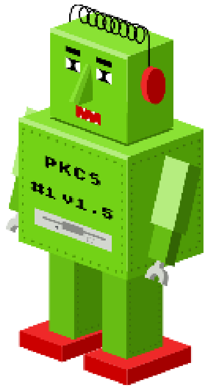


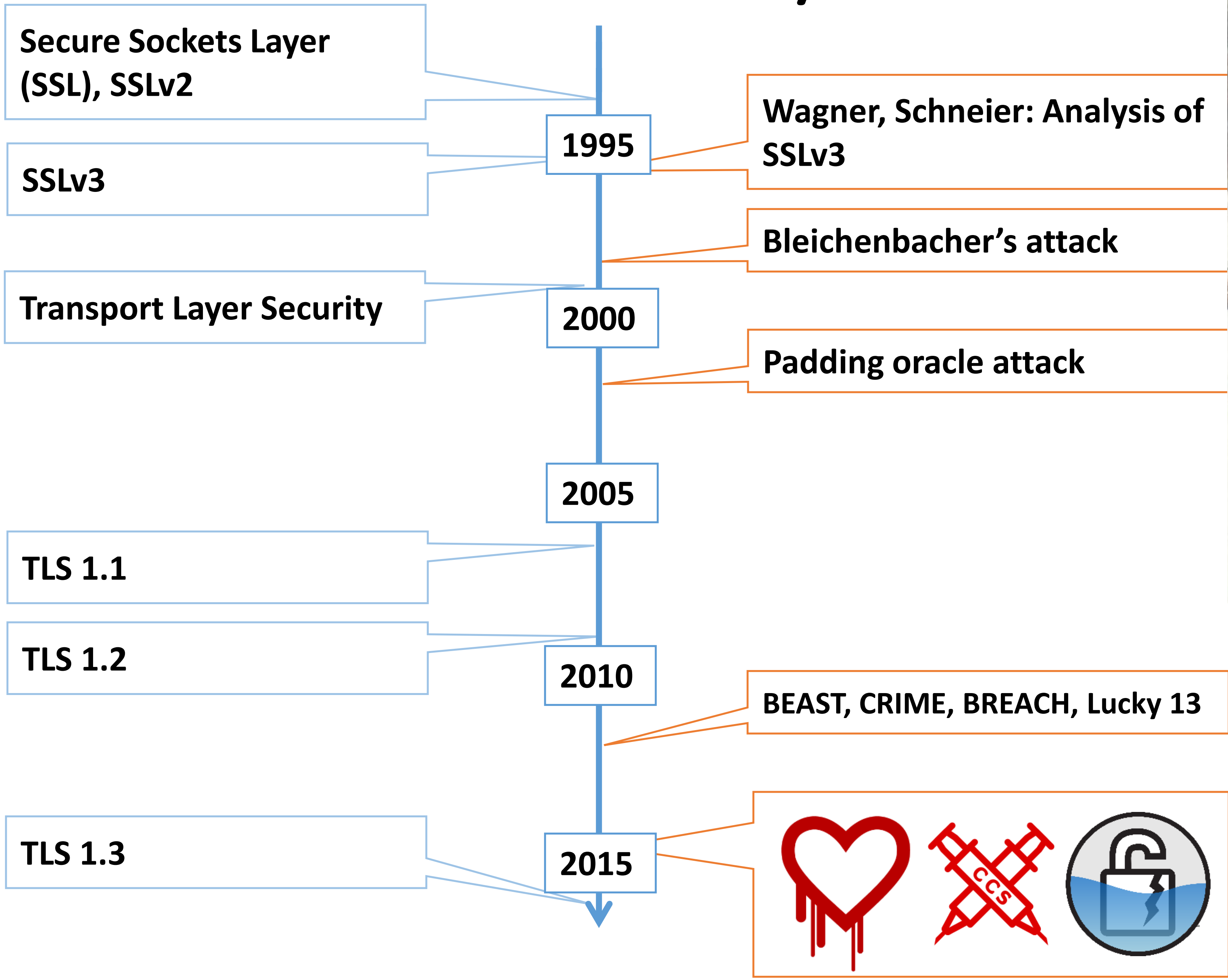
Scalable Scanning and Automatic Classification of TLS Padding Oracle Vulnerabilities

Juraj Somorovsky

RUHR UNIVERSITÄT BOCHUM RUB

HACKMANIT

TLS History



Secure Sockets Layer (SSL), SSLv2

1995

Wagner, Schneier: Analysis of SSLv3

SSLv3

Bleichenbacher's attack

Transport Layer Security

2000

Padding oracle attack

2005

TLS 1.1

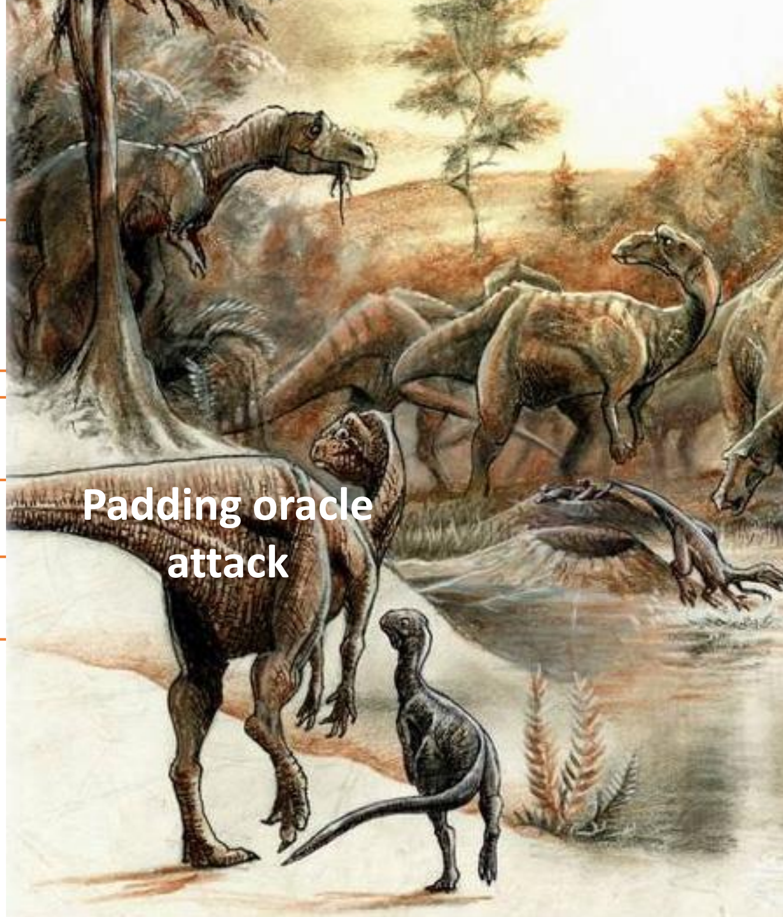
TLS 1.2

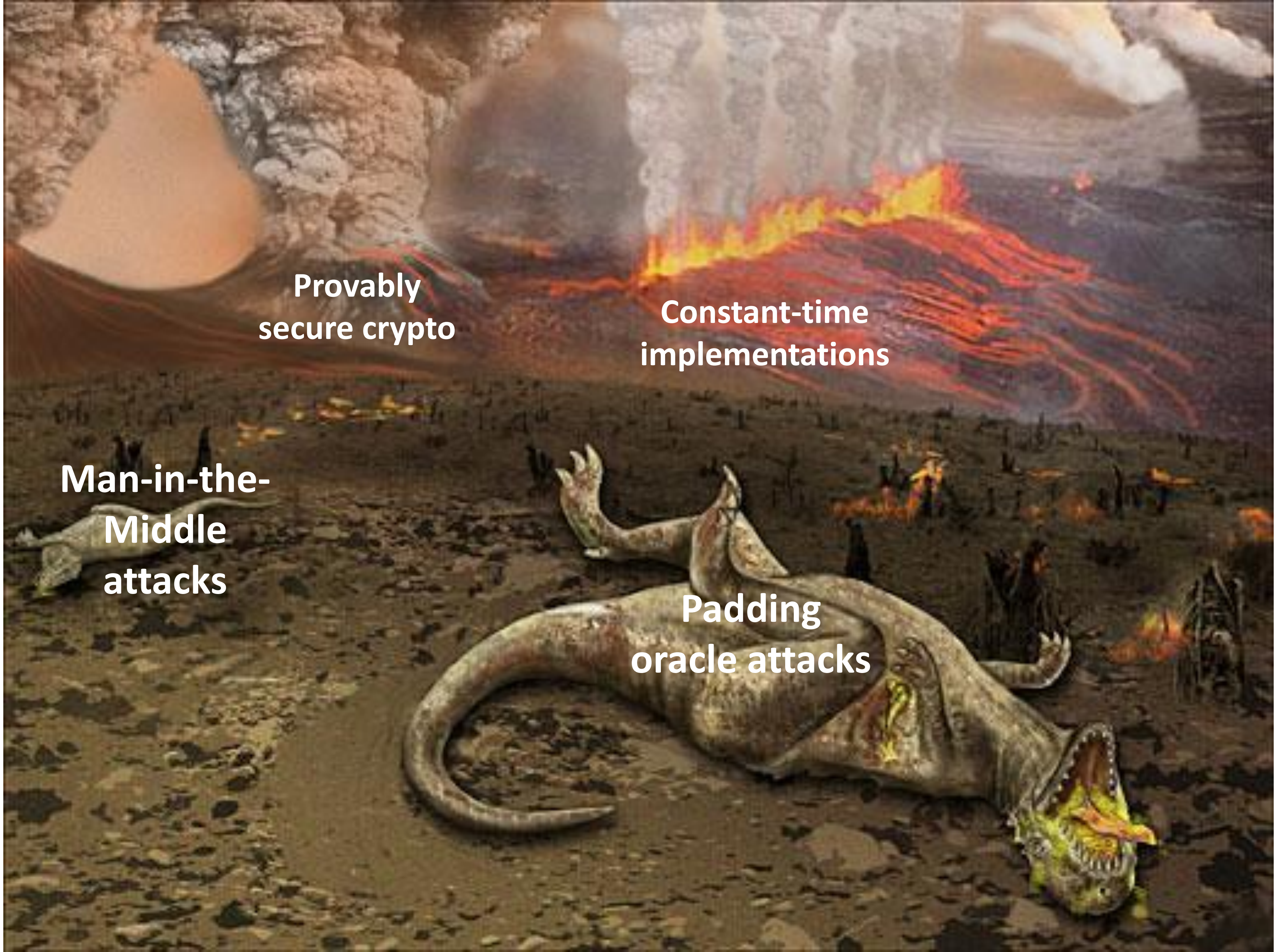
2010

BEAST, CRIME, BREACH, Lucky 13

TLS 1.3

2015



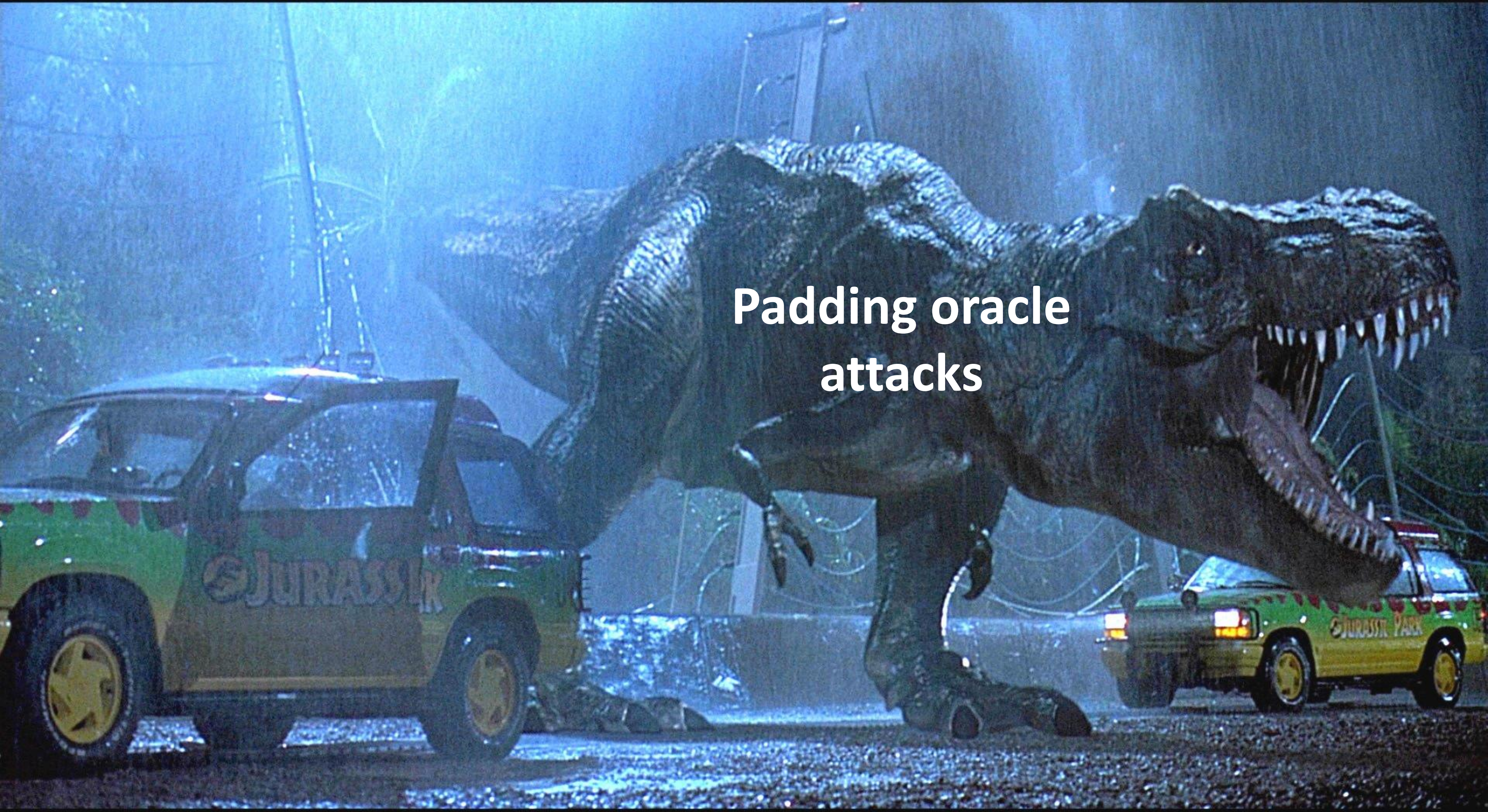


Provably secure crypto

Constant-time implementations

Man-in-the-Middle attacks

Padding oracle attacks



Padding oracle attacks

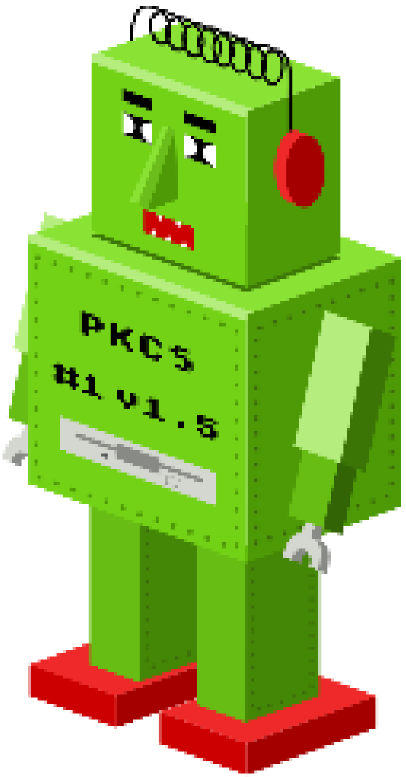
About this talk

- **Return Of Bleichenbacher’s Oracle Threat (ROBOT).** Hanno Böck, Juraj Somorovsky, Craig Young. USENIX Security 2018

- **Scalable Scanning and Automatic Classification of TLS Padding Oracle Vulnerabilities**. Robert Merget, Juraj Somorovsky, Nimrod Aviram, Craig Young, Janis Fliegenschmidt, Jörg Schwenk, Yuval Shavitt. USENIX Security 2019

Overview – ROBOT

1. **Bleichenbacher’s (padding oracle) attack**
2. **How we started – Attack on Facebook**
3. **Performing the scans**
4. **Responsible disclosure**
5. **What did we learn**

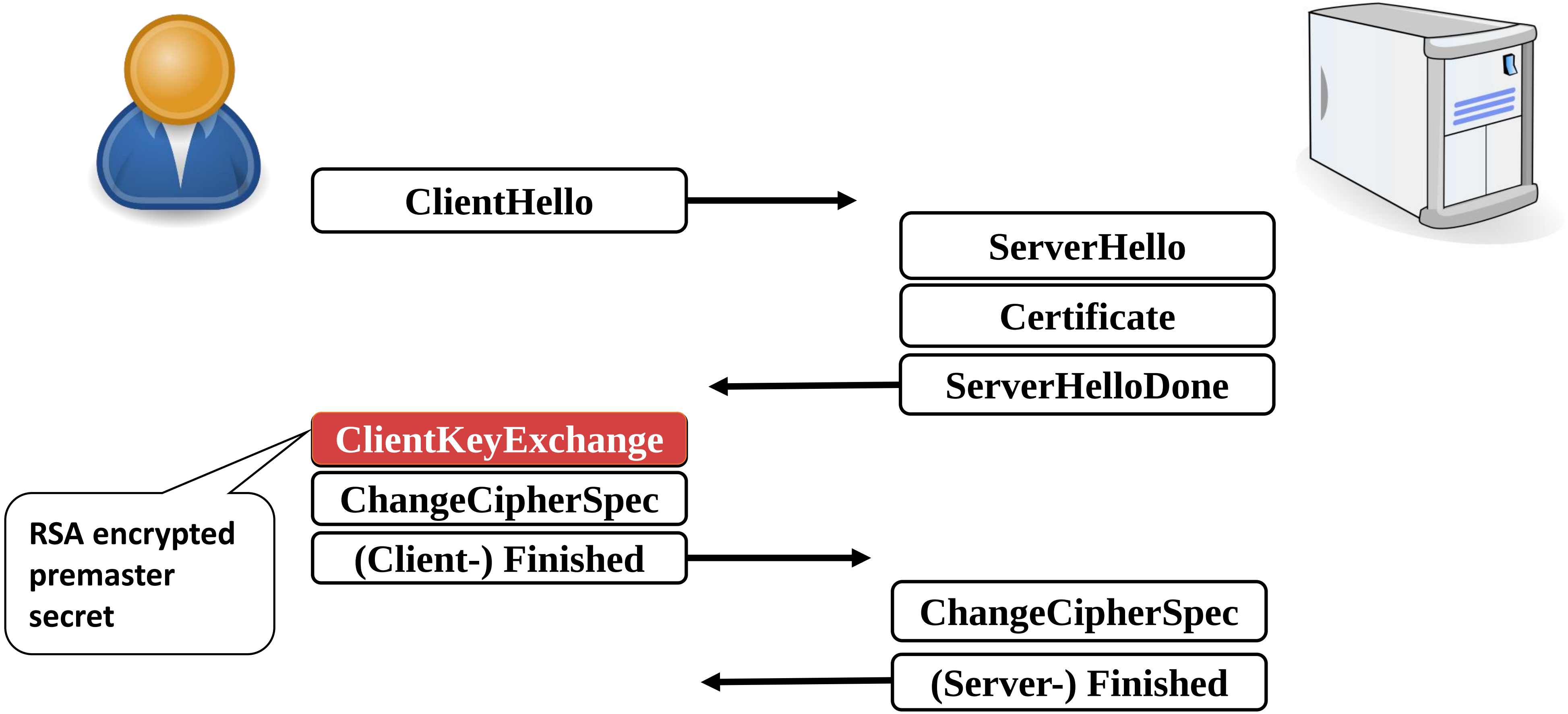


Designed by Ange Albertini

TLS Protocol (High Level Overview)

