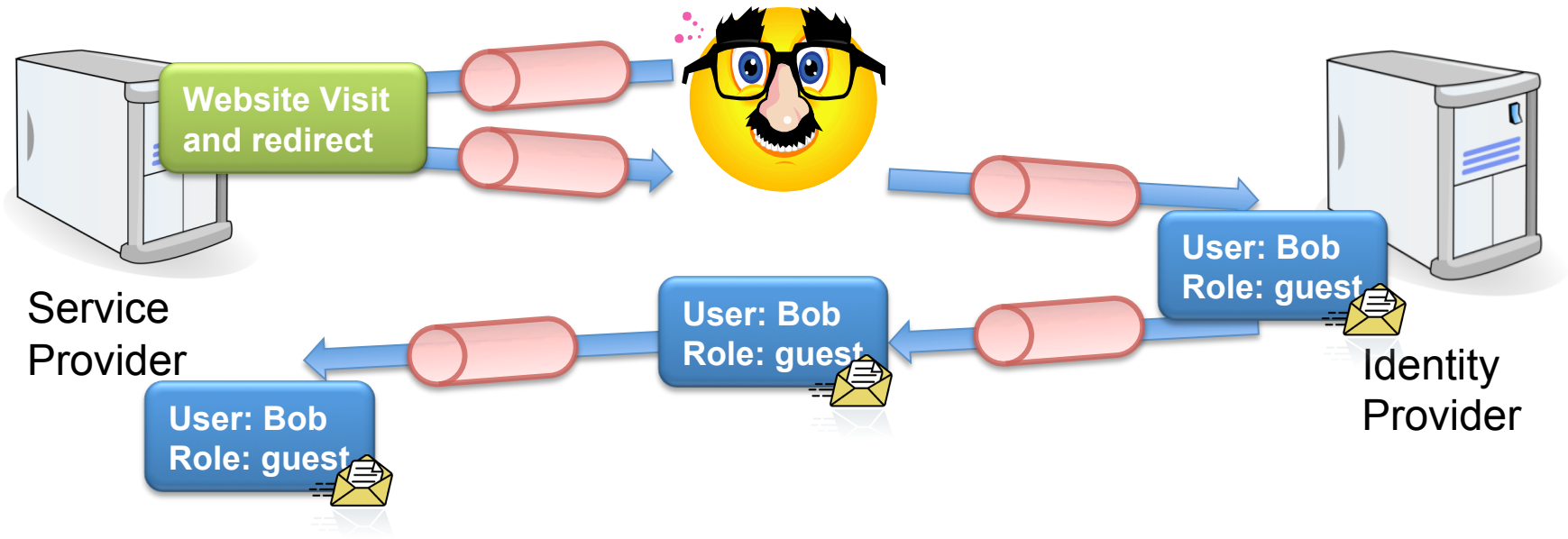
- Message level security



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

Motivation – XML Security

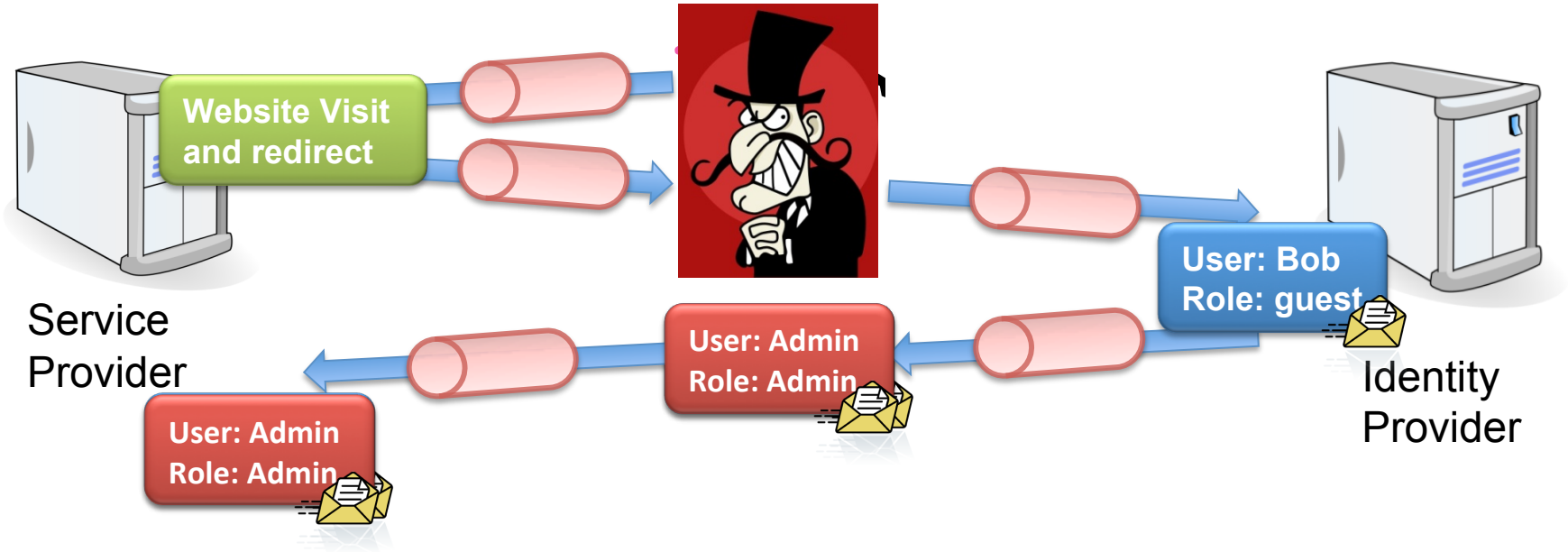
- Another example: Browser-based Single Sign-On



