Practical Attacks on Real World Crypto Implementations

Juraj Somorovsky

Ruhr-Universität Bochum, Hackmanit GmbH

Recent years revealed many crypto attacks…

- ESORICS 2004, Ba… …bility of SSL to Chosen Plaintext Attack

2011 BEAST

- Eurocrypt 2002, Vaudena… …s Induced by CBC Padding—Applications to SSL, IPSEC, … TLS

2013 Lucky13
2014 POODLE

- Crypto 1998, Bleich… …hertext Attacks Against Protocols based on … …yption Standard PKCS #1

2012 XML Encryption
2016 DROWN

ROMA MMXVI
APPSEC EUROPE

Standards updated

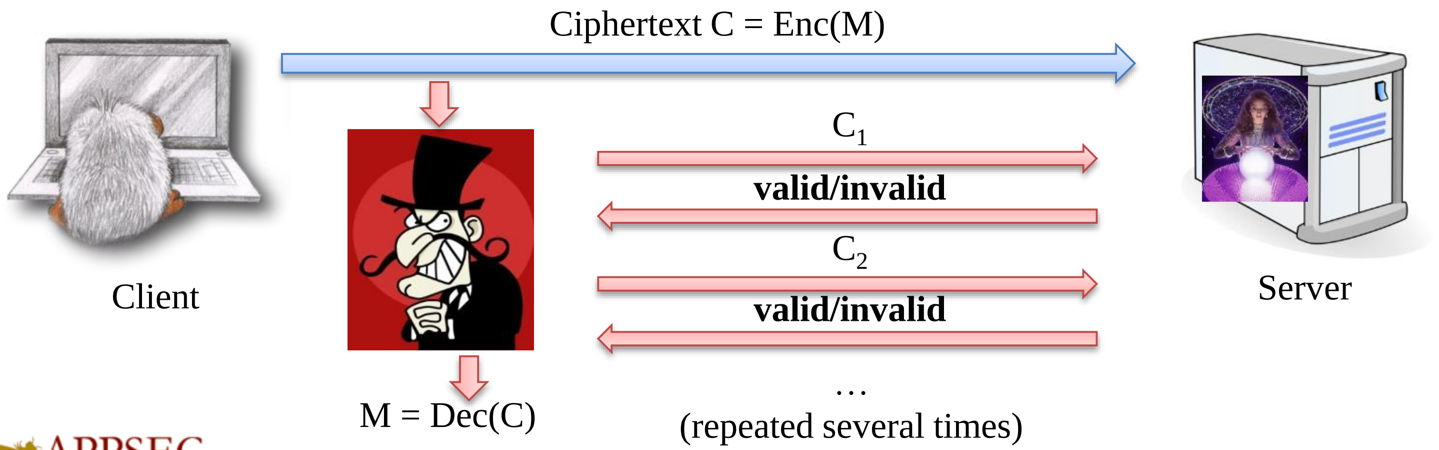
- Countermeasures defined
- What could go wrong in RWC implementations?

Overview

1. Bleichenbacher's Attack
   - XML Encryption
   - TLS
2. Invalid Curve Attack
   - TLS
3. Padding Oracle Attack
   - TLS
4. TLS-Attacker

RSA-PKCS#1 v1.5

- Used to encrypt symmetric keys
- Vulnerable to an adaptive chosen-ciphertext attack



RSA-PKCS#1 v1.5: Countermeasures

1. Use RSA-OAEP (PKCS#1 v2)
2. Apply specific countermeasure

```
generate random
decrypt ciphertext: m = dec(c)
if ( padding correct )
   proceed with m
else
   proceed with random
```

ROMA MMXVI APPSEC EUROPE

Overview

1. **Bleichenbacher's Attack**
   - **XML Encryption**
   - **TLS**
2. **Invalid Curve Attack**
   - **TLS**
3. **Padding Oracle Attack**
   - **TLS**
4. **TLS-Attacker**

RSA PKCS#1 v1.5 in XML Encryption

- Hybrid encryption:

```
k = Dec_pkcs(priv,C1)
m = Dec_aes128(k,C2)
```

1 `Dec_pkcs`

$\downarrow k$

2 `Dec_aes128`

Attack Countermeasure

- Hybrid encryption:

```
k = Dec_pkcs(priv,C1)
m = Dec_aes128(k,C2)
```

1 `Dec_pkcs`

k

Random:
128 b

2 `Dec_aes128`

Case Apache WSS4J

- Hybrid encryption:

```
k = Dec_pkcs(priv,C1)
m = Dec_aes128(k,C2)
```

1 `Dec_pkcs`

k

***Random:* 128 B**

2 `Dec_aes128`

Case Apache WSS4J

- Hybrid encryption:

```
k = Dec_pkcs(priv,C1)
m = Dec_aes128(k,C2)
```

1 `Dec_pkcs`

k

Random:
128 B

2 ~~Dec_aes128~~

Case Apache WSS4J

- Original bug much more complicated
- CVE-2015-0226
- Dennis Kupser, Christian Mainka, Jörg Schwenk, Juraj Somorovsky: **How to Break XML Encryption – Automatically** (WOOT‘15)
- Found automatically using WS-Attacker
- https://github.com/RUB-NDS/WS-Attacker