1. TLS Handshake
   - Selection of algorithm, version, extensions
   - Key exchange: **RSA**, (EC)DH, (EC)DHE
2. Encrypted and authenticated data transport

TLS RSA Handshake



ClientHello

ServerHello

Certificate

ServerHelloDone

ClientKeyExchange

ChangeCipherSpec

(Client-) Finished

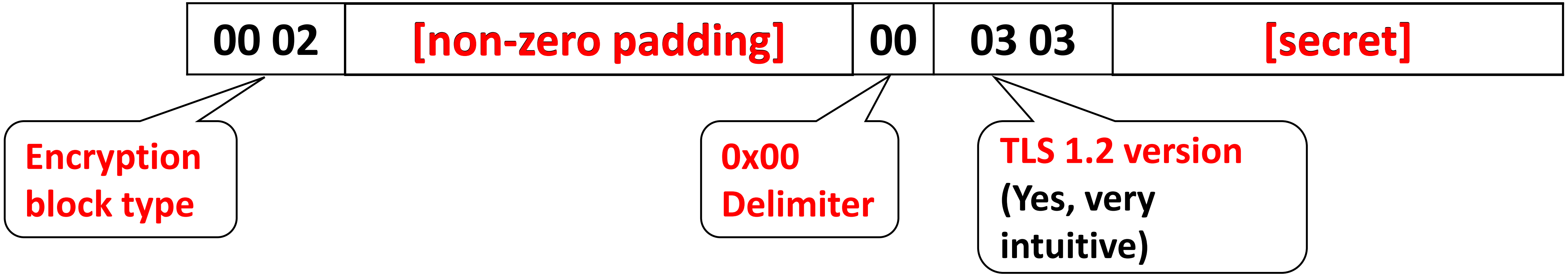
RSA encrypted premaster secret

ChangeCipherSpec

(Server-) Finished

RSA PKCS#1 v1.5

- Used to pad and encrypt the premaster secret:
  - To pad it to the RSA key length
  - To add randomization
- Example for TLS 1.2:



00 02	**[non-zero padding]**	**00**	**03 03**	**[secret]**

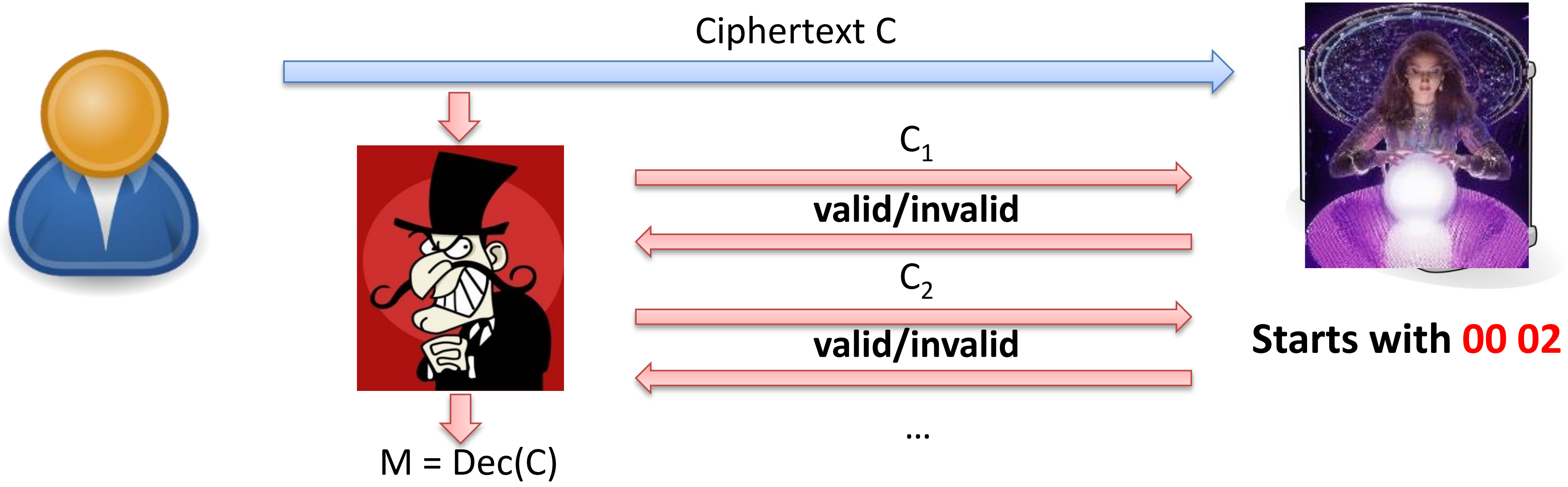
Encryption block type

0x00 Delimiter

TLS 1.2 version (Yes, very intuitive)

Bleichenbacher’s Attack

- 1998: Adaptive chosen-ciphertext attack
- Exploits strict **RSA PKCS#1 v1.5** padding validation

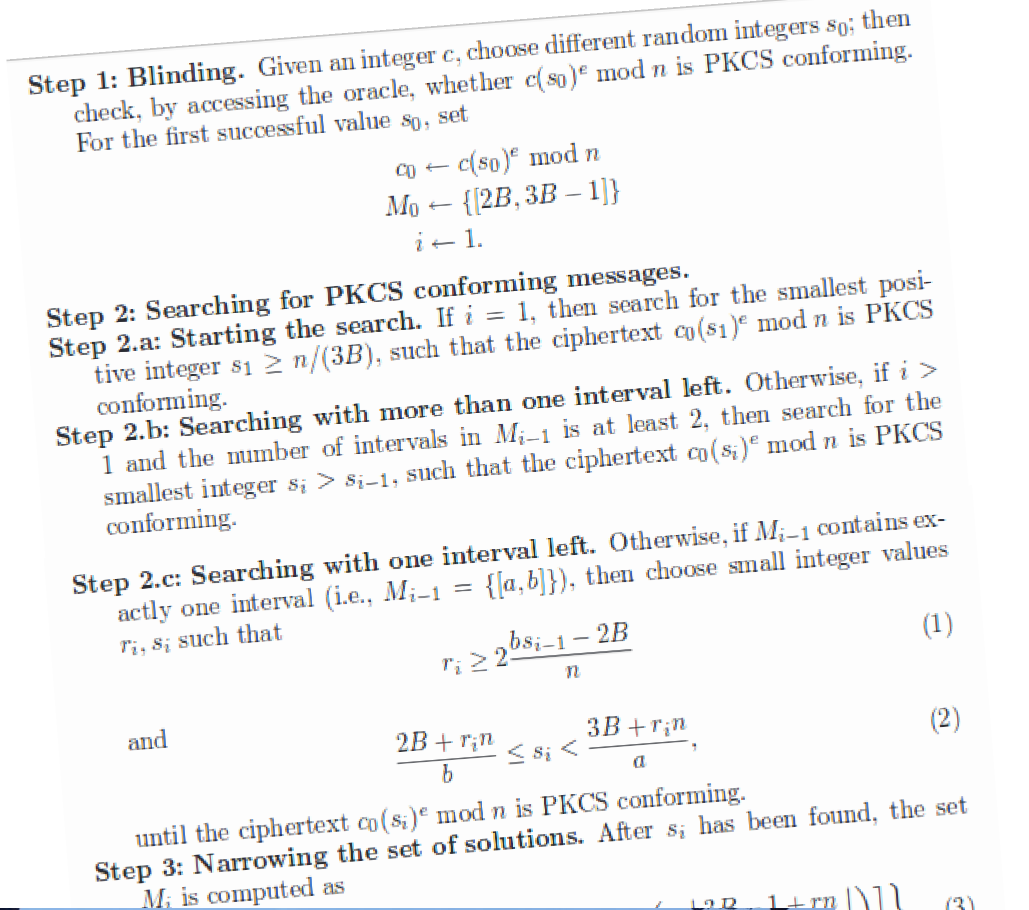


Starts with 00 02 ?

Bleichenbacher's Attack

- The attack needs some math (not going into details here)
- "Million message attack"
  (but could also be faster)

Questions:



Step 1: Blinding. Given an integer c, choose different random integers s_0; then check, by accessing the oracle, whether $c(s_0)^e \bmod n$ is PKCS conforming. For the first successful value s_0, set

$$c_0 \leftarrow c(s_0)^e \bmod n$$
$$M_0 \leftarrow \{[2B, 3B-1]\}$$
$$i \leftarrow 1.$$

Step 2: Searching for PKCS conforming messages.

Step 2.a: Starting the search. If $i = 1$, then search for the smallest positive integer $s_1 \geq n/(3B)$, such that the ciphertext $c_0(s_1)^e \bmod n$ is PKCS conforming.

Step 2.b: Searching with more than one interval left. Otherwise, if $i > 1$ and the number of intervals in M_{i-1} is at least 2, then search for the smallest integer $s_i > s_{i-1}$, such that the ciphertext $c_0(s_i)^e \bmod n$ is PKCS conforming.

Step 2.c: Searching with one interval left. Otherwise, if M_{i-1} contains exactly one interval (i.e., $M_{i-1} = \{[a,b]\}$), then choose small integer values r_i, s_i such that

$$r_i \geq 2\frac{bs_{i-1} - 2B}{n} \qquad (1)$$

and

$$\frac{2B + r_i n}{b} \leq s_i < \frac{3B + r_i n}{a}, \qquad (2)$$

until the ciphertext $c_0(s_i)^e \bmod n$ is PKCS conforming.

Step 3: Narrowing the set of solutions. After s_i has been found, the set M_i is computed as

$$\lfloor 2B - 1 + rn \rfloor)]\} \qquad (3)$$

2B .

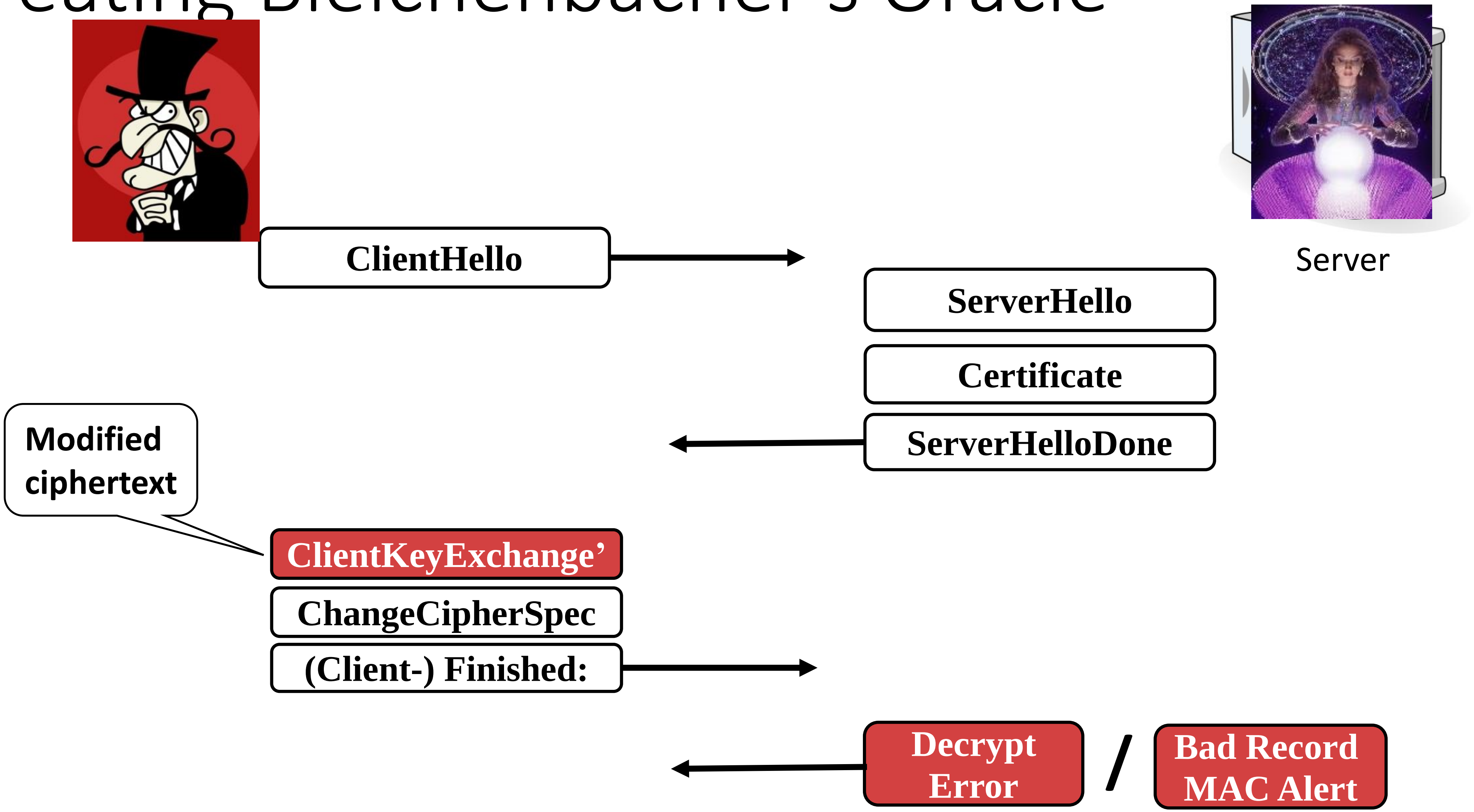
erval of length

m as solution

2.

ing (i.e., when

Creating Bleichenbacher’s Oracle



Server

ClientHello

ServerHello

Certificate

ServerHelloDone

Modified ciphertext

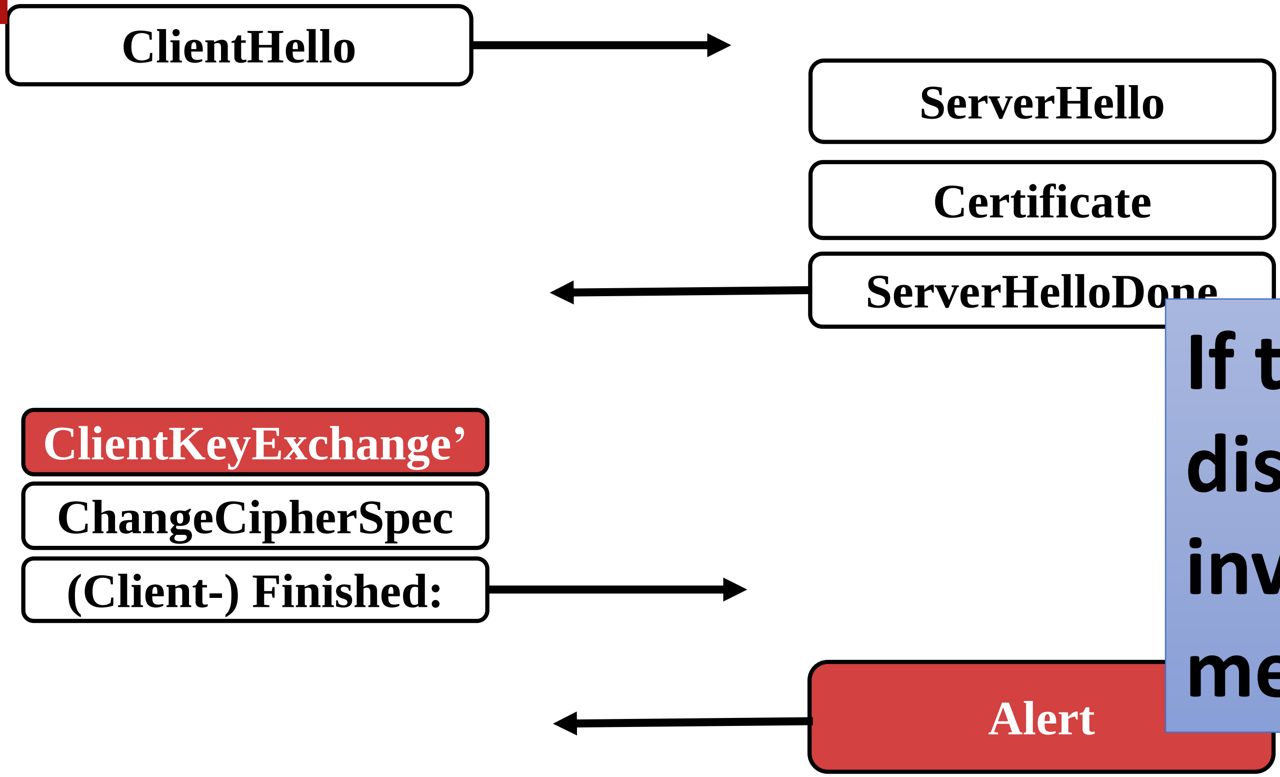
ClientKeyExchange’

ChangeCipherSpec

(Client-) Finished:

Decrypt Error / Bad Record MAC Alert

TLS Countermeasure



ClientHello

ServerHello

Certificate

ServerHelloDone

ClientKeyExchange’

ChangeCipherSpec

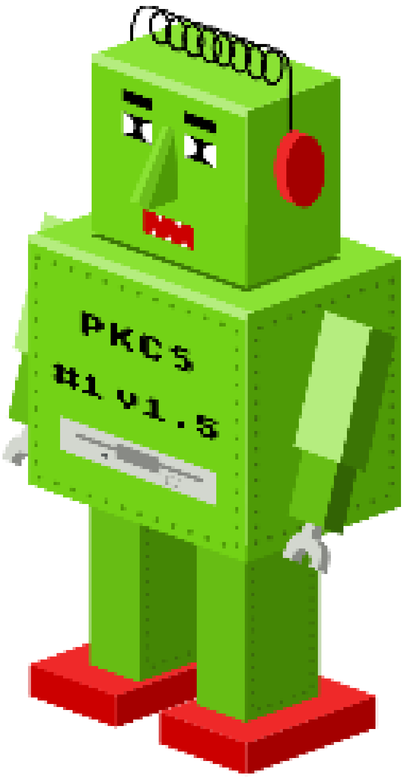
(Client-) Finished:

Alert

If the attacker can distinguish valid / invalid PKCS#1 messages, he wins

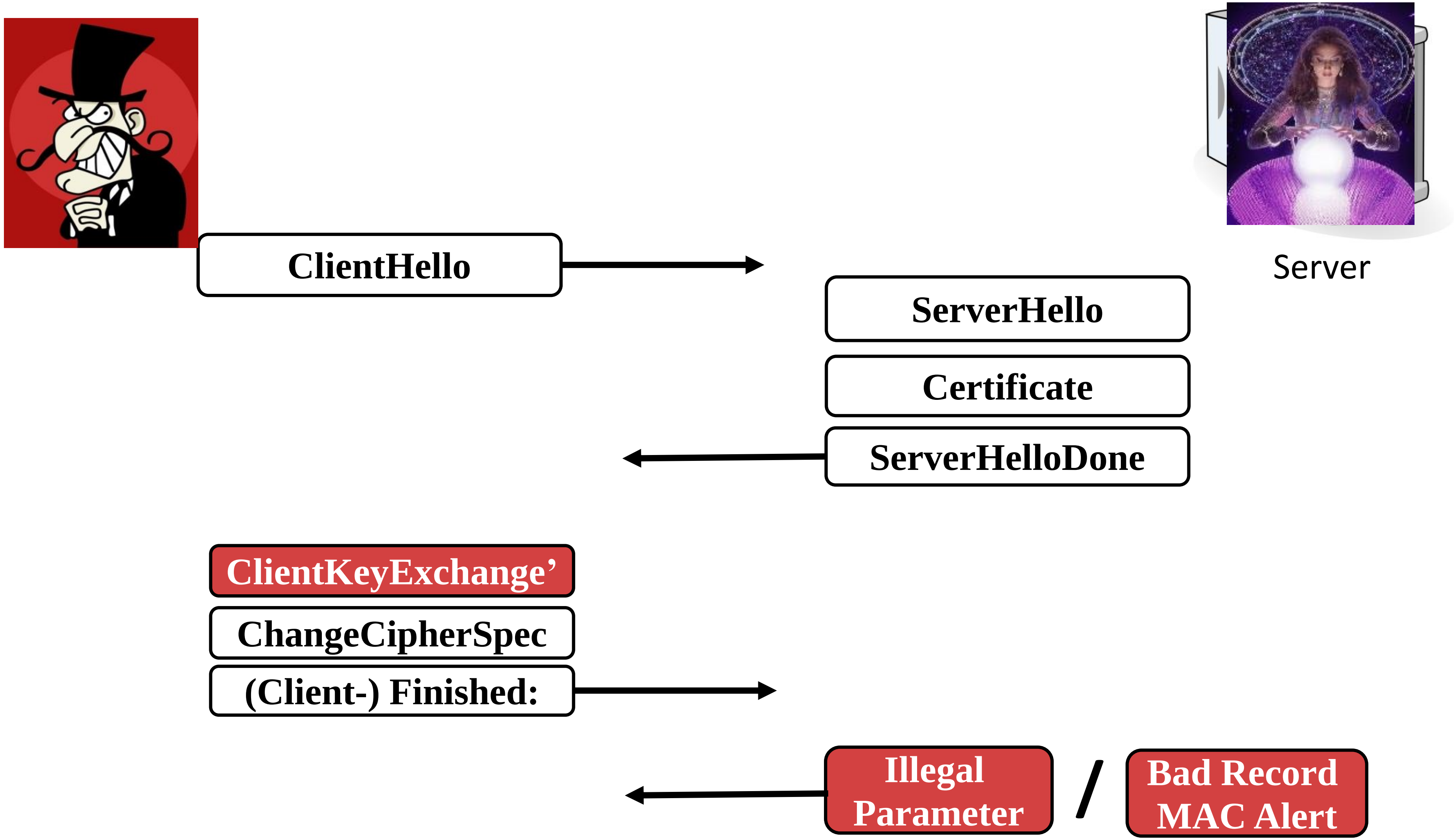
Overview – ROBOT

1. **Bleichenbacher’s (padding oracle) attack**
2. **How we started – Attack on Facebook**
3. **Performing the scans**
4. **Responsible disclosure**
5. **What did we learn**



Designed by Ange Albertini

Hanno Found a Weird Behavior of Facebook



Server

ClientHello

ServerHello

Certificate

ServerHelloDone

ClientKeyExchange’

ChangeCipherSpec

(Client-) Finished:

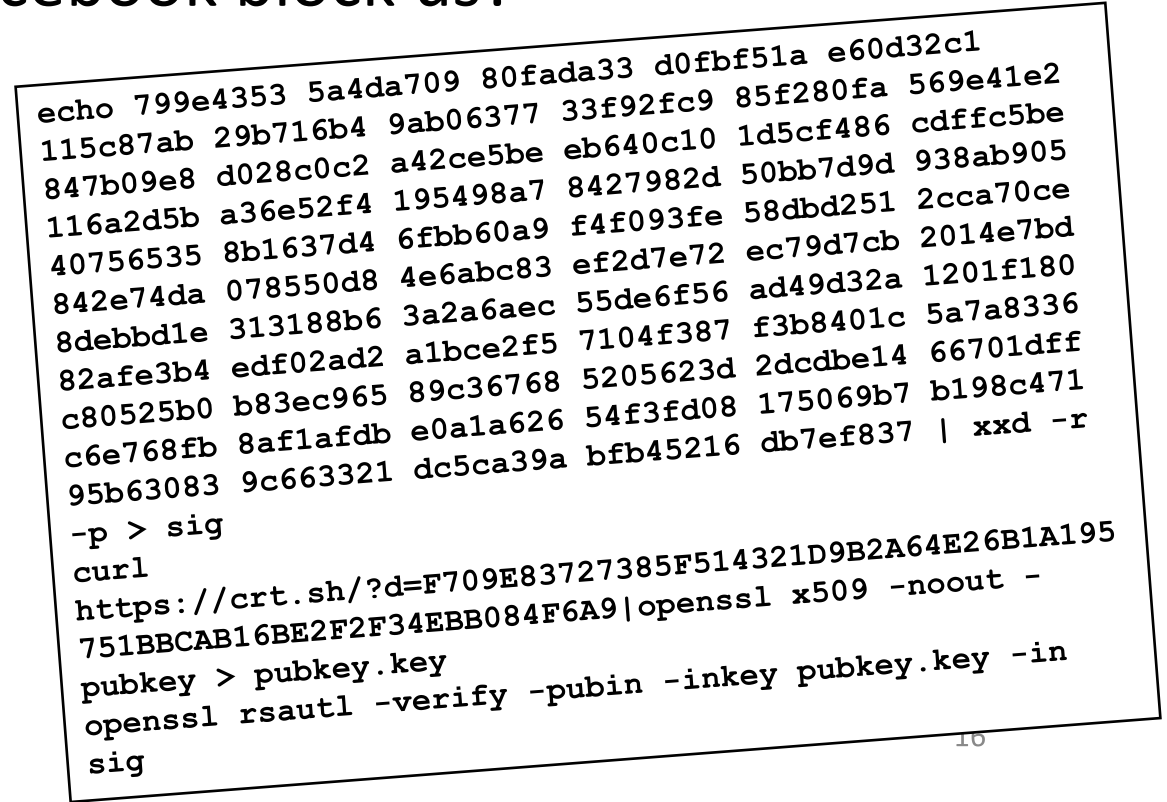
Illegal Parameter / Bad Record MAC Alert

Can We Exploit It?

- Idea: It would be funny to sign a message with Facebook's private key
  - Yes, **signing is possible** as well
- Millions of queries needed…would Facebook block us?
- Successful after several tries:

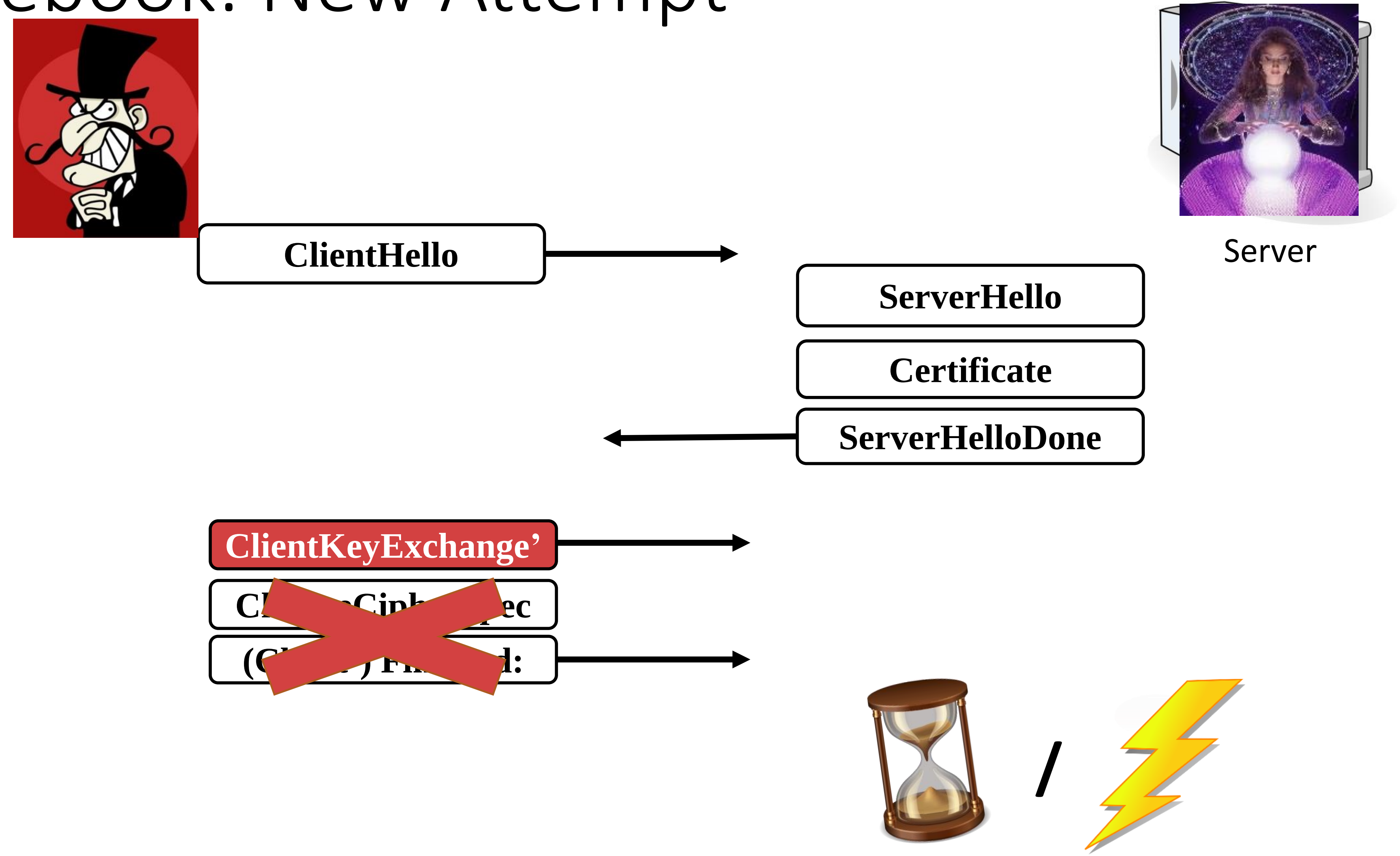
  "We hacked Facebook with a Bleichenbacher Oracle (JS/HB)."

- Facebook fixed



```
echo 799e4353 5a4da709 80fada33 d0fbf51a e60d32c1
115c87ab 29b716b4 9ab06377 33f92fc9 85f280fa 569e41e2
847b09e8 d028c0c2 a42ce5be eb640c10 1d5cf486 cdffc5be
116a2d5b a36e52f4 195498a7 8427982d 50bb7d9d 938ab905
40756535 8b1637d4 6fbb60a9 f4f093fe 58dbd251 2cca70ce
842e74da 078550d8 4e6abc83 ef2d7e72 ec79d7cb 2014e7bd
8debbd1e 313188b6 3a2a6aec 55de6f56 ad49d32a 1201f180
82afe3b4 edf02ad2 a1bce2f5 7104f387 f3b8401c 5a7a8336
c80525b0 b83ec965 89c36768 5205623d 2dcdbe14 66701dff
c6e768fb 8af1afdb e0a1a626 54f3fd08 175069b7 b198c471
95b63083 9c663321 dc5ca39a bfb45216 db7ef837 | xxd -r
-p > sig
curl
https://crt.sh/?d=F709E83727385F514321D9B2A64E26B1A195
751BBCAB16BE2F2F34EBB084F6A9|openssl x509 -noout -
pubkey > pubkey.key
openssl rsautl -verify -pubin -inkey pubkey.key -in
sig
```

Facebook: New Attempt



Server

ClientHello

ServerHello

Certificate

ServerHelloDone

ClientKeyExchange’

~~ChangeCipherSpec~~

~~(Client) Finished:~~

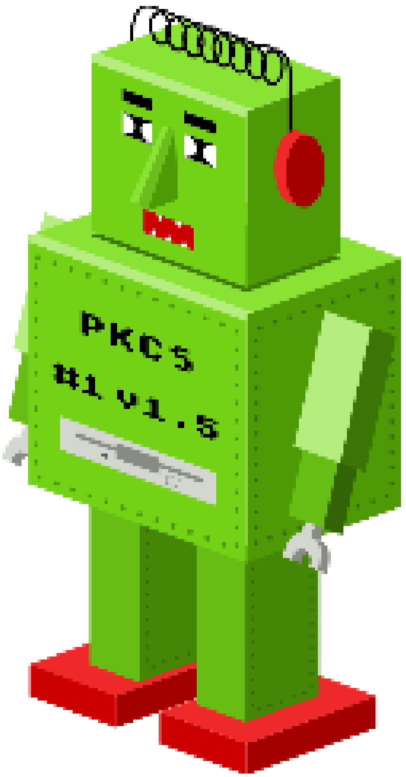
/

Facebook Fixed Again

- This is interesting. So how about other servers?

Overview – ROBOT

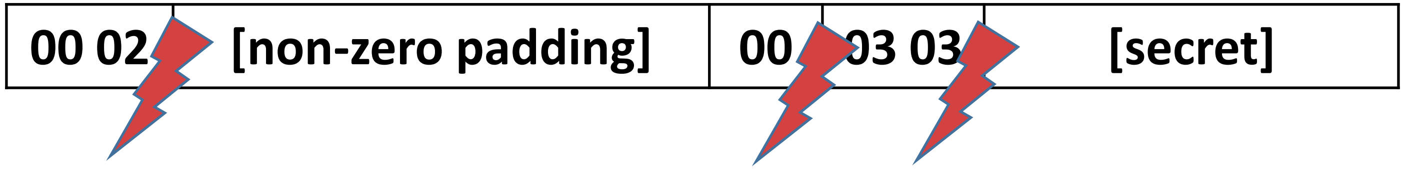
1. **Bleichenbacher’s (padding oracle) attack**
2. **How we started – Attack on Facebook**
3. **Performing the scans**
4. **Responsible disclosure**
5. **What did we learn**



Designed by Ange Albertini

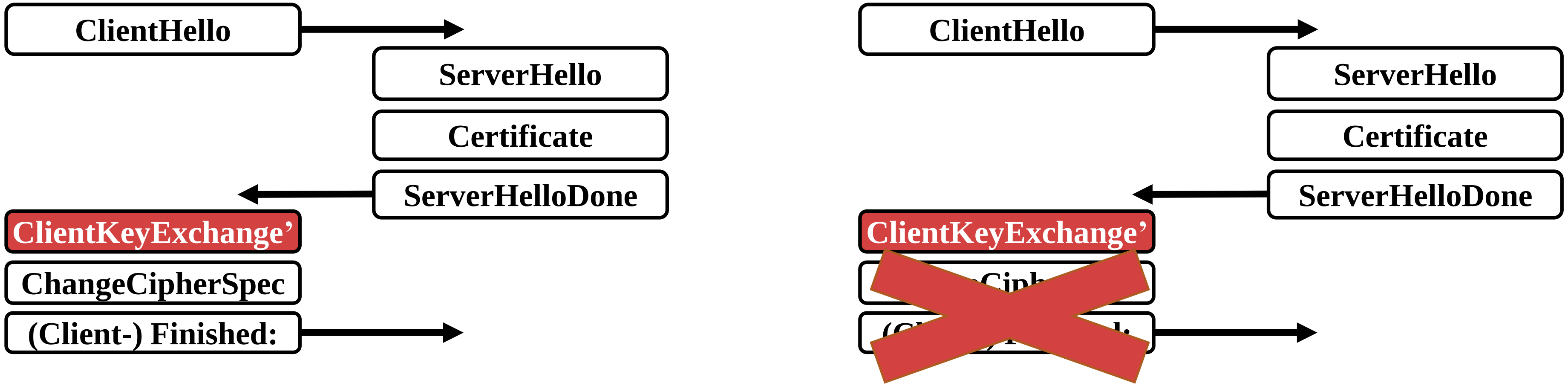
Let's Start Scanning

- Careful selection of ClientKeyExchange messages:
  - Wrong TLS version
  - Wrong padding length
  - Not starting with 0x00 02



00 02	[non-zero padding]	00	03 03	[secret]

- Full / Shortened TLS handshakes:



ClientHello

ServerHello

Certificate

ServerHelloDone

ClientKeyExchange'

ChangeCipherSpec

(Client-) Finished:

ClientHello

ServerHello

Certificate

ServerHelloDone

ClientKeyExchange'

Alexa Top 1 Million Scan

- 2,8 % vulnerable
- PayPal, Apple, ebay, Cisco, …
- Different behaviors…different combinations:

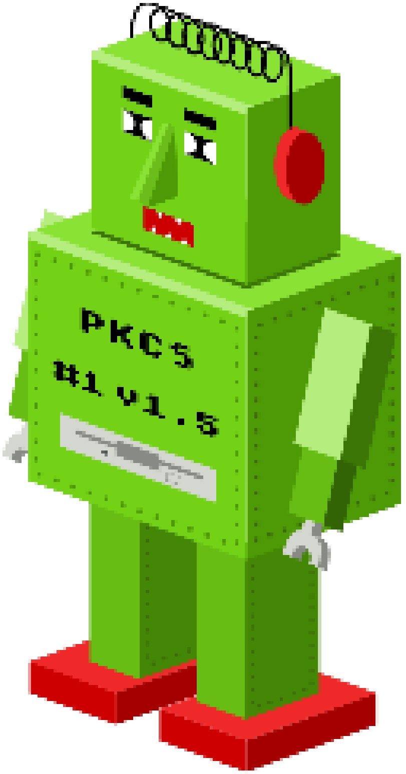
TCP connection resets ⚡ Timeouts ⌛

Different alerts Illegal Parameter / Bad Record MAC Alert / Handshake Failure / Internal Error /..

Duplicate alerts Alert / Alert Alert

Overview – ROBOT

1. **Bleichenbacher’s (padding oracle) attack**
2. **How we started – Attack on Facebook**
3. **Performing the scans**
4. **Responsible disclosure**
5. **What did we learn**



Designed by Ange Albertini

Who Is Responsible for These Mistakes?