- Messages secured only during transport!

Motivation – XML Security

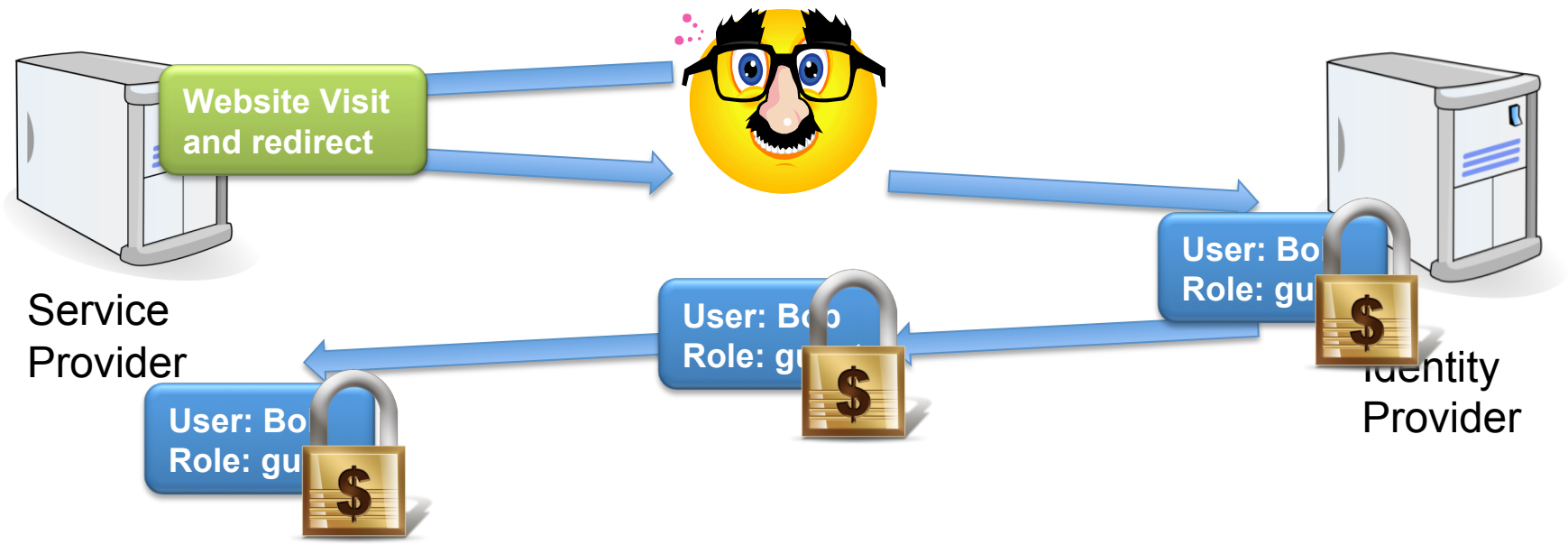
- Does SSL / TLS help?



- Need for message level security!