ROMA MMXVI | APPSEC EUROPE

Overview

1. **Bleichenbacher's Attack**
   - **XML Encryption**
   - **TLS**
2. **Invalid Curve Attack**
   - **TLS**
3. **Padding Oracle Attack**
   - **TLS**
4. **TLS-Attacker**

How About TLS?

- Christopher Meyer, Juraj Somorovsky, Jörg Schwenk, Eugen Weiss, Sebastian Schinzel, Erik Tews: **Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks**. USENIX Security 2014
- Practical attacks on JSSE, Bouncy Castle, Cavium Accelerator
- Bug in OpenSSL

Case JSSE

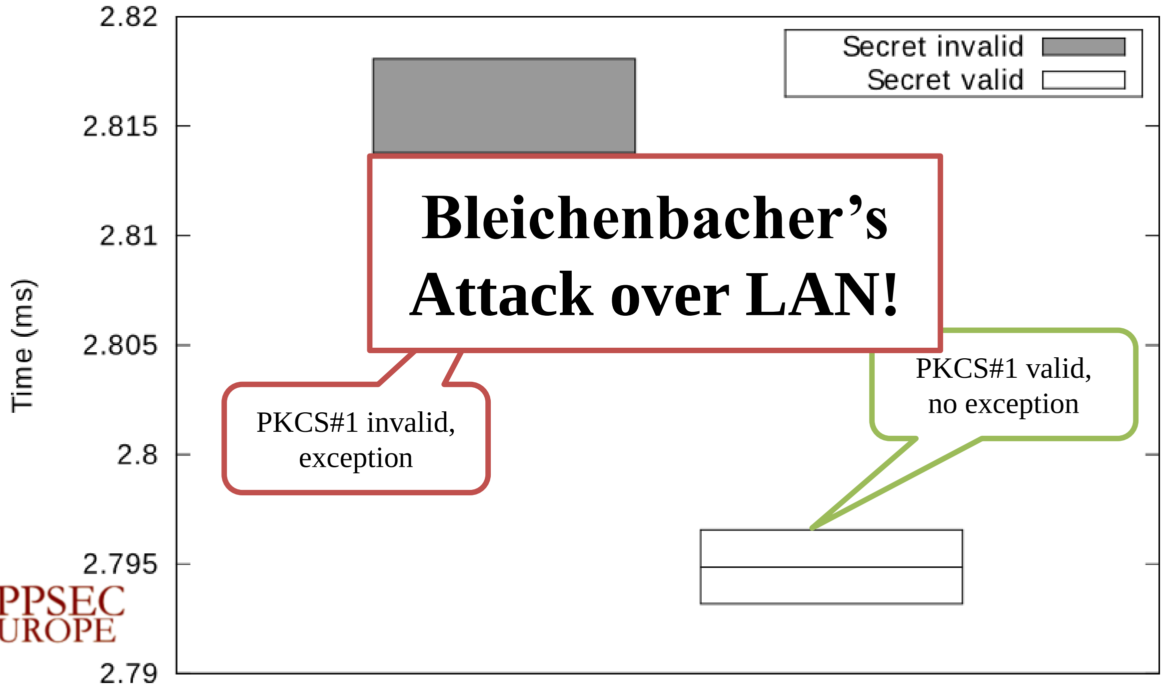
- No direct TLS error messages
- Uses PKCS#1 unpadding function:

```
private byte [] unpadV15 (byte[] padded) {
    if (PKCS valid) {
        return unpadded text;
    } else {
        throw new BadPaddingException();
    }
}
```

- Caught, random generated…what’s wrong?

Case JSSE (CVE-2014-411)

- Exception consumes about 20 microseconds!