- Reporting is not always that easy …

Your server is vulnerable
to Bleichenbacher‘s attack.

No worries, we use
millitary grade encryption.

Don’t Fix for Some Vendors … Cisco ACE

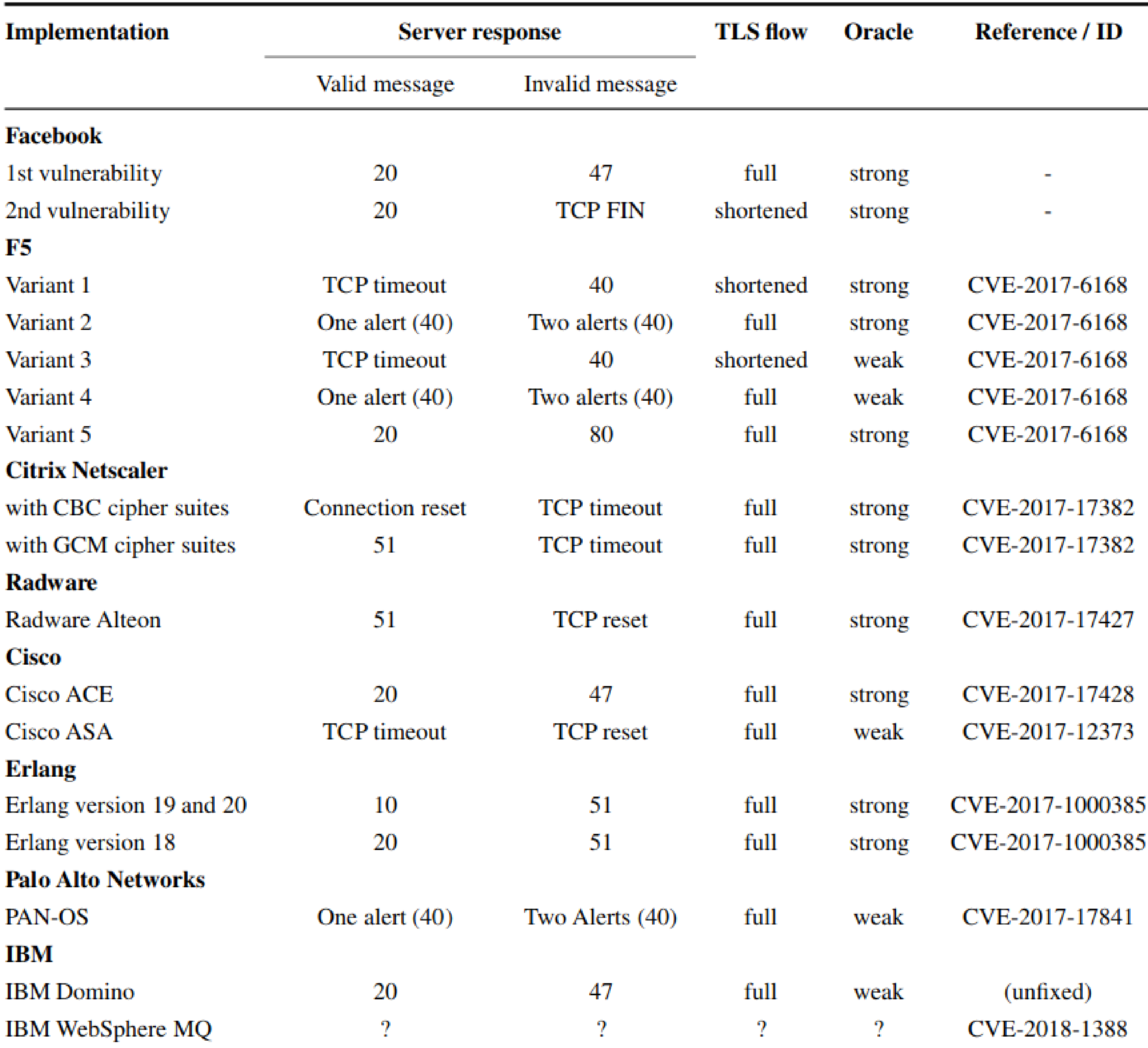
- Supports only TLS RSA
- Cisco: We won't fix it, it's out of support for several years

- But there were plenty of webpages still running with these devices

Like cisco.com

Identified (Most of) Them

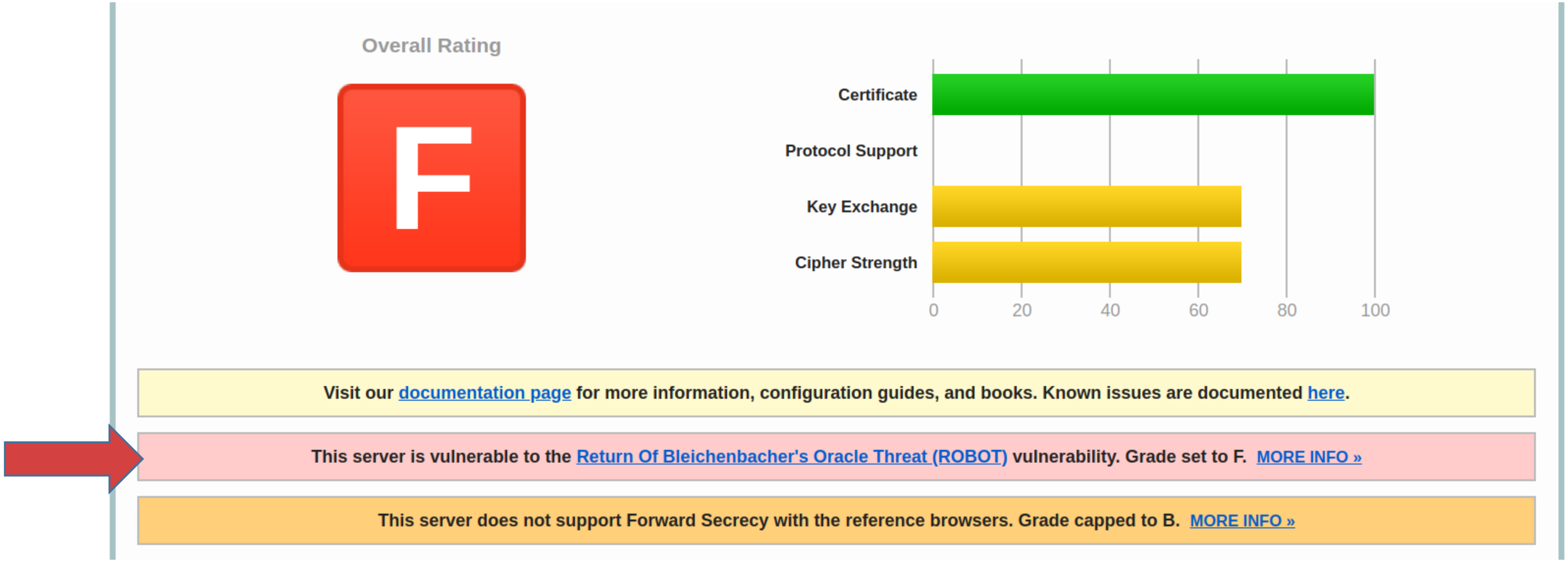
F5 had 5 different vulnerabilities!



Implementation	Server response		TLS flow	Oracle	Reference / ID
	Valid message	Invalid message			
Facebook					
1st vulnerability	20	47	full	strong	-
2nd vulnerability	20	TCP FIN	shortened	strong	-
F5					
Variant 1	TCP timeout	40	shortened	strong	CVE-2017-6168
Variant 2	One alert (40)	Two alerts (40)	full	strong	CVE-2017-6168
Variant 3	TCP timeout	40	shortened	weak	CVE-2017-6168
Variant 4	One alert (40)	Two alerts (40)	full	weak	CVE-2017-6168
Variant 5	20	80	full	strong	CVE-2017-6168
Citrix Netscaler					
with CBC cipher suites	Connection reset	TCP timeout	full	strong	CVE-2017-17382
with GCM cipher suites	51	TCP timeout	full	strong	CVE-2017-17382
Radware					
Radware Alteon	51	TCP reset	full	strong	CVE-2017-17427
Cisco					
Cisco ACE	20	47	full	strong	CVE-2017-17428
Cisco ASA	TCP timeout	TCP reset	full	weak	CVE-2017-12373
Erlang					
Erlang version 19 and 20	10	51	full	strong	CVE-2017-1000385
Erlang version 18	20	51	full	strong	CVE-2017-1000385
Palo Alto Networks					
PAN-OS	One alert (40)	Two Alerts (40)	full	weak	CVE-2017-17841
IBM					
IBM Domino	20	47	full	weak	(unfixed)
IBM WebSphere MQ	?	?	?	?	CVE-2018-1388

Test Tools

- No easily usable test tool for Bleichenbacher attacks available
- Currently implemented in SSL Labs, testssl.sh, TLS-Attacker, tlsfuzzer



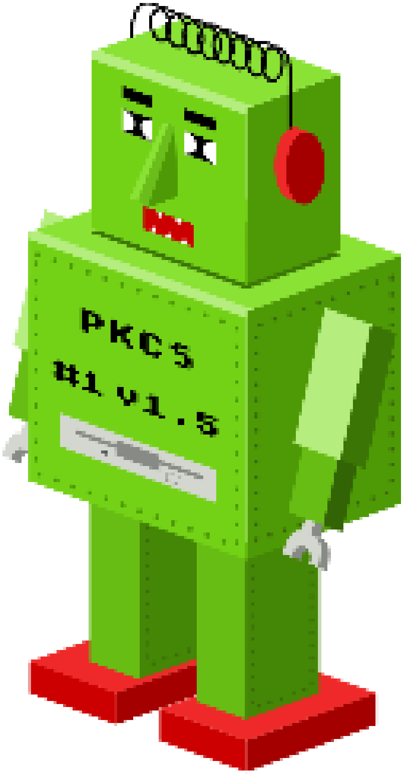
Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server is vulnerable to the Return Of Bleichenbacher's Oracle Threat (ROBOT) vulnerability. Grade set to F. MORE INFO »

This server does not support Forward Secrecy with the reference browsers. Grade capped to B. MORE INFO »

Overview – ROBOT

1. **Bleichenbacher's (padding oracle) attack**
2. **How we started – Attack on Facebook**
3. **Performing the scans**
4. **Responsible disclosure**
5. **What did we learn**



Designed by Ange Albertini

What Did We Learn?

- **20** year old attacks still work
- New side channels
  - Timeouts
  - TCP resets
  - Duplicated alerts

- How about scanning for other vulnerabilities?

About this talk

- **Return Of Bleichenbacher’s Oracle Threat (ROBOT).** Hanno Böck, Juraj Somorovsky, Craig Young. USENIX Security 2018

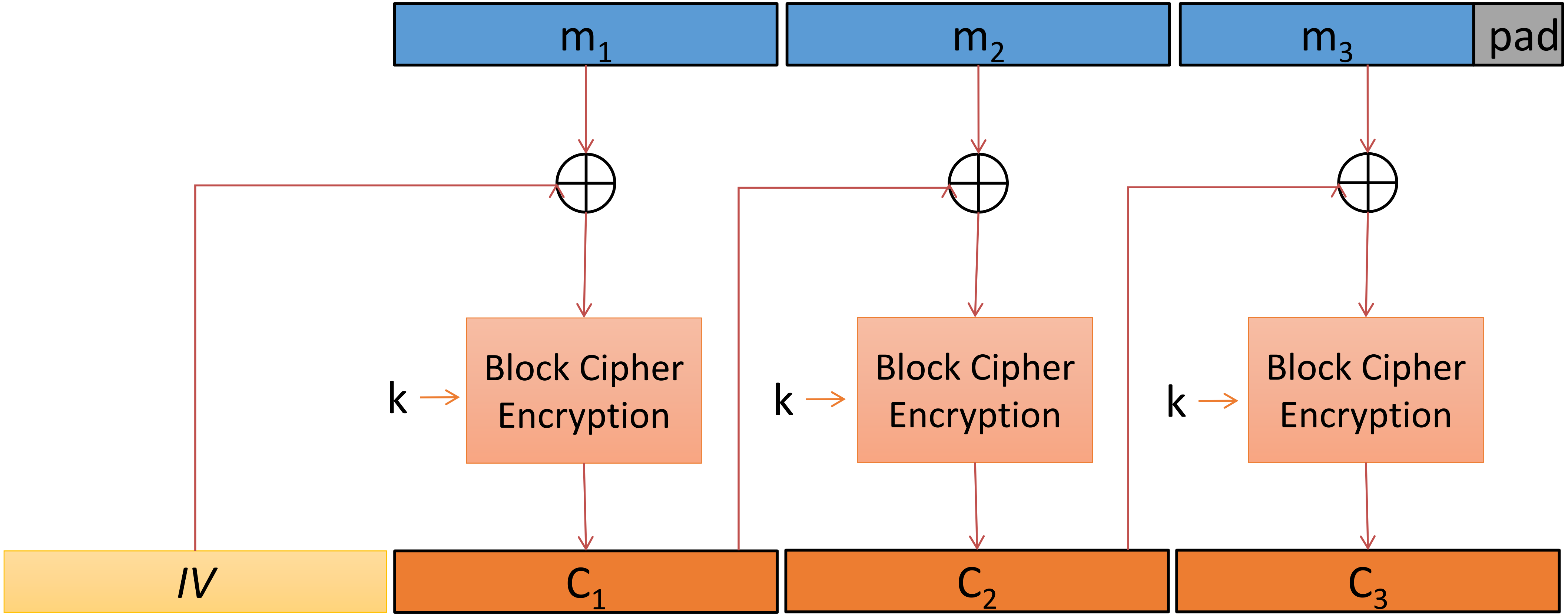
- **Scalable Scanning and Automatic Classification of TLS Padding Oracle Vulnerabilities**. Robert Merget, Juraj Somorovsky, Nimrod Aviram, Craig Young, Janis Fliegenschmidt, Jörg Schwenk, Yuval Shavitt. USENIX Security 2019

  - Can we effectively scan for CBC oracles at scale?
  - Can we also identify vulnerabilities and avoid false positives/negatives?
  - Are these vulnerabilities exploitable?

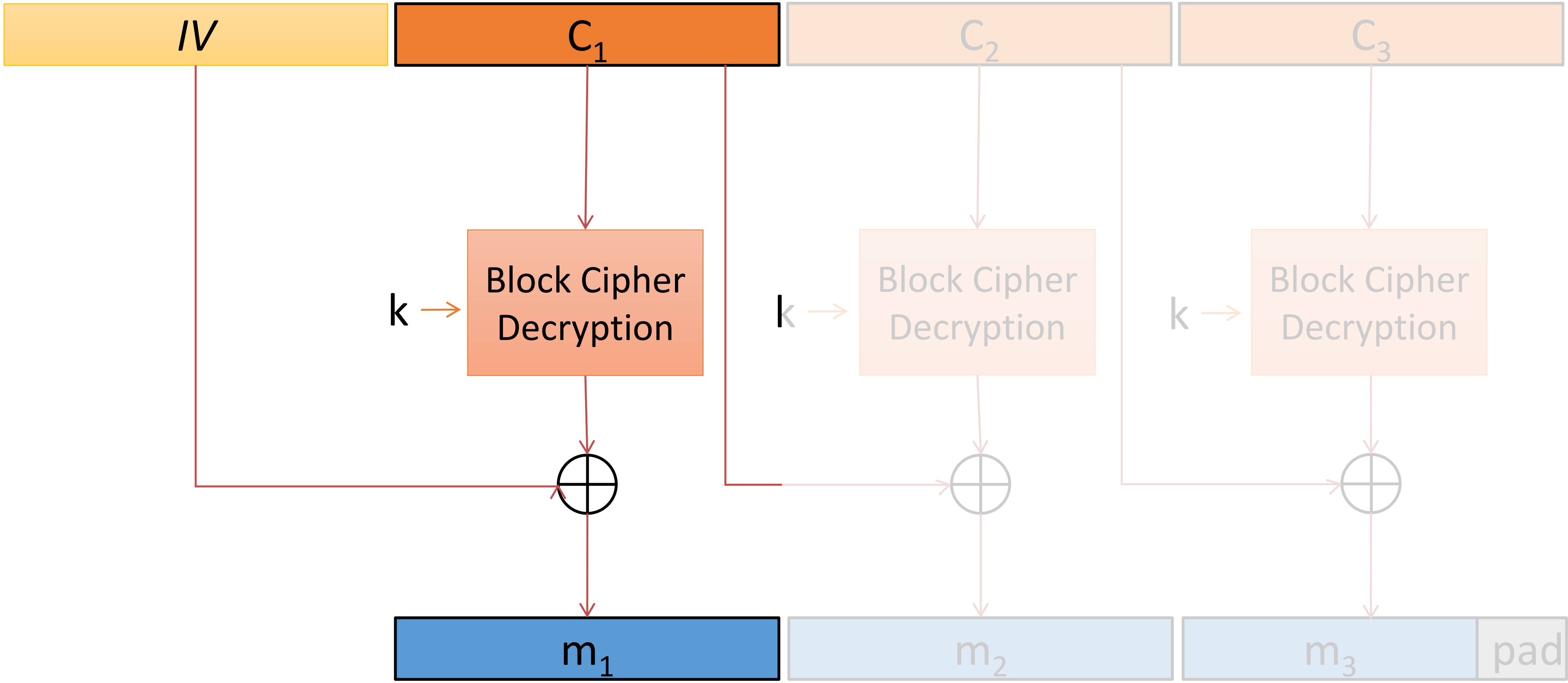
Overview – Padding Oracle Attacks

1. **Vaudenay's (padding oracle) attack**
2. **Padding oracle attack on TLS**
3. **Scalable scanning**
4. **Vulnerability clustering**
5. **Findings**