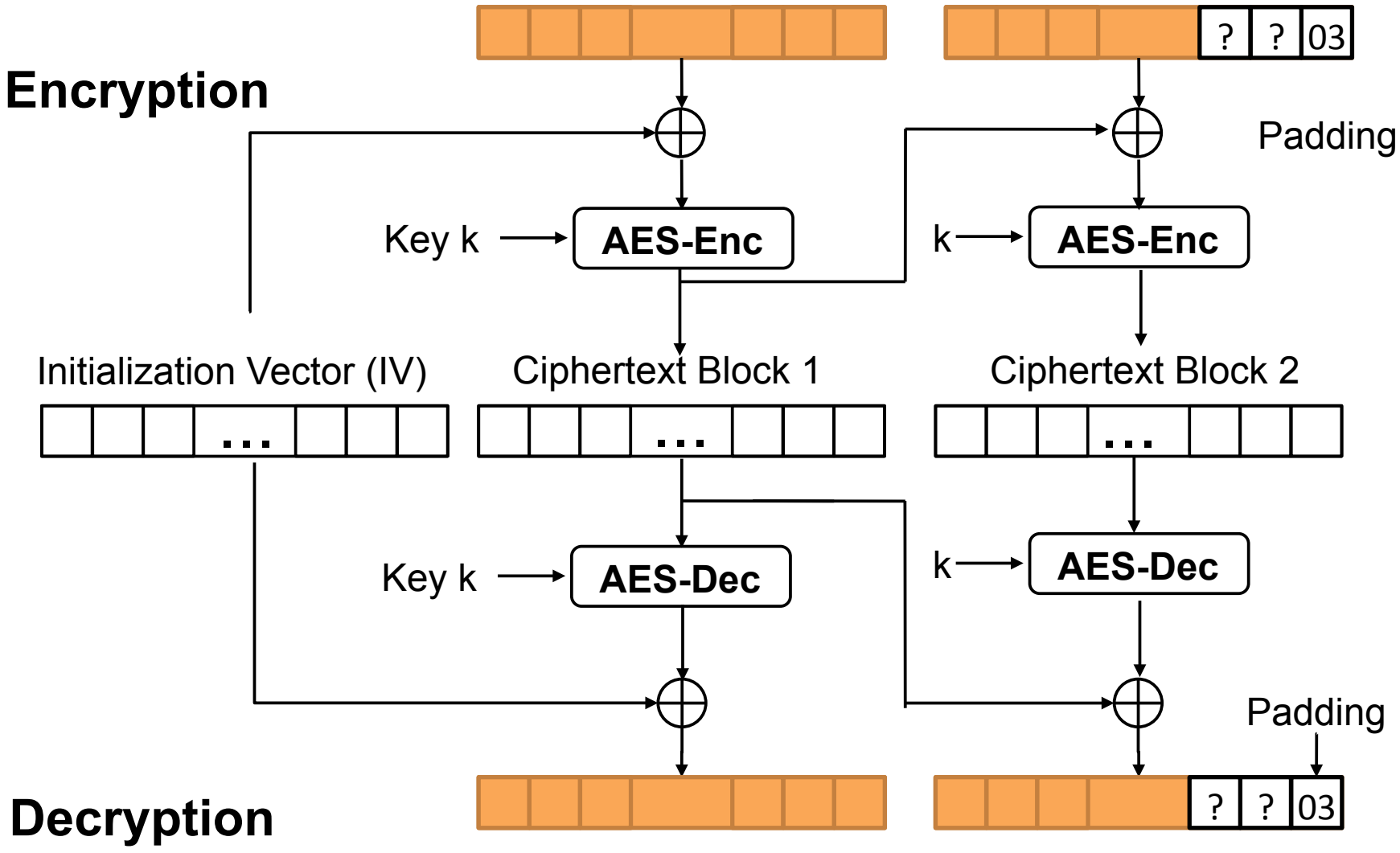
Motivation – XML Security

- Another example: Browser-based Single Sign-On

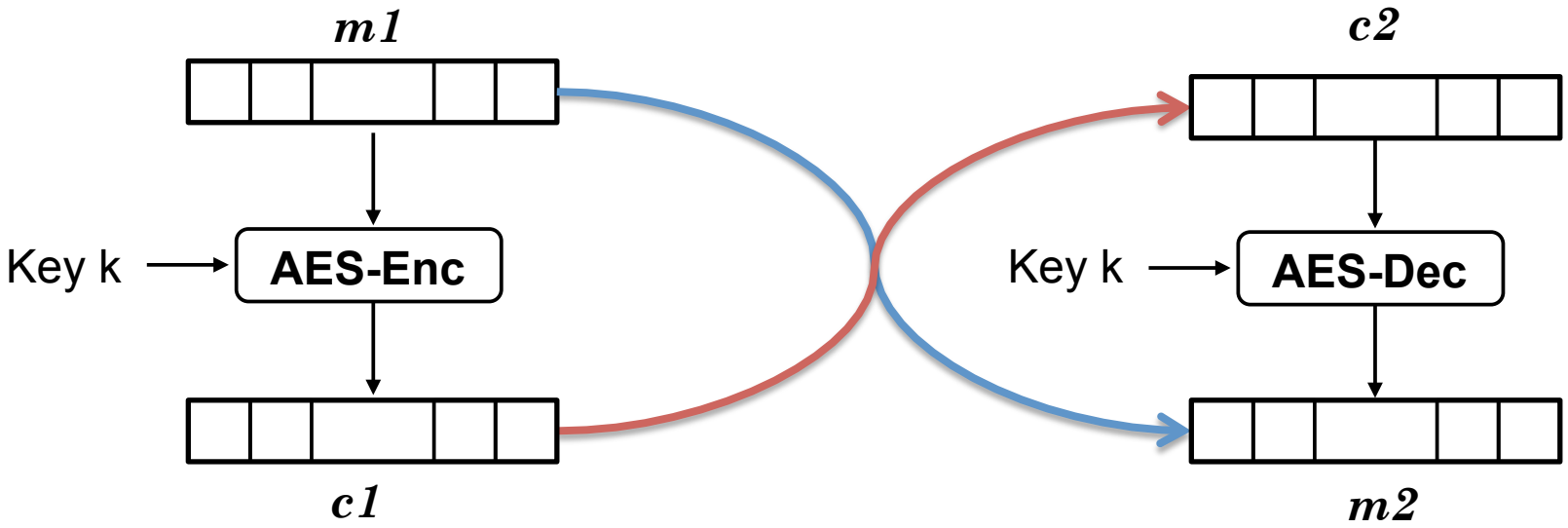


- Could be realized using XML Signature and XML Encryption

Cipher Block Chaining (CBC)



Ingredient 1: Symmetric Enc- / Decryption



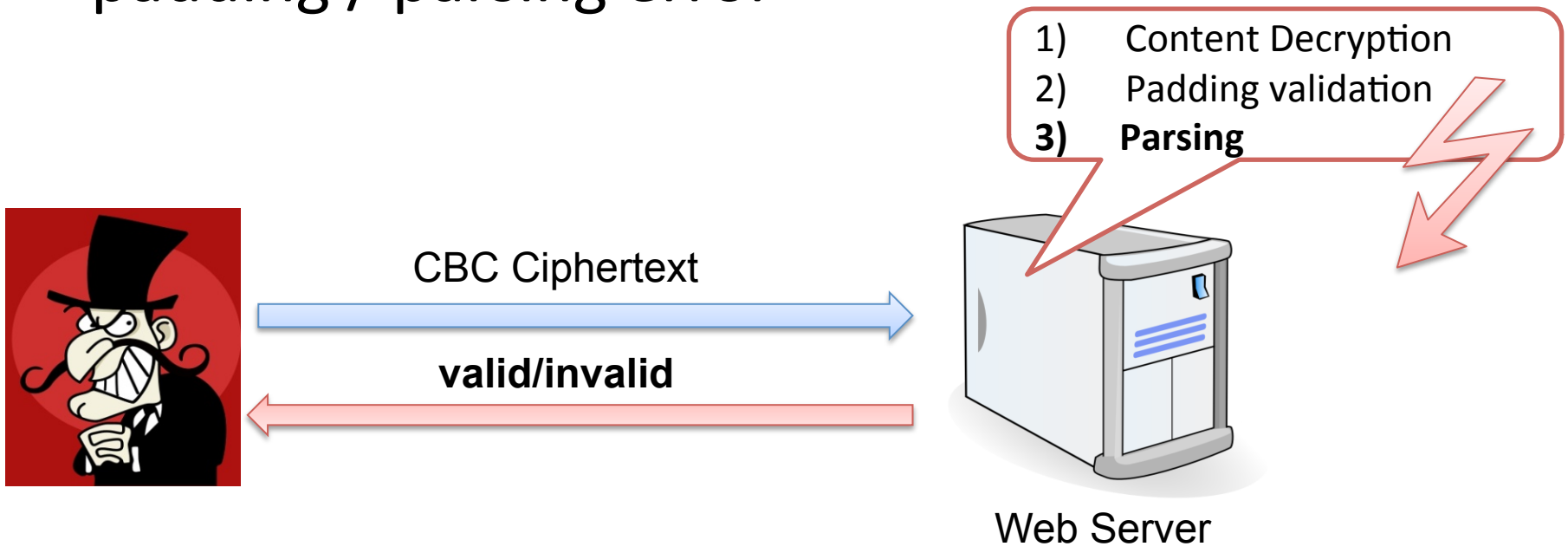
- Obviously…

  If (m1 == m2)

  Then (c1 == c2)

Ingredient 2: CBC ciphertext processing

- Random ciphertexts are decrypted to random plaintexts
- Random plaintexts cause most probably a padding / parsing error



Breaking indistinguishability of AES-GCM

known

guessed

Motivation: transform GCM -> CBC