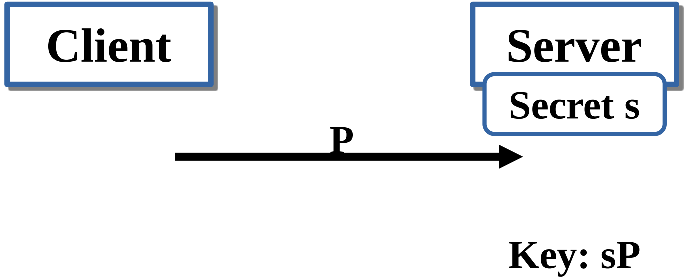


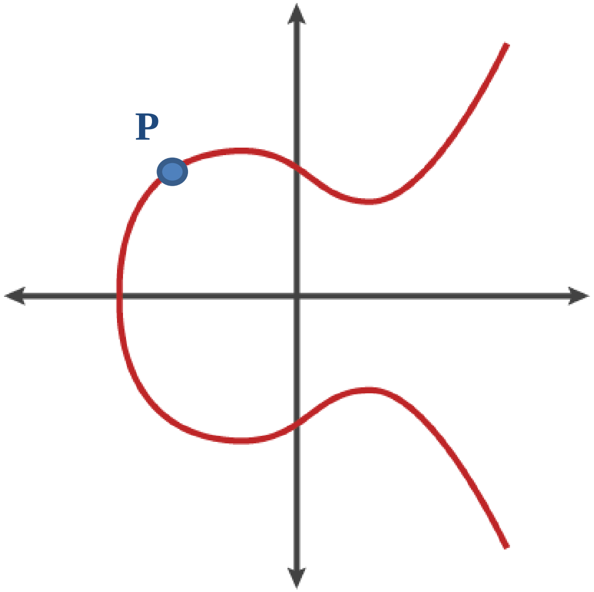
Overview

1. **Bleichenbacher's Attack**
   - **XML Encryption**
   - **TLS**
2. **Invalid Curve Attack**
   - **TLS**
3. **Padding Oracle Attack**
   - **TLS**
4. **TLS-Attacker**

Elliptic Curve

- Set of points over a finite field
- Used e.g. for key exchange





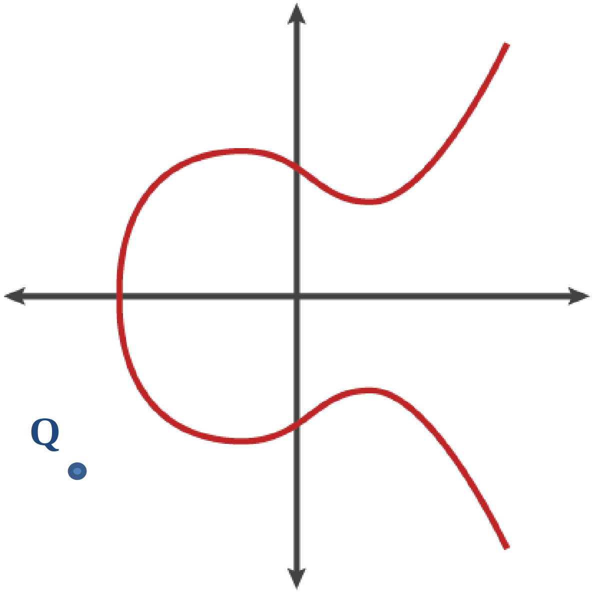
Invalid Curve Attack

- Crypto 2000: Biehl, Meyer, Müller
- Attacker sends an invalid point of small order (e.g. 5)

Q

Server

Secret s



- Attacker computes:

$$s_1 = s \; mod \; 5$$

ROMA MMXVI APPSEC EUROPE

Invalid Curve Attack

- Choose points of small co-prime order (5, 7, 11, …)
- Send to the server
- Compute:

  $s_1 = s \, mod \, 5$

  $s_2 = s \, mod \, 7$

  $s_3 = s \, mod \, 11$

  $s_4 = s \, mod \, 13$

- Compute ***s*** with CRT

Overview

1. **Bleichenbacher's Attack**
   - **XML Encryption**
   - **TLS**
2. **Invalid Curve Attack**
   - **TLS**
3. **Padding Oracle Attack**
   - **TLS**
4. **TLS-Attacker**

Practical Attacks?

- Tibor Jager, Jörg Schwenk, Juraj Somorovsky: **Practical Invalid Curve Attacks on TLS-ECDH.** ESORICS 2015
- Analyzed 8 libraries
- 2 vulnerable
  - Bouncy Castle: 3300 TLS queries
  - Oracle JSSE: 17000 TLS queries
- Further vulnerability found in a Hardware Security module

ROMA MMXVI | APPSEC EUROPE

Impact

- Attacks extract server private keys
- Java servers using EC certificates vulnerable
  - For example Apache Tomcat

Demo

APPSEC EUROPE
ROMA MMXVI

Overview

1. **Bleichenbacher's Attack**
   - **XML Encryption**
   - **TLS**
2. **Invalid Curve Attack**
   - **TLS**
3. **Padding Oracle Attack**
   - **TLS**
4. **TLS-Attacker**