Cipher Block Chaining (CBC) Encryption

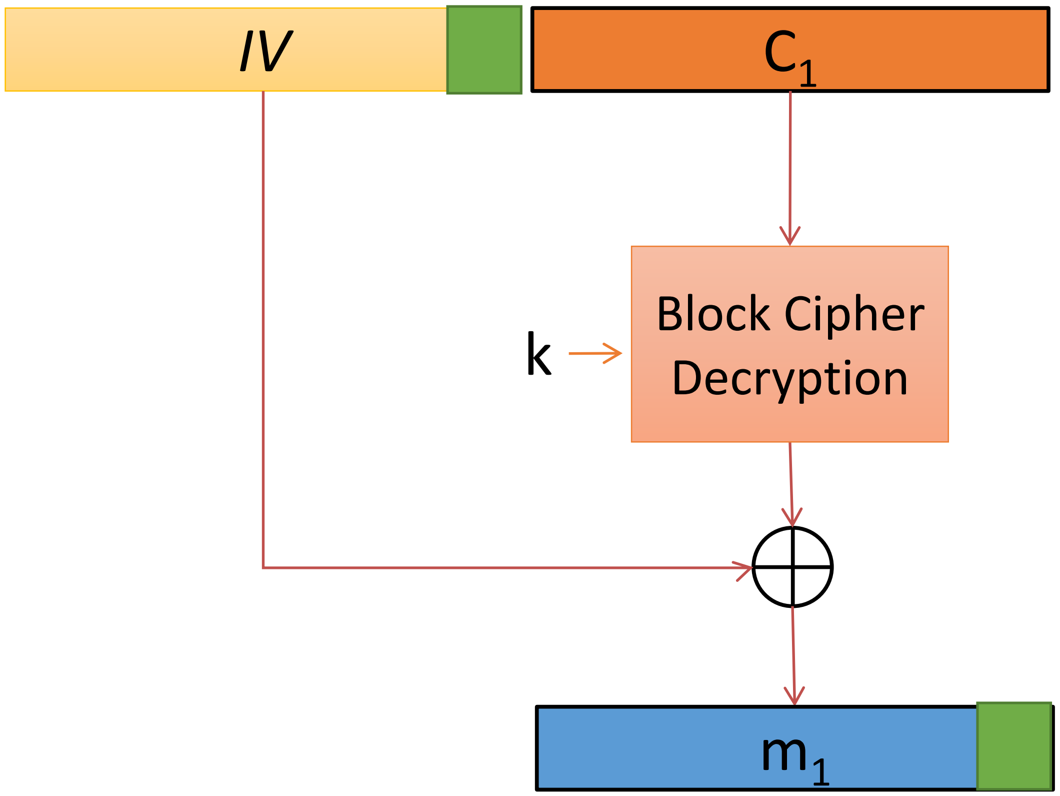


Cipher Block Chaining (CBC) Decryption



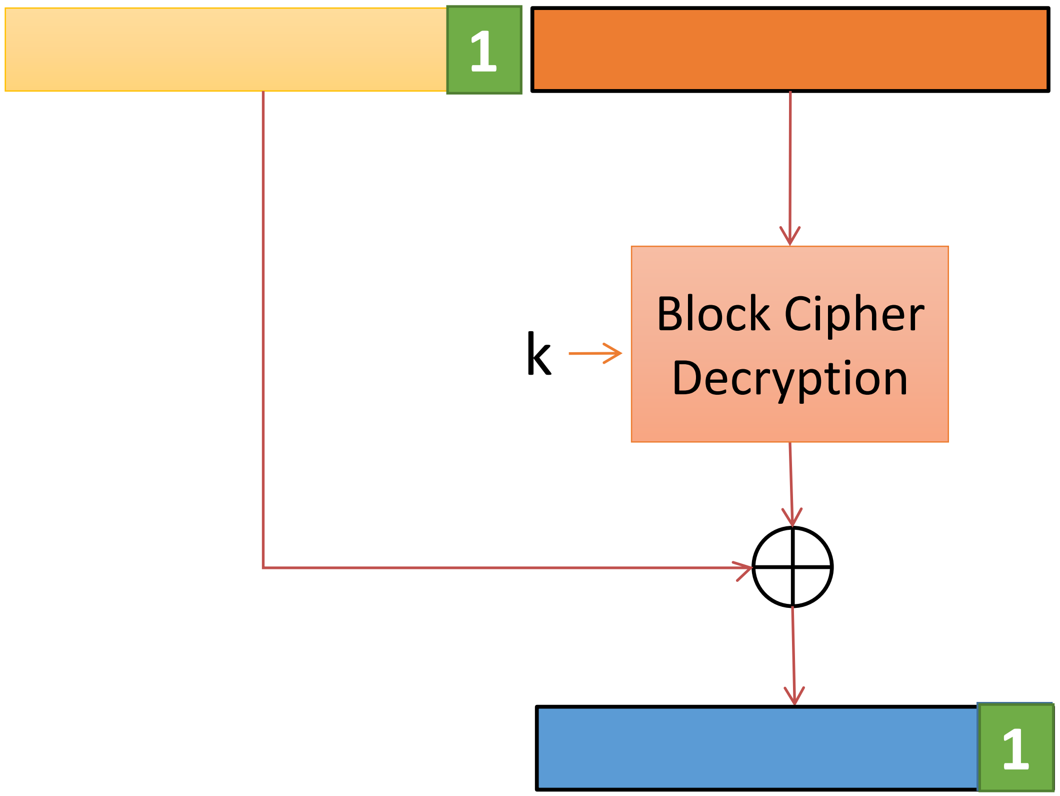
CBC Malleability

CBC decryption



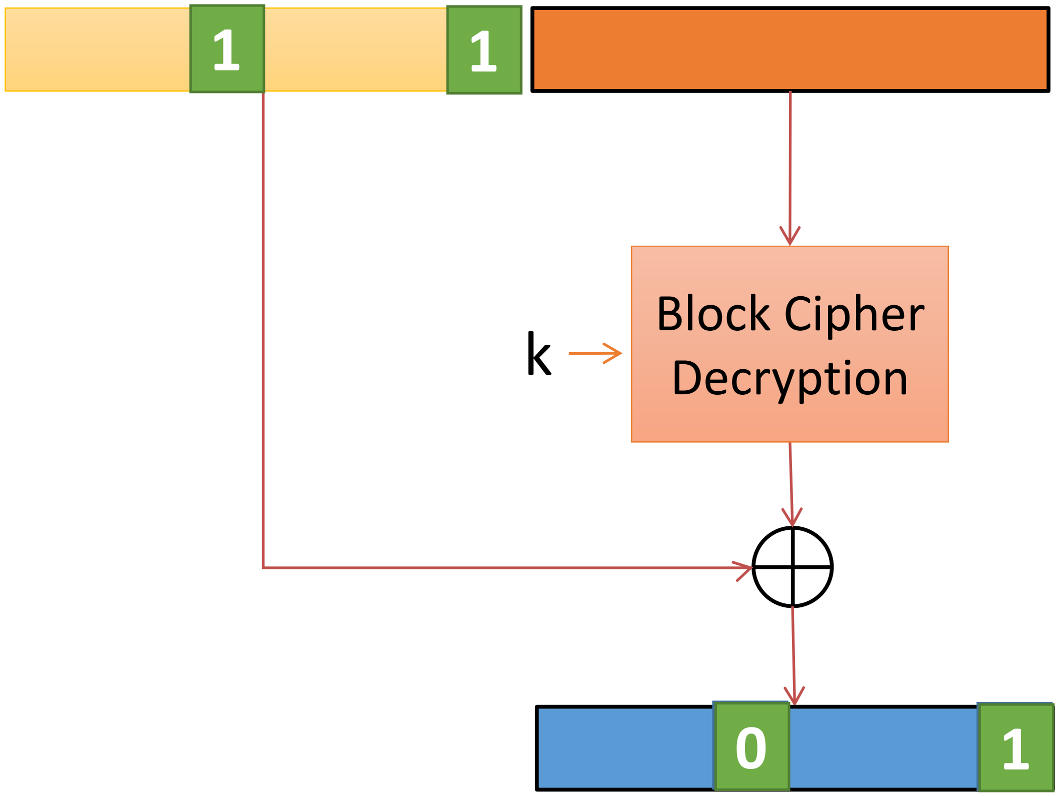
CBC Malleability: Example

CBC decryption



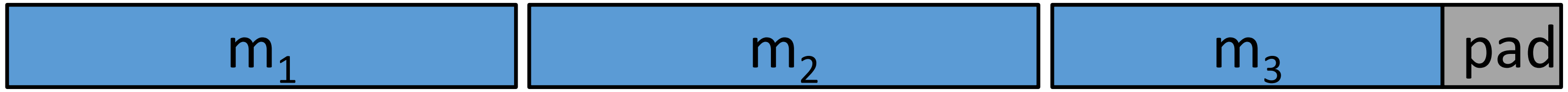
CBC Malleability: Example

CBC decryption



Why Are We Talking about Padding Oracles?

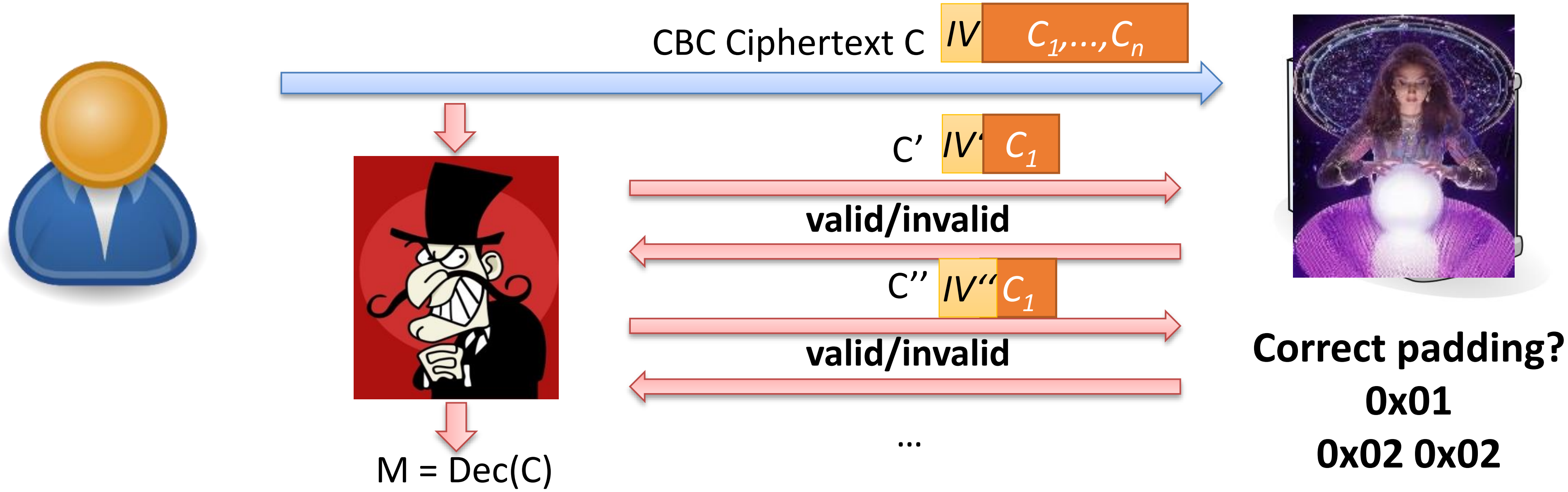
- CBC has a fixed padding structure



- Valid padding values:
  - 0x01
  - 0x02 0x02
  - 0x03 0x03 0x03
  - ….

Vaudenay’s Padding Oracle Attack Scenario

- 2002
- Adaptive chosen-ciphertext attack



CBC Ciphertext C *IV* $C_1,\dots,C_n$

C’ *IV‘* C_1

valid/invalid

C’’ *IV“* C_1

valid/invalid

…

M = Dec(C)

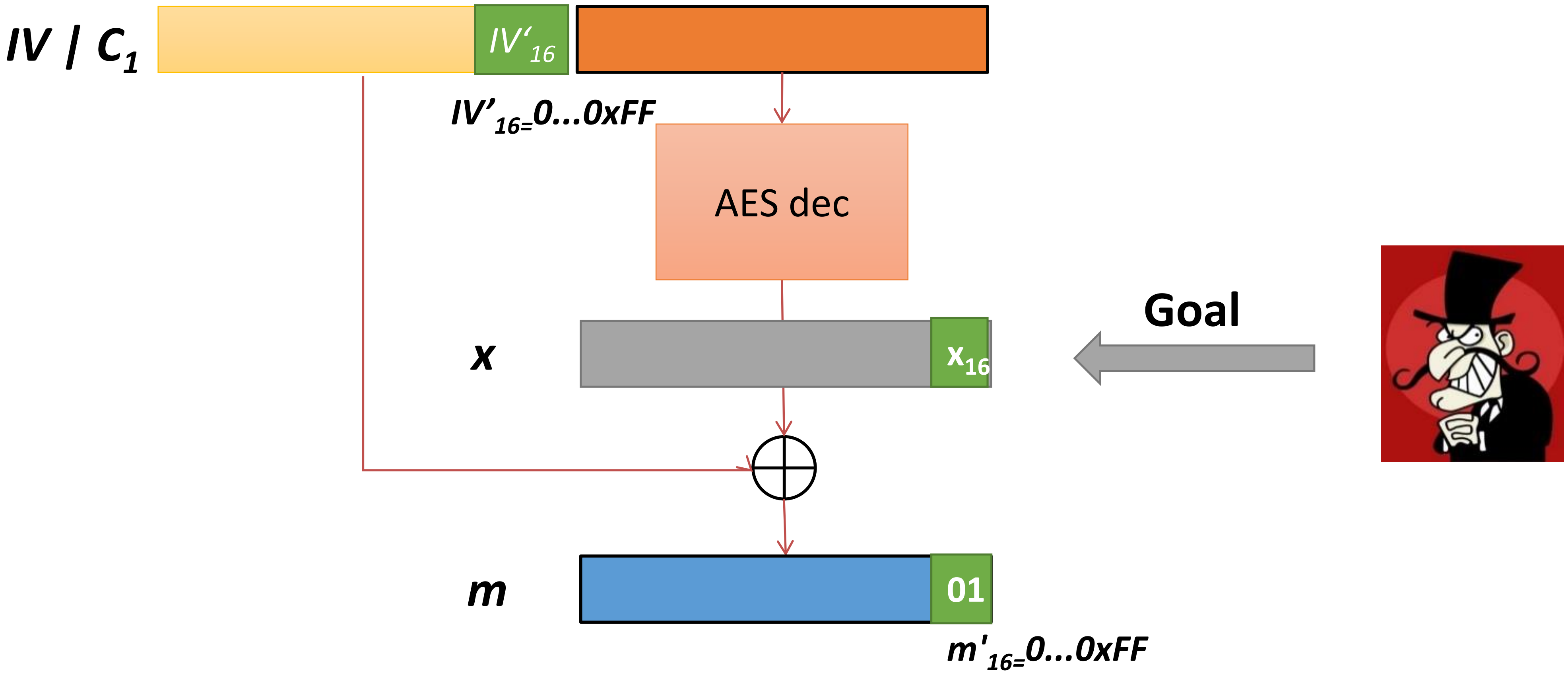
Correct padding?

0x01

0x02 0x02

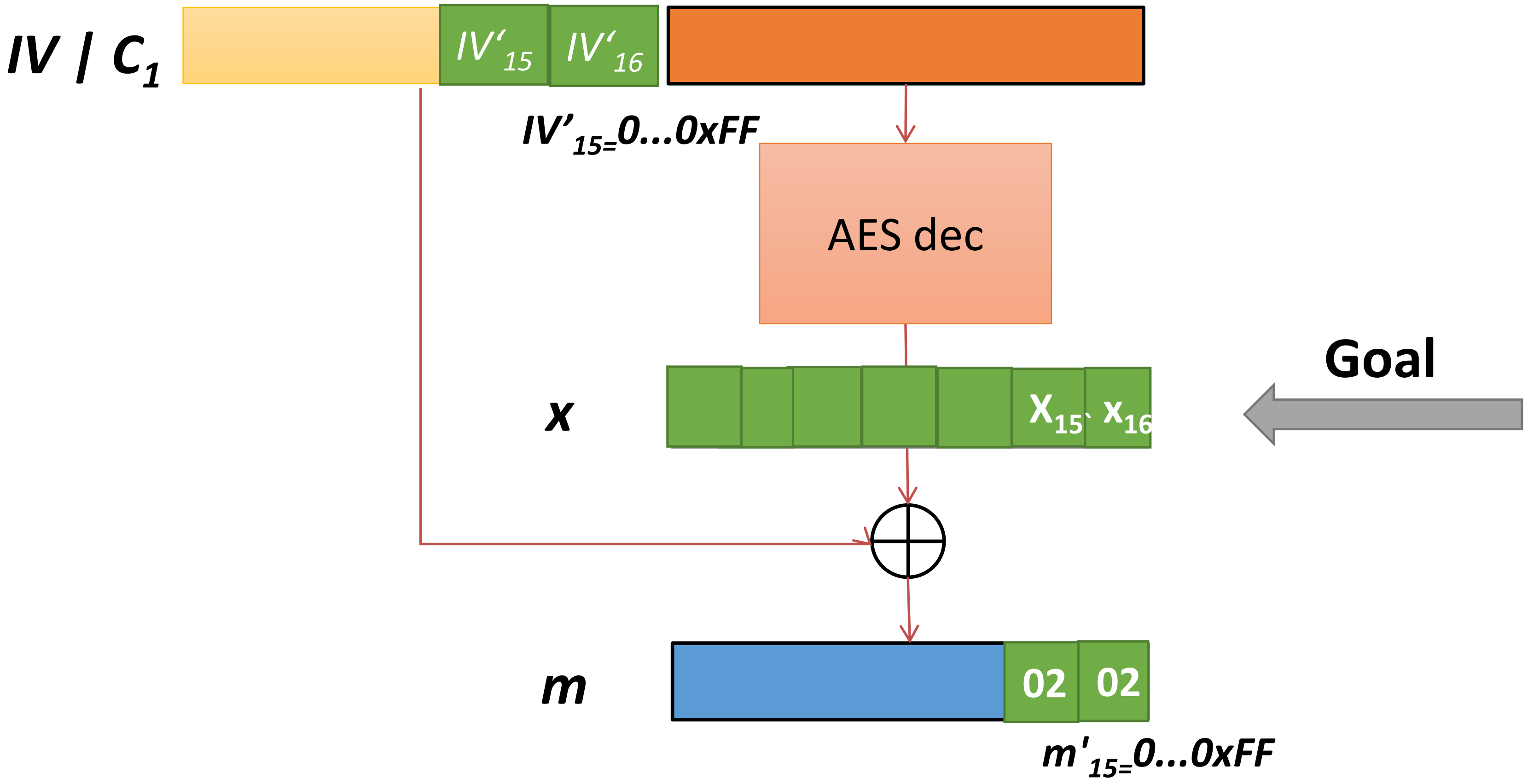
…

Vaudenay’s Padding Oracle Attack

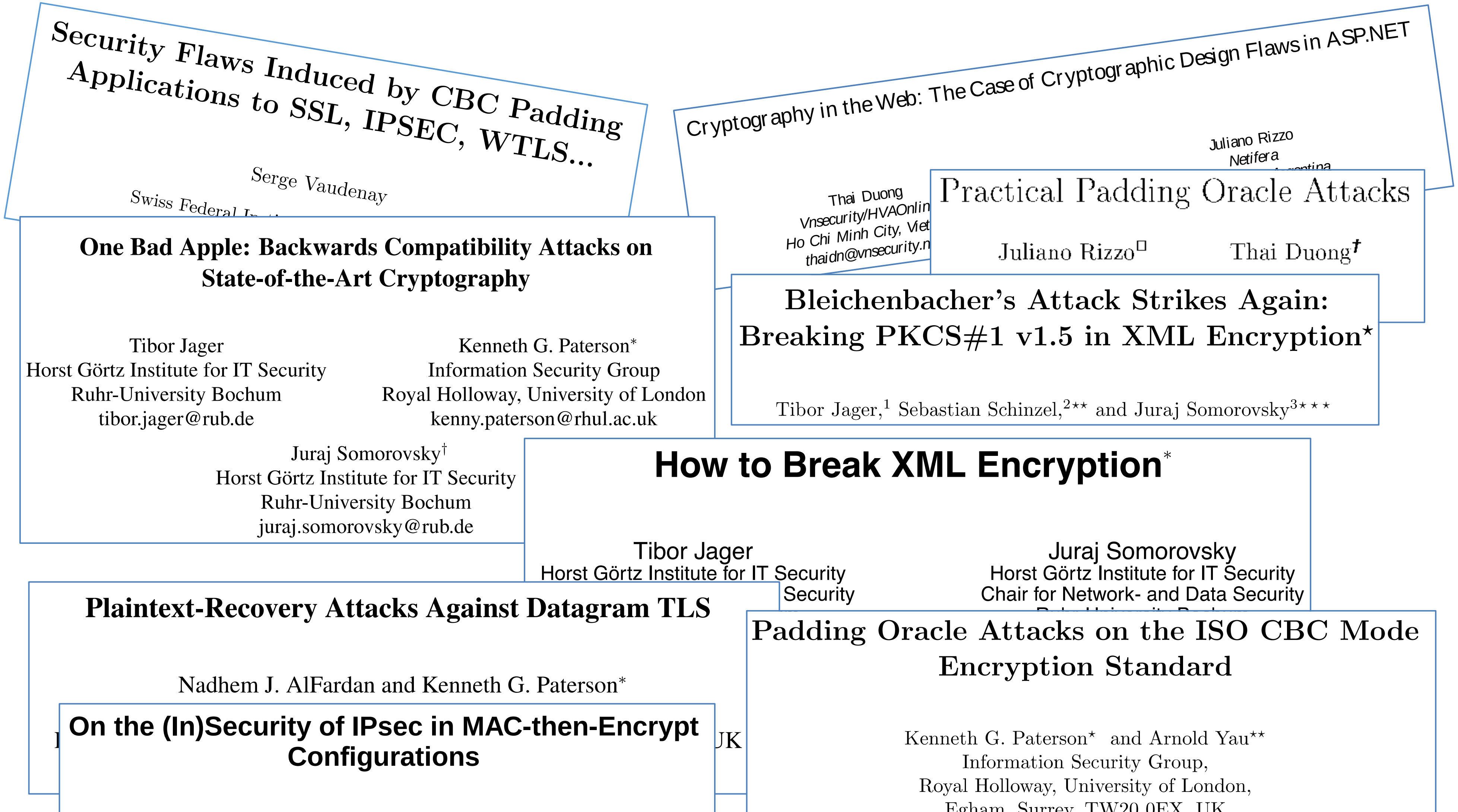


$$x_{16} = 0x01 \oplus IV'_{16}$$

Vaudenay’s Padding Oracle Attack



Padding Oracles Exploited in Many Scenarios



Security Flaws Induced by CBC Padding Applications to SSL, IPSEC, WTLS...

Serge Vaudenay

Swiss Federal In

Cryptography in the Web: The Case of Cryptographic Design Flaws in ASP.NET

Juliano Rizzo
Netifera

Thai Duong
Vnsecurity/HVAOnlin
Ho Chi Minh City, Viet
thaidn@vnsecurity.n

Practical Padding Oracle Attacks

Juliano Rizzo Thai Duong†

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

Tibor Jager
Horst Görtz Institute for IT Security
Ruhr-University Bochum
tibor.jager@rub.de

Kenneth G. Paterson*
Information Security Group
Royal Holloway, University of London
kenny.paterson@rhul.ac.uk

Juraj Somorovsky†
Horst Görtz Institute for IT Security
Ruhr-University Bochum
juraj.somorovsky@rub.de

Bleichenbacher's Attack Strikes Again: Breaking PKCS#1 v1.5 in XML Encryption⋆

Tibor Jager,[1] Sebastian Schinzel,[2]⋆⋆ and Juraj Somorovsky[3]⋆⋆⋆

How to Break XML Encryption*

Tibor Jager
Horst Görtz Institute for IT Security
Security

Juraj Somorovsky
Horst Görtz Institute for IT Security
Chair for Network- and Data Security

Plaintext-Recovery Attacks Against Datagram TLS

Nadhem J. AlFardan and Kenneth G. Paterson*

UK

On the (In)Security of IPsec in MAC-then-Encrypt Configurations

Padding Oracle Attacks on the ISO CBC Mode Encryption Standard

Kenneth G. Paterson⋆ and Arnold Yau⋆⋆
Information Security Group,
Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, UK

Overview – Padding Oracle Attacks

1. **Vaudenay's (padding oracle) attack**
2. **Padding oracle attack on TLS**
3. **Scalable scanning**
4. **Vulnerability clustering**
5. **Findings**

CBC in TLS

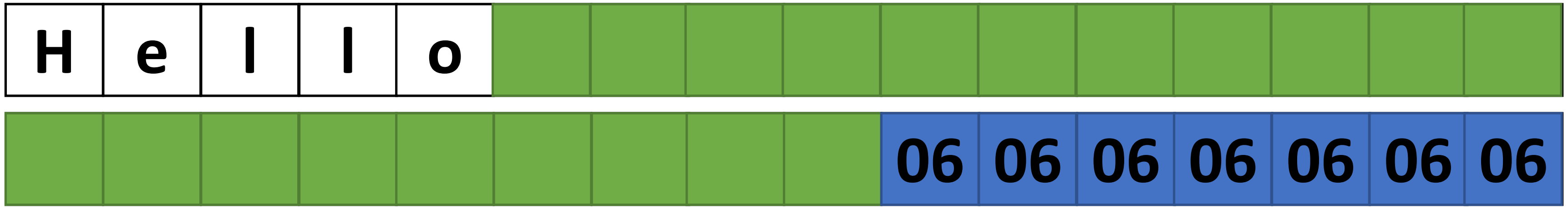
- Protects TLS records after a successful handshake
- Available since SSLv2
- Used in many cipher suites:
  - TLS_RSA_WITH_AES_256_CBC_SHA
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - …

AES-CBC in TLS

- MAC-Pad-Encrypt
- Example:
  - Two blocks
  - Message: Hello
  - MAC size: 20 bytes (SHA-1)
  - Padding size: 32 – 5 – 20 = 7

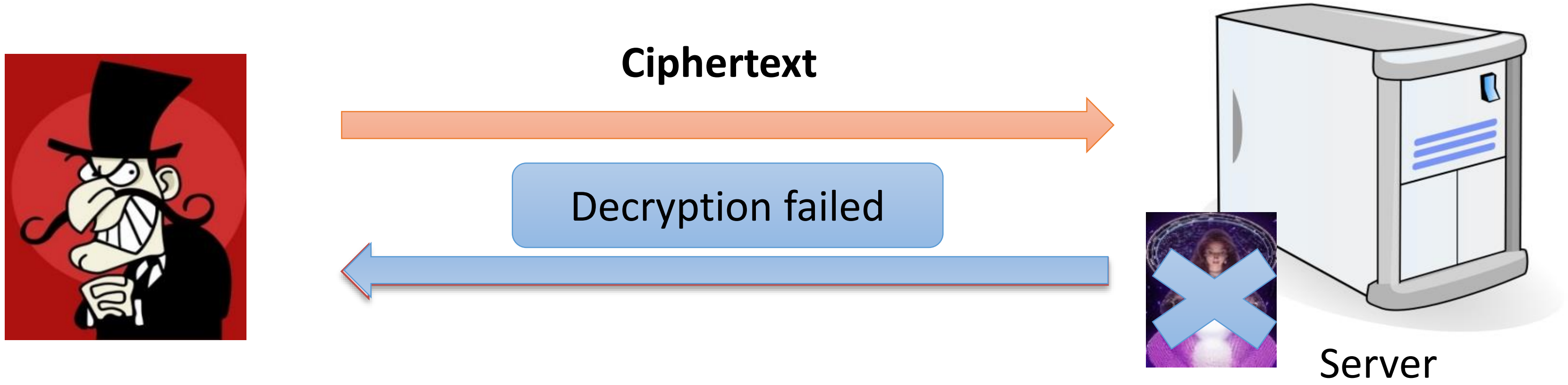
🟦 pad

🟩 mac



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

Preventing Padding Oracles



Ciphertext

Decryption failed

Server

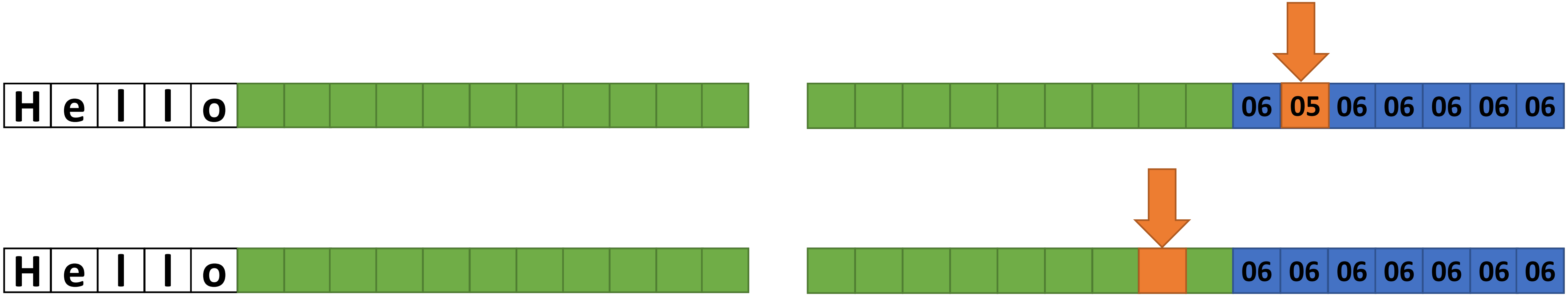
Challenge: not to reveal padding validity

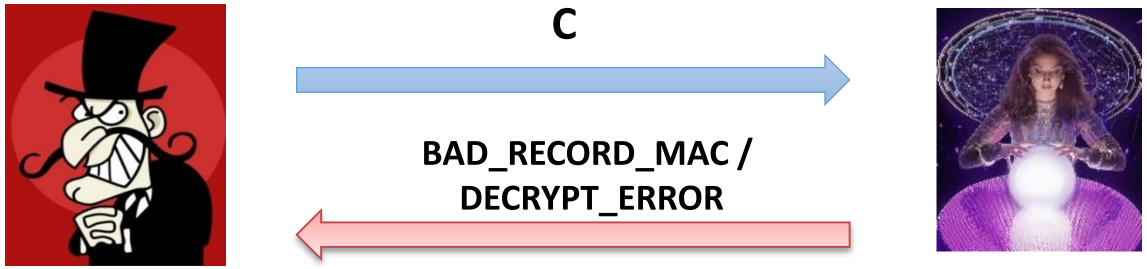
Same error message

Triggering a Typical Padding Oracle

🟦 pad

🟩 mac



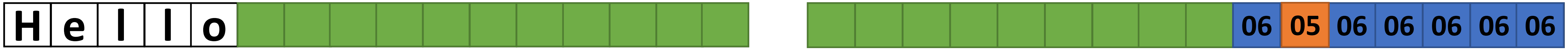


Botan (CVE-2015-7824)

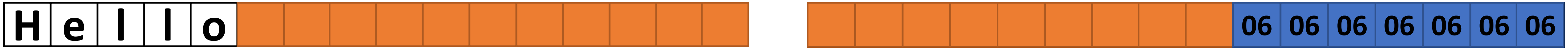
🟦 pad

🟩 mac

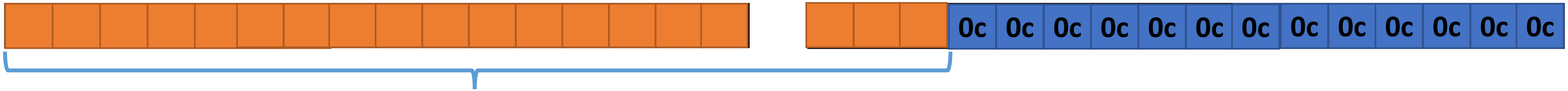
- Bad padding: **BAD_RECORD_MAC**



- Bad MAC: **BAD_RECORD_MAC**



- **Special case: DECODING_ERROR**



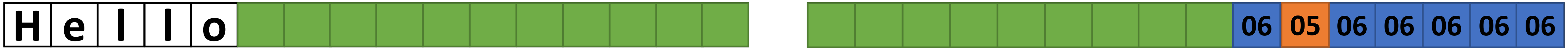
Not enough bytes for an HMAC (19)

OpenSSL (CVE-2016-2107)

🟦 pad

🟩 mac

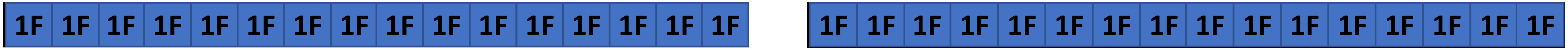
- Bad MAC / padding: **BAD_RECORD_MAC**



H e l l o

06 05 06 06 06 06 06

- Bad MAC: **RECORD_OVERFLOW**



1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F

1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F

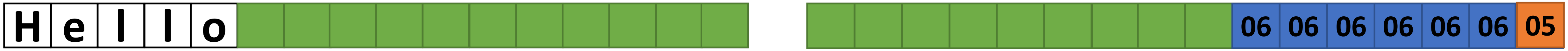
- Introduced after patching Lucky13
- Only available in AES-NI supported cipher suites:
  - TLS_RSA_WITH_AES_*_CBC_SHA
  - TLS_RSA_WITH_AES_*_CBC_SHA256

🟦 pad

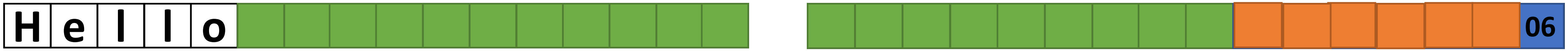
🟩 mac

POODLE

- Bad MAC / last padding byte: **BAD_RECORD_MAC**



- Arbitrary padding bytes: no error



Everything ok. SSLv3 uses a different padding scheme.

Overview – Padding Oracle Attacks

1. **Vaudenay’s (padding oracle) attack**
2. **Padding oracle attack on TLS**
3. **Scalable scanning**
4. **Vulnerability clustering**
5. **Findings**

Goals

- Generate test vectors
  - Use knowledge from previous works
- Scan Alexa 1M
- Identify vulnerabilities and report them

Test Vector Generation

1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F

1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F 1F

- Malformed TLS records (25)
  - Flipping bits in MAC
  - Flipping bits in padding
  - Overlong padding
- Using all …
  - TLS versions (see the SSLv3 POODLE vulnerability)
  - CBC cipher suites (see CVE-2016-2107)

25 malformed records in total

Test Vector Reduction

- Problem example:
  - 25 malformed records
  - Server supports TLS 1.0 and TLS 1.1 with 10 CBC cipher suites
  - Results in **500** TLS handshakes (without rescanning)

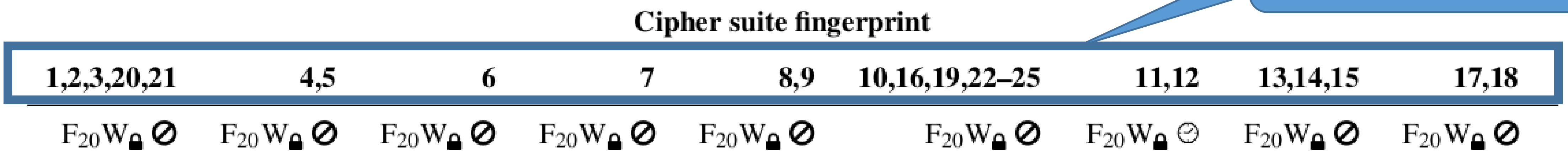
- Can we identify all vulnerabilities with less vectors?
  - Pre-scanning 50k random hosts on port 443
  - Reducing the set of malformed records from 25 to **4**
  - (still scanning with all TLS versions and cipher suites)

Alexa Top 1 Million Scan

- Based on our TLS-Attacker:
  - https://github.com/RUB-NDS/TLS-Attacker
- Supported by redis and mongoDB
- 72 hours, 627,493 hosts supporting CBC cipher suites
- 18,257 vulnerable
  - Observed different TLS alerts, TCP resets, timeouts, …

Malformed Records Triggering Different Fingerprints

Different malformed records



Cipher suite fingerprint

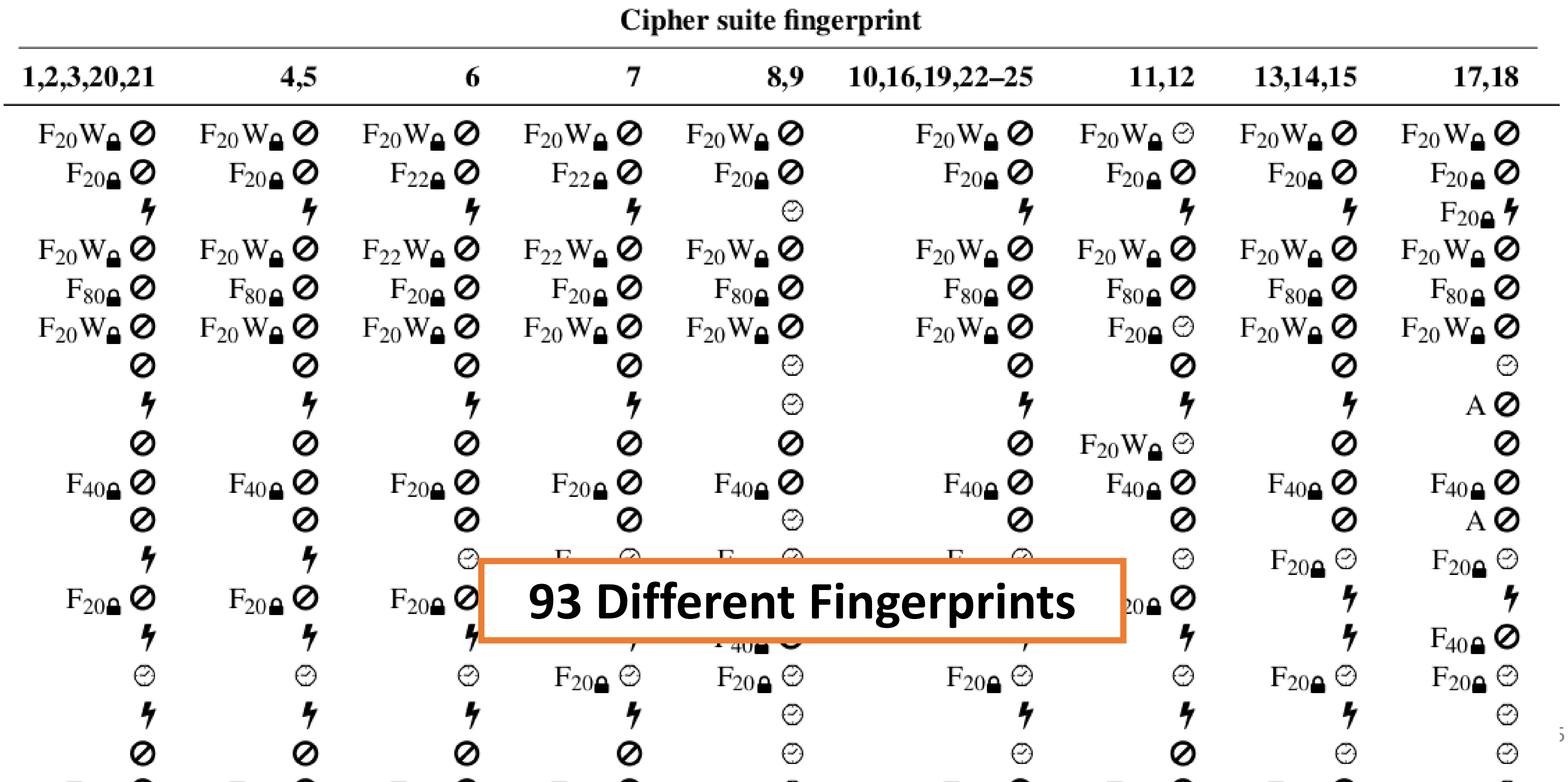
1,2,3,20,21	4,5	6	7	8,9	10,16,19,22–25	11,12	13,14,15	17,18
$F_{20}W$🔒⊘	$F_{20}W$🔒⊘	$F_{20}W$🔒⊘	$F_{20}W$🔒⊘	$F_{20}W$🔒⊘	$F_{20}W$🔒⊘	$F_{20}W$🔒⏱	$F_{20}W$🔒⊘	$F_{20}W$🔒⊘

Response:
- Fatal BAD_RECORD_MAC
- Warning
- Connection close

Response:
- Fatal BAD_RECORD_MAC
- Warning
- Timeout

How many different fingerprints do we have?