AES

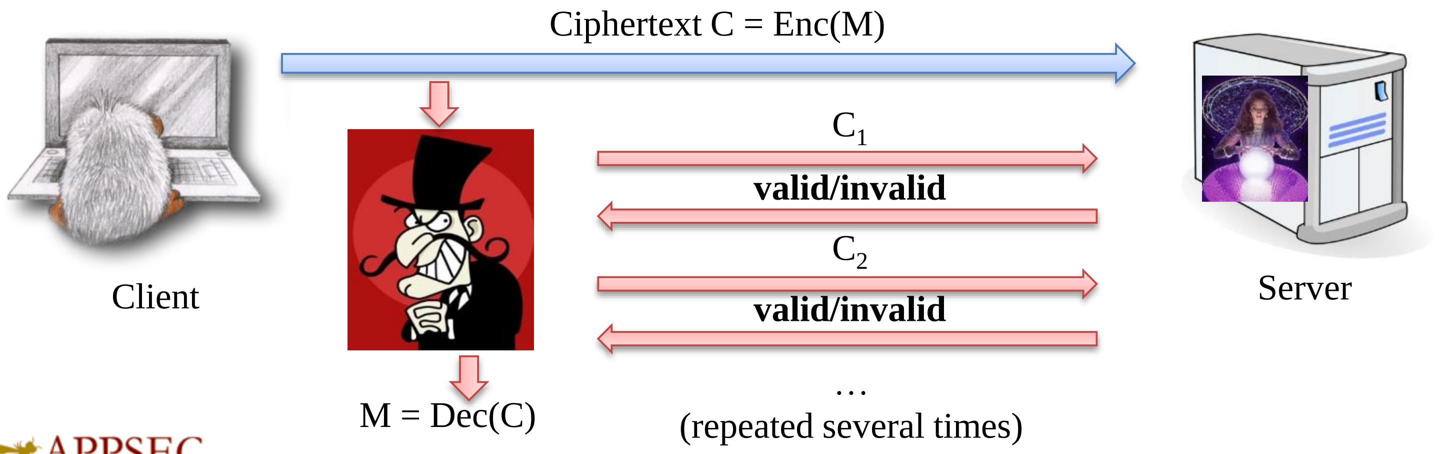
- Block cipher
- Cipher block chaining mode (CBC) of operation
- Last block has to be **padded**
  - Message: HelloOwasp
  - Padding size: 16 – 10 = 6

H	e	l	l	o	O	w	a	s	p	05	05	05	05	05	05

ROMA MMXVI APPSEC EUROPE

AES-CBC

- Vulnerable to an adaptive chosen-ciphertext attack
- Padding oracle attack



Padding Oracle in AES-CBC

- **Flipping** bits in AES-CBC possible

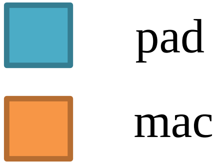
H	e	l	l	o	O	w	a	s	p	05	15	05	05	05	05

- Countermeasure: Authenticated encryption
  - Messages cannot be modified

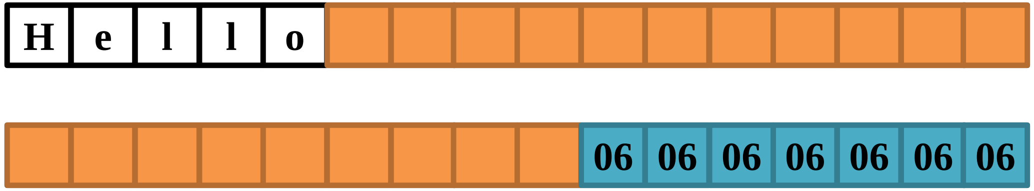
Overview

1. **Bleichenbacher's Attack**
   - **XML Encryption**
   - **TLS**
2. **Invalid Curve Attack**
   - **TLS**
3. **Padding Oracle Attack**
   - **TLS**
4. **TLS-Attacker**

AES-CBC in TLS



- MAC-Pad-Encrypt
- Example:
  - Two blocks
  - Message: Hello
  - MAC size: 20 bytes
  - Padding size: 32 – 5 – 20 = 7



APPSEC EUROPE
ROMA MMXVI

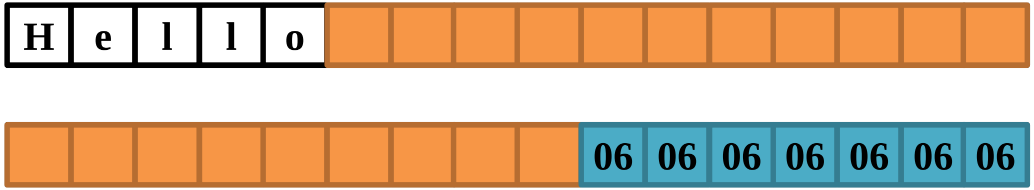
AES-CBC in TLS

pad

mac

- Challenge: not to reveal padding validity
- Always:
  - Padding validation
  - MAC validation
- **Same** error message

What can go wrong?