Malformed Records Triggering Different Fingerprints

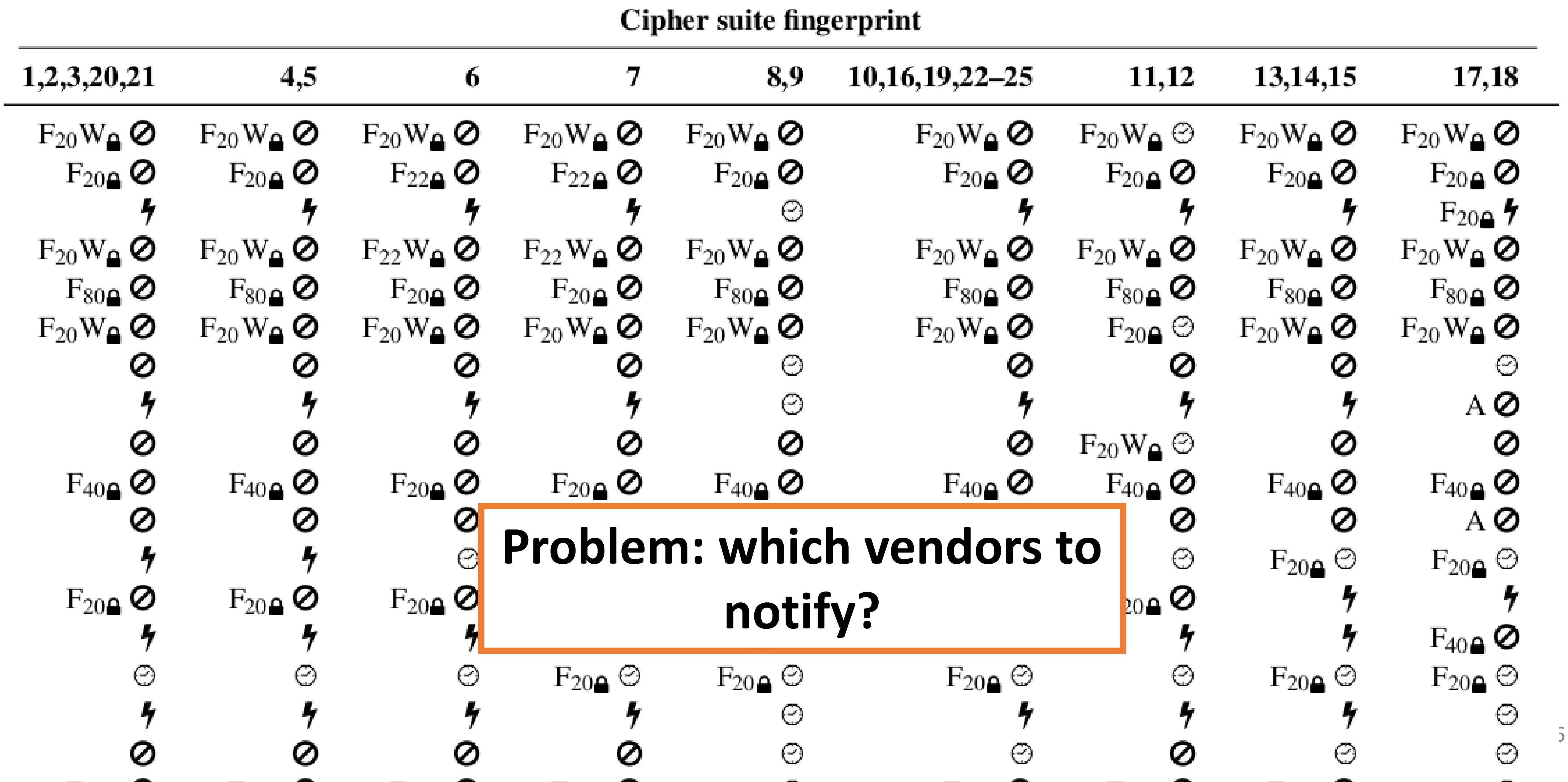


Cipher suite fingerprint

1,2,3,20,21	4,5	6	7	8,9	10,16,19,22–25	11,12	13,14,15	17,18
F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⏲	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘
F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{22}$🔒 ⊘	F$_{22}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘
ϟ	ϟ	ϟ	ϟ	⏲	ϟ	ϟ	ϟ	F$_{20}$🔒 ϟ
F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{22}$W🔒 ⊘	F$_{22}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘
F$_{80}$🔒 ⊘	F$_{80}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{80}$🔒 ⊘	F$_{80}$🔒 ⊘	F$_{80}$🔒 ⊘	F$_{80}$🔒 ⊘	F$_{80}$🔒 ⊘
F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$🔒 ⏲	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘
⊘	⊘	⊘	⊘	⏲	⊘	⊘	⊘	⏲
ϟ	ϟ	ϟ	ϟ	⏲	ϟ	ϟ	ϟ	A ⊘
⊘	⊘	⊘	⊘	⊘	⊘	F$_{20}$W🔒 ⏲	⊘	⊘
F$_{40}$🔒 ⊘	F$_{40}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{40}$🔒 ⊘	F$_{40}$🔒 ⊘	F$_{40}$🔒 ⊘	F$_{40}$🔒 ⊘	F$_{40}$🔒 ⊘
⊘	⊘	⊘	⊘	⏲	⊘	⊘	⊘	A ⊘
ϟ	ϟ	⏲	[illegible] ⏲	[illegible] ⏲	[illegible] ⏲	⏲	F$_{20}$🔒 ⏲	F$_{20}$🔒 ⏲
F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	[illegible]	[illegible]	[illegible]	[illegible]🔒 ⊘	ϟ	ϟ
ϟ	ϟ	ϟ	[illegible]	[illegible]	[illegible]	ϟ	ϟ	F$_{40}$🔒 ⊘
⏲	⏲	⏲	F$_{20}$🔒 ⏲	F$_{20}$🔒 ⏲	F$_{20}$🔒 ⏲	⏲	F$_{20}$🔒 ⏲	F$_{20}$🔒 ⏲
ϟ	ϟ	ϟ	ϟ	⏲	ϟ	ϟ	ϟ	⏲
⊘	⊘	⊘	⊘	⏲	⏲	⊘	⏲	⏲

93 Different Fingerprints

Malformed Records Triggering Different Fingerprints



Cipher suite fingerprint

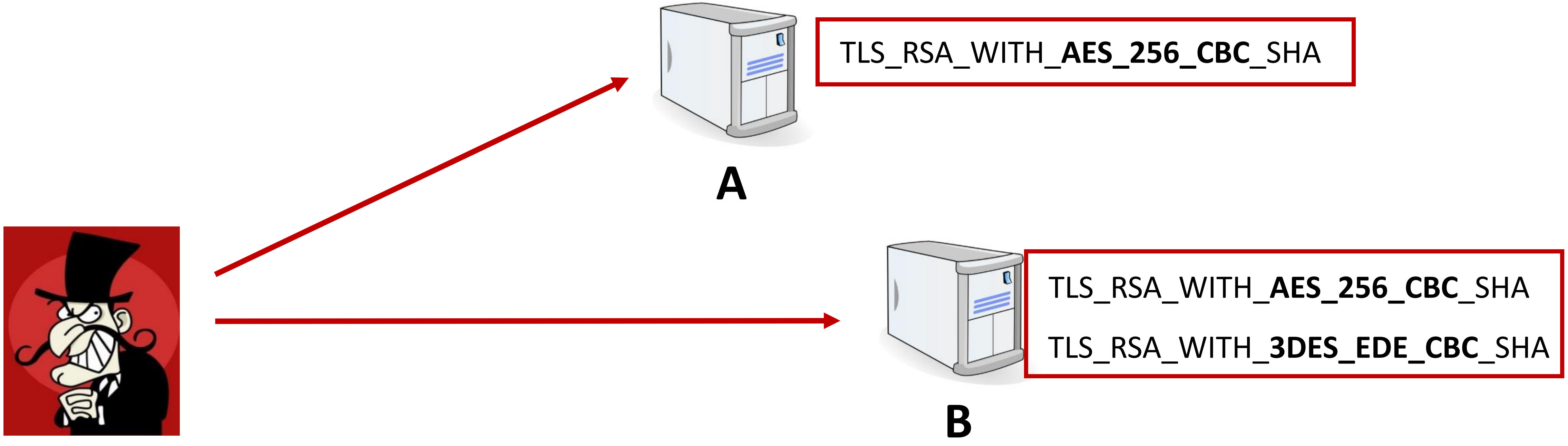
1,2,3,20,21	4,5	6	7	8,9	10,16,19,22–25	11,12	13,14,15	17,18
F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ◷	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘
F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{22}$🔒 ⊘	F$_{22}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘
ϟ	ϟ	ϟ	ϟ	◷	ϟ	ϟ	ϟ	F$_{20}$🔒 ϟ
F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{22}$W🔒 ⊘	F$_{22}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘
F$_{80}$🔒 ⊘	F$_{80}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{80}$🔒 ⊘	F$_{80}$🔒 ⊘	F$_{80}$🔒 ⊘	F$_{80}$🔒 ⊘	F$_{80}$🔒 ⊘
F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘	F$_{20}$🔒 ◷	F$_{20}$W🔒 ⊘	F$_{20}$W🔒 ⊘
⊘	⊘	⊘	⊘	◷	⊘	⊘	⊘	◷
ϟ	ϟ	ϟ	ϟ	◷	ϟ	ϟ	ϟ	A ⊘
⊘	⊘	⊘	⊘	⊘	⊘	F$_{20}$W🔒 ◷	⊘	⊘
F$_{40}$🔒 ⊘	F$_{40}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{40}$🔒 ⊘	F$_{40}$🔒 ⊘	F$_{40}$🔒 ⊘	F$_{40}$🔒 ⊘	F$_{40}$🔒 ⊘
⊘	⊘	⊘				⊘	⊘	A ⊘
ϟ	ϟ	◷				◷	F$_{20}$🔒 ◷	F$_{20}$🔒 ◷
F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘	F$_{20}$🔒 ⊘				20🔒 ⊘	ϟ	ϟ
ϟ	ϟ	ϟ				ϟ	ϟ	F$_{40}$🔒 ⊘
◷	◷	◷	F$_{20}$🔒 ◷	F$_{20}$🔒 ◷	F$_{20}$🔒 ◷	◷	F$_{20}$🔒 ◷	F$_{20}$🔒 ◷
ϟ	ϟ	ϟ	ϟ	◷	ϟ	ϟ	ϟ	◷
⊘	⊘	⊘	⊘	◷	◷	⊘	◷	◷

Problem: which vendors to notify?

Overview – Padding Oracle Attacks

1. **Vaudenay's (padding oracle) attack**
2. **Padding oracle attack on TLS**
3. **Scalable scanning**
4. **Vulnerability clustering**
5. **Findings**

Vulnerability Clustering



TLS_RSA_WITH_**AES_256_CBC**_SHA

A

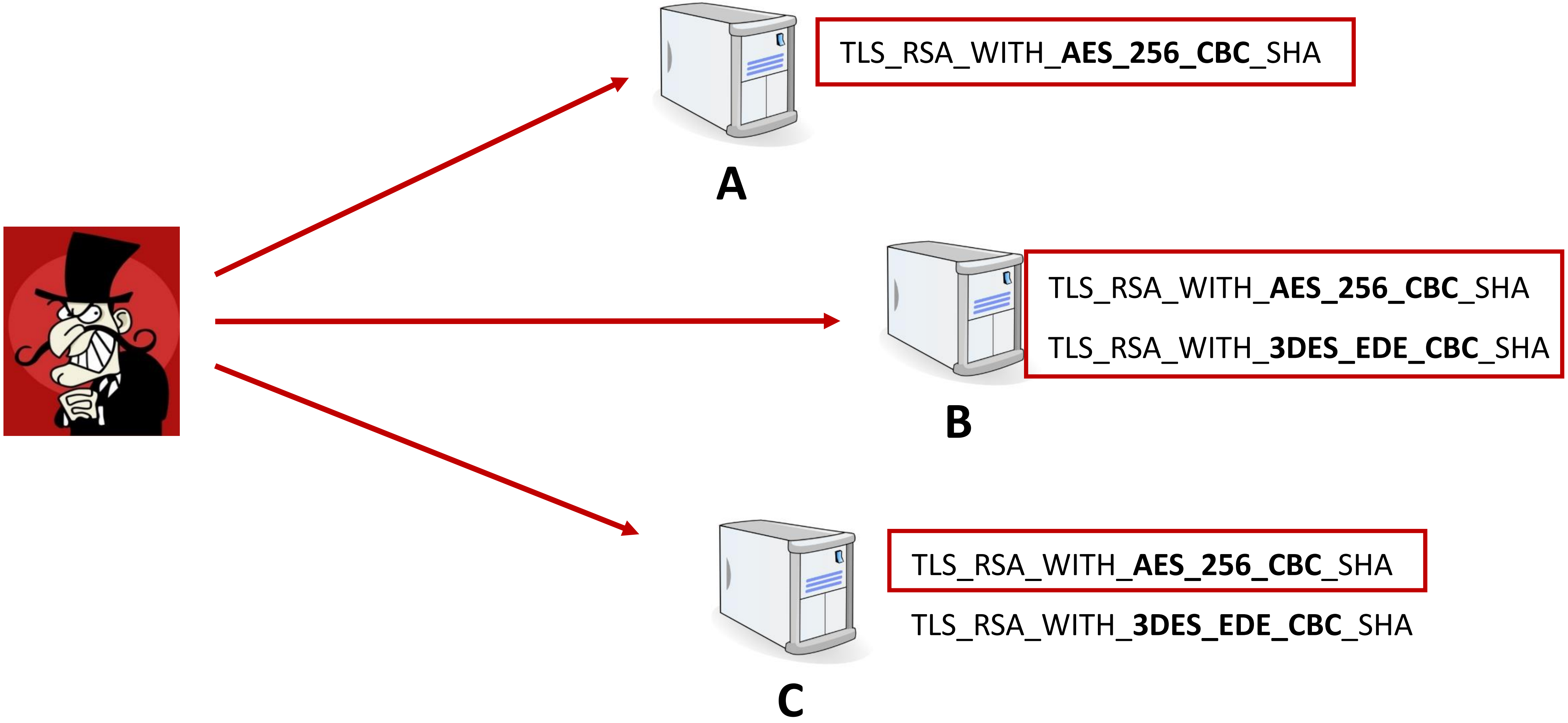
TLS_RSA_WITH_**AES_256_CBC**_SHA

TLS_RSA_WITH_**3DES_EDE_CBC**_SHA

B

Are A and B implementations equal?

Vulnerability Clustering



A

TLS_RSA_WITH_**AES_256_CBC**_SHA

B

TLS_RSA_WITH_**AES_256_CBC**_SHA

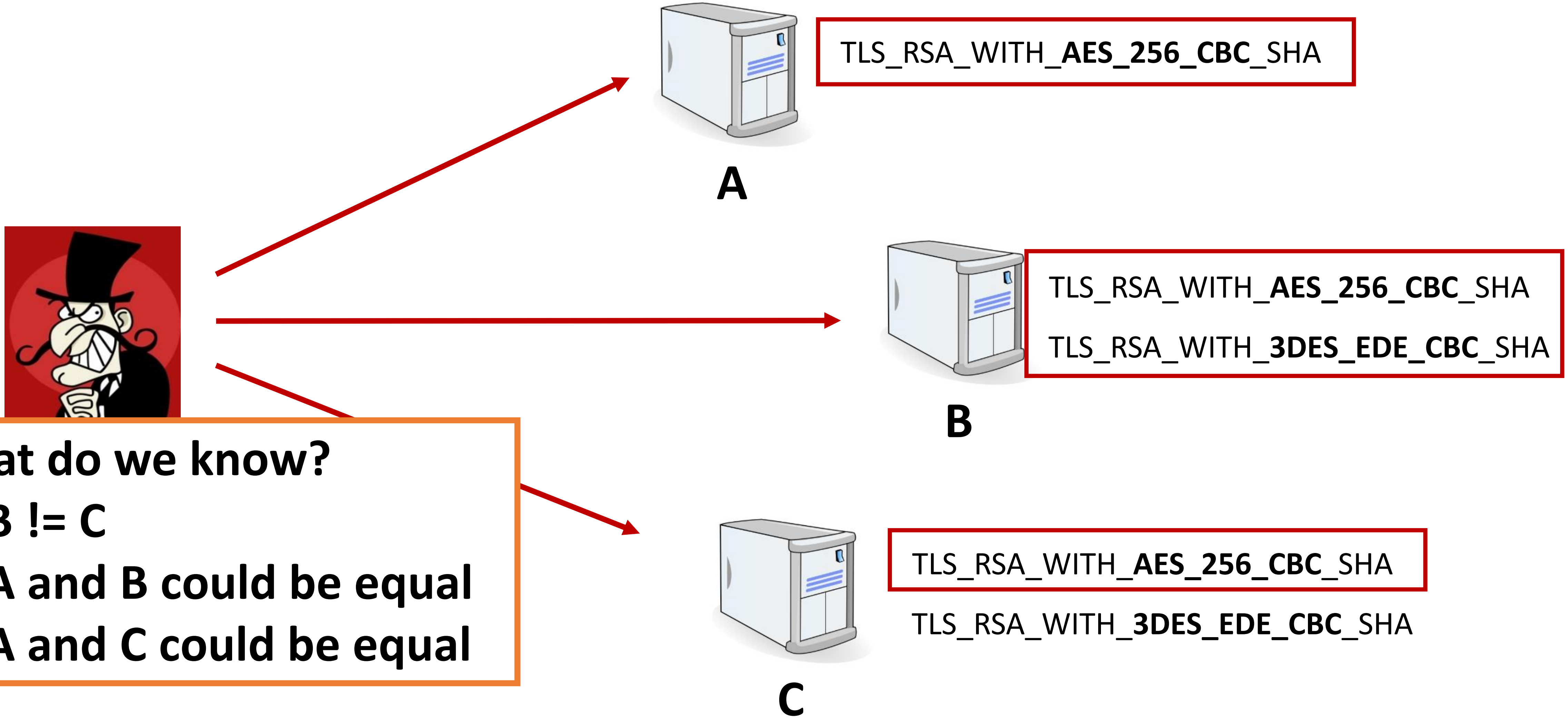
TLS_RSA_WITH_**3DES_EDE_CBC**_SHA

C

TLS_RSA_WITH_**AES_256_CBC**_SHA

TLS_RSA_WITH_**3DES_EDE_CBC**_SHA

Vulnerability Clustering



A

TLS_RSA_WITH_**AES_256_CBC**_SHA

B

TLS_RSA_WITH_**AES_256_CBC**_SHA

TLS_RSA_WITH_**3DES_EDE_CBC**_SHA

C

TLS_RSA_WITH_**AES_256_CBC**_SHA

TLS_RSA_WITH_**3DES_EDE_CBC**_SHA

What do we know?

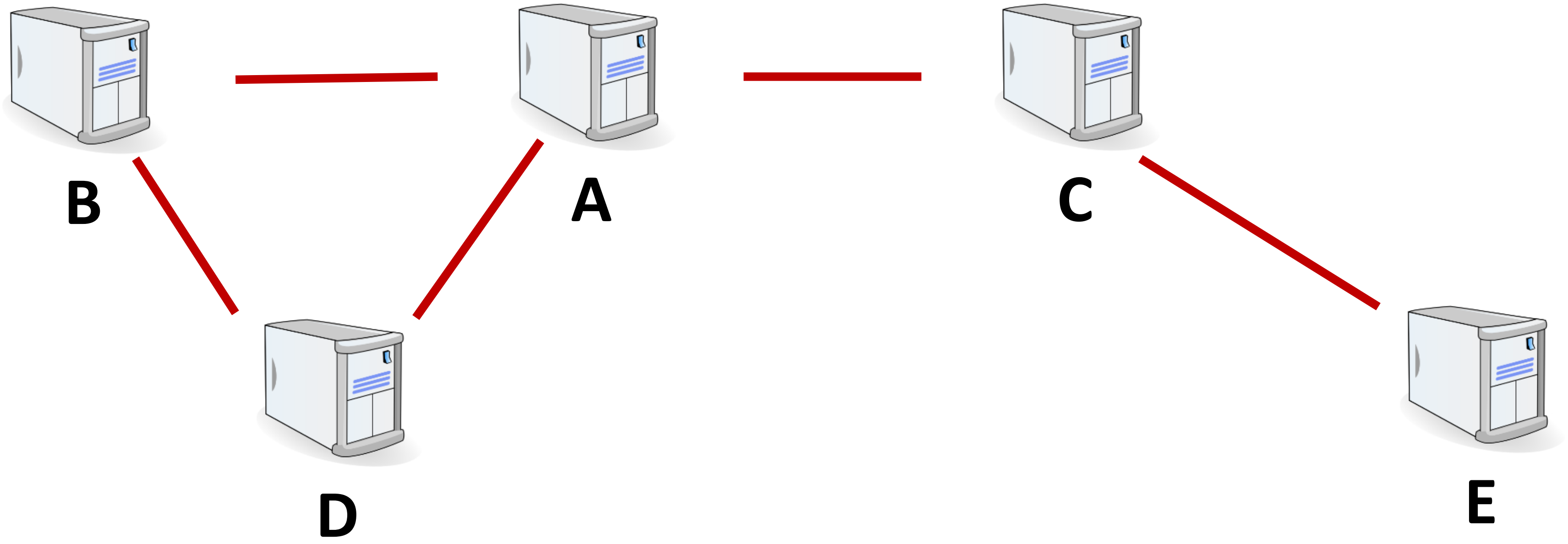
- **B != C**
- **A and B could be equal**
- **A and C could be equal**

Vulnerability Clustering: Usage of Force Atlas2

- Two-dimensional graph
- Edges between equally behaving servers
- Contains as few crossing edges as possible

What do we know?

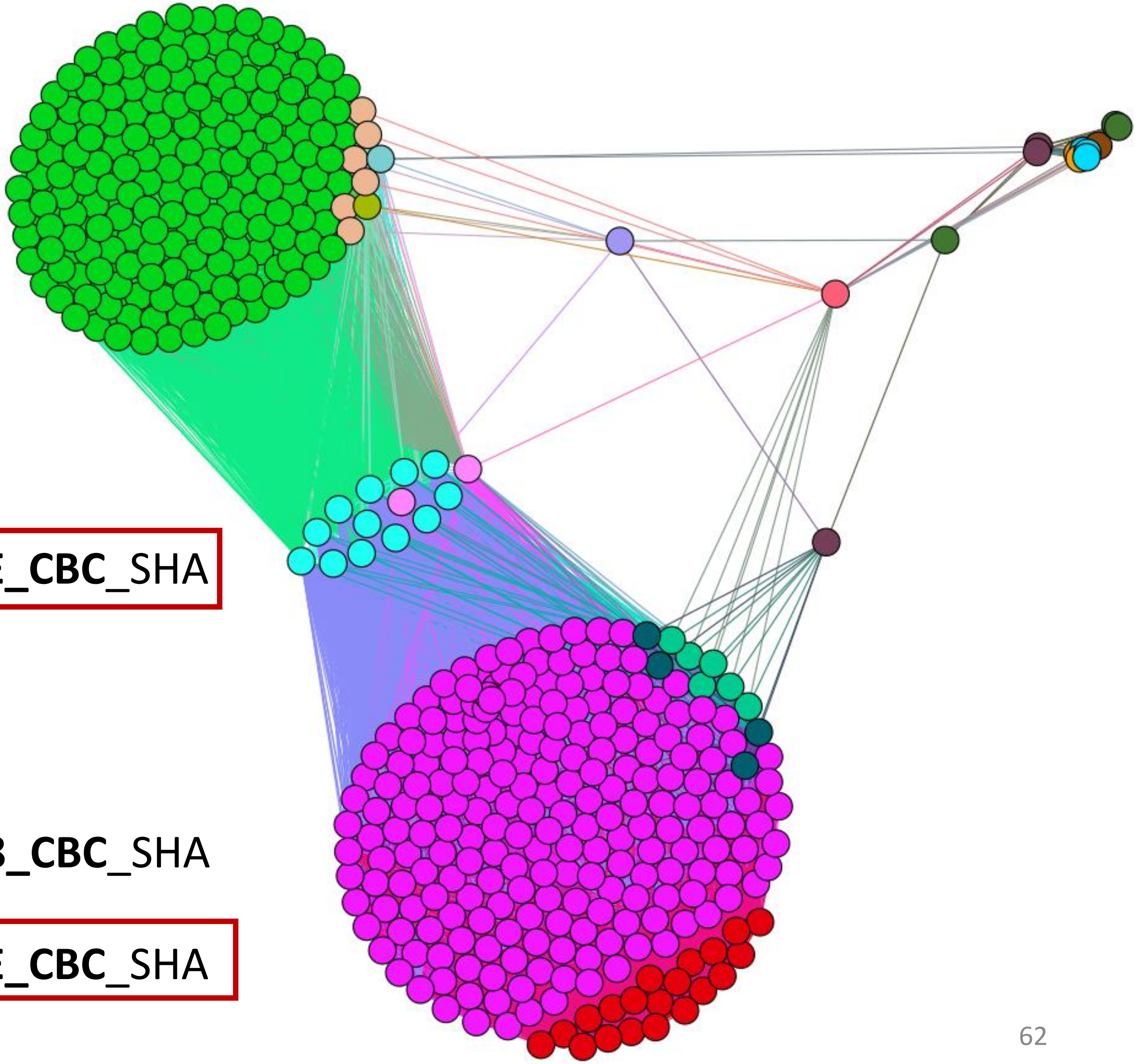
- **B != C**
- **A and B could be equal**
- **A and C could be equal**



Vulnerability Clustering: Example

TLS_RSA_WITH_**AES_128_CBC**_SHA

TLS_DHE_RSA_WITH_**3DES_EDE_CBC**_SHA



TLS_DHE_RSA_WITH_**3DES_EDE_CBC**_SHA

TLS_RSA_WITH_**AES_128_CBC**_SHA

TLS_DHE_RSA_WITH_**3DES_EDE_CBC**_SHA

Overview – Padding Oracle Attacks

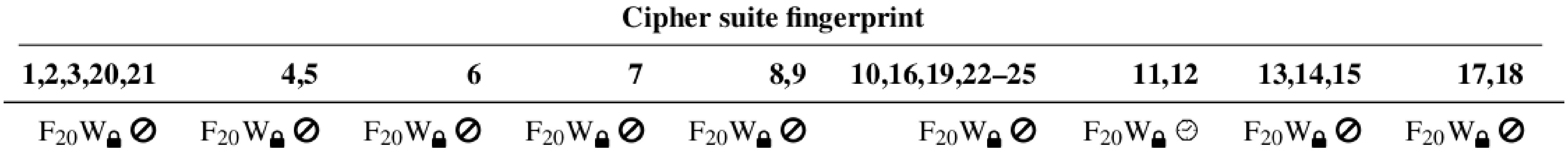
1. **Vaudenay’s (padding oracle) attack**
2. **Padding oracle attack on TLS**
3. **Scalable scanning**
4. **Vulnerability clustering**
5. **Findings**

Results

- https://github.com/RUB-NDS/TLS-Padding-Oracles
- **OpenSSL**. CVE-2019-1559.
- Citrix. CVE-2019-6485.
- F5 TMM TLS virtual server. CVE-2019-6593.
- SonicWall SonicOs. CVE-2019-7477

OpenSSL (CVE-2019-1559)

- Identified with the help of the Amazon security team
- Only in stitched cipher suites
  - Highly optimized, no AES-NI



Cipher suite fingerprint

1,2,3,20,21	4,5	6	7	8,9	10,16,19,22–25	11,12	13,14,15	17,18
F_{20} W🔒 ⊘	F_{20} W🔒 ⊘	F_{20} W🔒 ⊘	F_{20} W🔒 ⊘	F_{20} W🔒 ⊘	F_{20} W🔒 ⊘	F_{20} W🔒 ⏲	F_{20} W🔒 ⊘	F_{20} W🔒 ⊘

Response:
- **Fatal BAD_RECORD_MAC**
- **Warning**
- **Connection close**

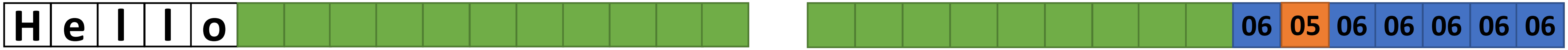
Response:
- **Fatal BAD_RECORD_MAC**
- **Warning**
- **Timeout**

🟦 pad

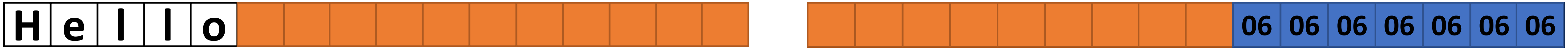
🟩 mac

OpenSSL (CVE-2019-1559)

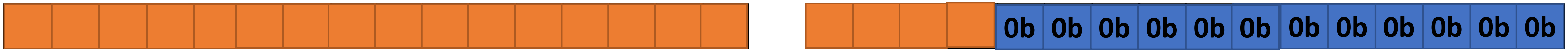
- Bad padding: **BAD_RECORD_MAC, Connection Close**



- Bad MAC: **BAD_RECORD_MAC, Connection Close**



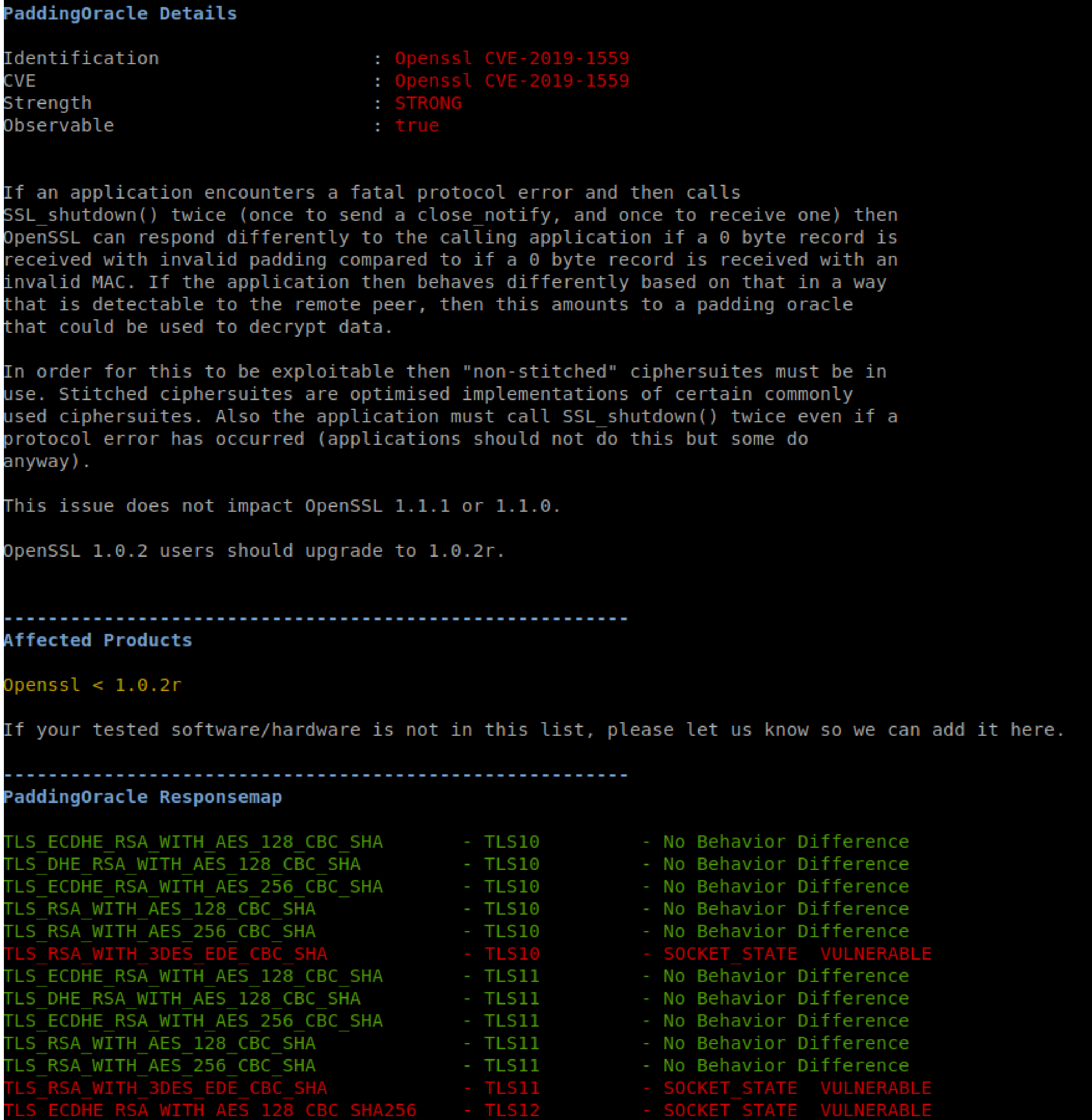
- **0-length record: BAD_RECORD_MAC, Timeout**



Disclaimer

- CBC padding oracle attacks are much harder to exploit …
    - BEAST scenario needed
    - Active MitM attacker
    - …

Demo: TLS-Scanner



```
PaddingOracle Details

Identification                   : Openssl CVE-2019-1559
CVE                              : Openssl CVE-2019-1559
Strength                         : STRONG
Observable                       : true


If an application encounters a fatal protocol error and then calls
SSL_shutdown() twice (once to send a close_notify, and once to receive one) then
OpenSSL can respond differently to the calling application if a 0 byte record is
received with invalid padding compared to if a 0 byte record is received with an
invalid MAC. If the application then behaves differently based on that in a way
that is detectable to the remote peer, then this amounts to a padding oracle
that could be used to decrypt data.

In order for this to be exploitable then "non-stitched" ciphersuites must be in
use. Stitched ciphersuites are optimised implementations of certain commonly
used ciphersuites. Also the application must call SSL_shutdown() twice even if a
protocol error has occurred (applications should not do this but some do
anyway).

This issue does not impact OpenSSL 1.1.1 or 1.1.0.

OpenSSL 1.0.2 users should upgrade to 1.0.2r.


--------------------------------------------------------
Affected Products

Openssl < 1.0.2r

If your tested software/hardware is not in this list, please let us know so we can add it here.

--------------------------------------------------------
PaddingOracle Responsemap

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA       - TLS10         - No Behavior Difference
TLS_DHE_RSA_WITH_AES_128_CBC_SHA         - TLS10         - No Behavior Difference
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA       - TLS10         - No Behavior Difference
TLS_RSA_WITH_AES_128_CBC_SHA             - TLS10         - No Behavior Difference
TLS_RSA_WITH_AES_256_CBC_SHA             - TLS10         - No Behavior Difference
TLS_RSA_WITH_3DES_EDE_CBC_SHA            - TLS10         - SOCKET_STATE  VULNERABLE
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA       - TLS11         - No Behavior Difference
TLS_DHE_RSA_WITH_AES_128_CBC_SHA         - TLS11         - No Behavior Difference
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA       - TLS11         - No Behavior Difference
TLS_RSA_WITH_AES_128_CBC_SHA             - TLS11         - No Behavior Difference
TLS_RSA_WITH_AES_256_CBC_SHA             - TLS11         - No Behavior Difference
TLS_RSA_WITH_3DES_EDE_CBC_SHA            - TLS11         - SOCKET_STATE  VULNERABLE
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256    - TLS12         - SOCKET_STATE  VULNERABLE
```

Conclusions

- **20** years old attacks still work
- New side-channels (timeouts, TCP resets, …)
- New attacks can be found by IPv4 scanning

- Disable RSA and CBC cipher suites (not used in TLS 1.3)
- Stop using RSA PKCS#1 v1.5
  - Use elliptic curves (or RSA-OAEP if RSA needed)
- Stop using CBC
  - Use authenticated modes of operation like AES-GCM