Botan Vulnerability

```
cbc_decrypt_record(record, record_len);

verify_padding(record, record_len);
if (padding_bad)
      pad_size = 0;

size_t mac_pad_size = mac_size + pad_size;

if(record_len < mac_pad_size)
      throw TLS_Exception(Alert::DECODING_ERROR);

u16bit plaintext_length = record_len - mac_pad_size;

verify_mac(plaintext, plaintext_length, mac_size);

if(mac_bad || padding_bad)
       throw TLS_Exception(Alert::BAD_RECORD_MAC);
```

ROMA MMXVI
APPSEC EURO

Botan Vulnerability

(blue)	pad
(orange)	mac

- Bad padding: **BAD_RECORD_MAC**

H	e	l	l	o											
									16	06	06	06	06	06	06

- Bad MAC: **BAD_RECORD_MAC**

H	e	l	l	o											
									06	06	06	06	06	06	06

- Special case: **Decoding_Error**

				0c	0c	0c	0c	0c	0c	0c	0c	0c	0c	0c	0c	0c

ROMA MMXVI | APPSEC EUROPE

Constant Time Processing

- Timing side-channel can introduce a padding oracle
- **Hard** to exploit
- Patches applied after **Lucky13**

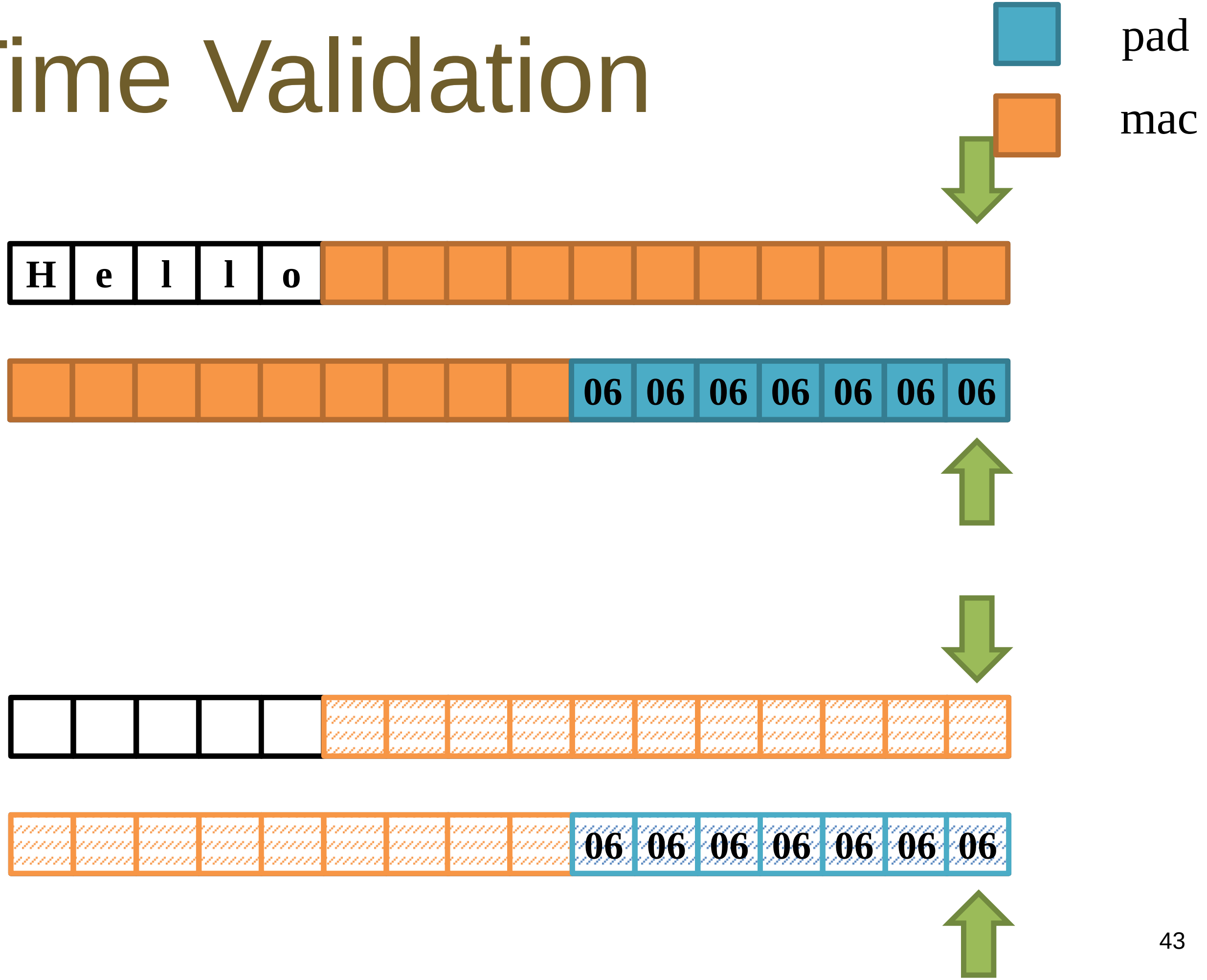
- Let's analyze some of them …

Constant Time Validation

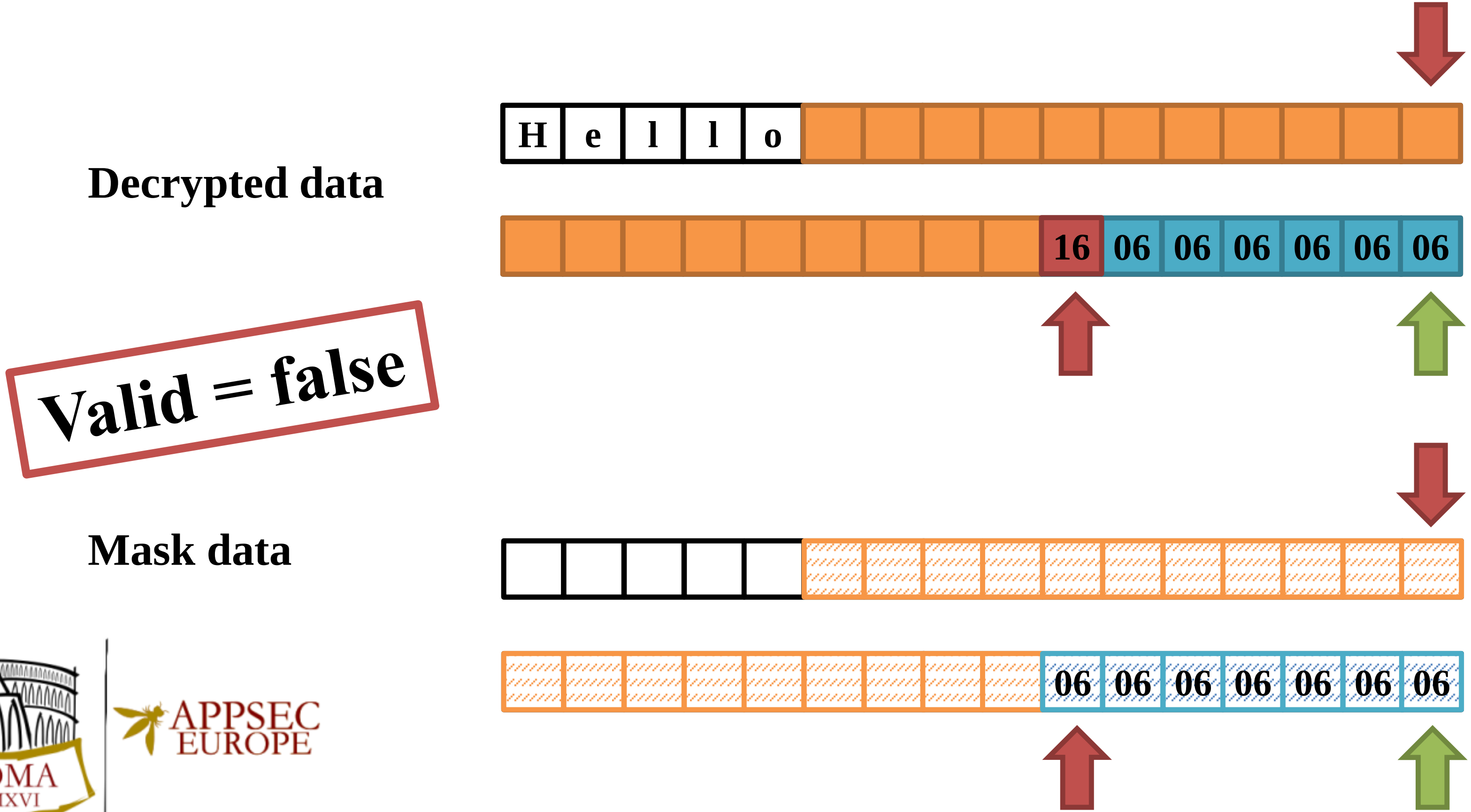
Decrypted data

Valid = true

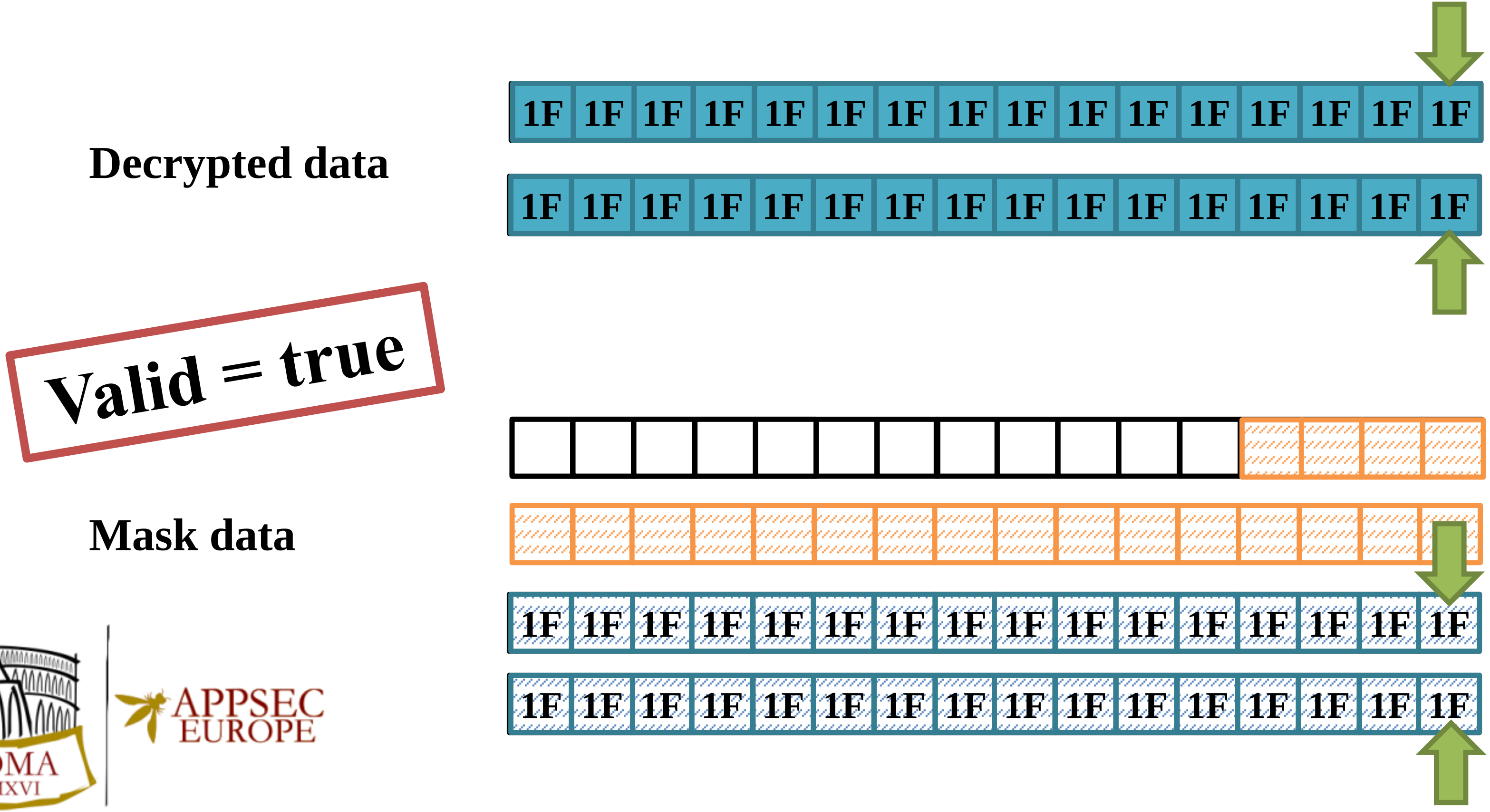
Mask data



Constant Time Validation



OpenSSL Vulnerability



ROMA MMXVI APPSEC EUROPE

OpenSSL Vulnerability

Direct padding oracle

- CVE-2016-2107
- Leads to a different server response

C

RECORD OVERFLOW / BAD RECORD MAC

- Exploitable in BEAST scenarios
  - Decryption of 16 bytes possible

http://web-in-security.blogspot.it/2016/05/curious-padding-oracle-in-openssl-cve.html

MatrixSSL Vulnerability

- Tried to fix Lucky13
- Introduced a buffer overread
- Fixed in 3.8.3
- https://github.com/matrixssl/matrixssl/blob/master/CHANGES.md

ROMA MMXVI | APPSEC EUROPE

Overview

1. **Bleichenbacher's Attack**
    - **XML Encryption**
    - **TLS**
2. **Invalid Curve Attack**
    - **TLS**
3. **Padding Oracle Attack**
    - **TLS**
4. **TLS-Attacker**

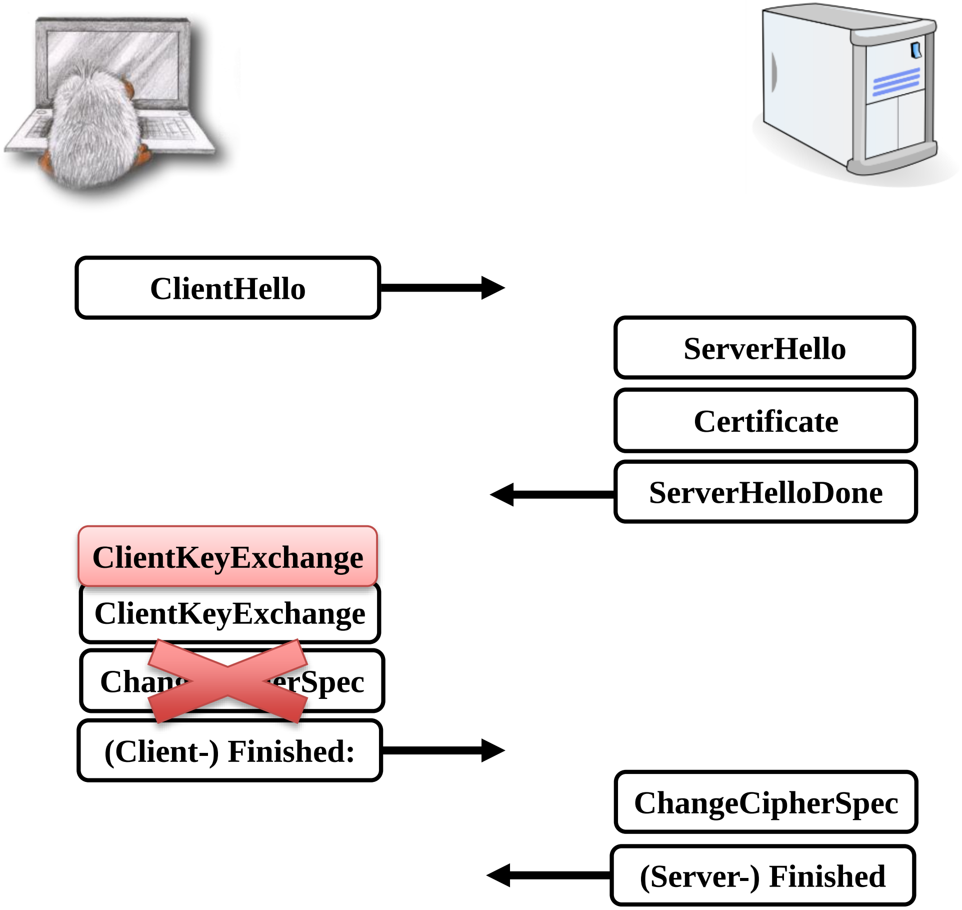
Recent Attacks on TLS

- Not only crypto attacks …
- Buffer overflows / overreads
  - Heartbleed
- Invalid protocol flows
  - FREAK
  - Early CCS
- Tool for flexible protocol executions needed

ROMA MMXVI | APPSEC EUROPE

TLS-Attacker

- Flexible
  - Protocol flow definition
  - Variable modifications



ROMA MMXVI | APPSEC EUROPE

TLS-Attacker

```
WorkflowExecutor workflowExecutor =
configHandler.initializeWorkflowExecutor(transportHandler, tlsContext);

List<ProtocolMessage> protocolMessages = new LinkedList<>();
protocolMessages.add(new ClientHelloMessage(ConnectionEnd.CLIENT));

protocolMessages.add(new ServerHelloMessage(ConnectionEnd.SERVER));
protocolMessages.add(new CertificateMessage(ConnectionEnd.SERVER));
protocolMessages.add(new ServerHelloDoneMessage(ConnectionEnd.SERVER));
protocolMessages.add(new RSAClientKeyExchangeMessage(ConnectionEnd.CLIENT));
protocolMessages.add(new RSAClientKeyExchangeMessage(ConnectionEnd.CLIENT));
protocolMessages.add(new ChangeCipherSpecMessage(ConnectionEnd.CLIENT));
protocolMessages.add(new FinishedMessage(ConnectionEnd.CLIENT));

protocolMessages.add(new ChangeCipherSpecMessage(ConnectionEnd.SERVER));
protocolMessages.add(new FinishedMessage(ConnectionEnd.SERVER));

workflowExecutor.executeWorkflow();
```

TLS-Attacker

- https://github.com/RUB-NDS/TLS-Attacker
- Crypto attacks
- Fuzzing
- Protocol suite framework
- Written in Java

Demo

ROMA MMXVI | APPSEC EUROPE

Conclusions

- Old attacks relevant for RWC implementations
- Old algorithms in the newest standards
  - RSA PKCS#1 v1.5 (attack: 1998)

    2008: TLS 1.2

    2013: XML Encryption 1.1

    2015: JSON Web Encryption
- New add-hoc countermeasures can introduce new flaws

Conclusions

- For standard designers:
  - Remove old crypto
- For developers:
  - Analyze possible side-channels, best practices
    - Check point is on curve
- For pentesters:
  - More tools / analyses of crypto applications needed
  - TLS-Attacker

ROMA MMXVI | APPSEC EUROPE