

Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks

Juraj Somorovsky
Ruhr University Bochum

3curity GmbH
juraj.somorovsky@3curity.de



About me

hg  NDS



- Security Researcher at:
 - Chair for Network and Data Security, Ruhr University Bochum
 - 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...
- Co-founder of 3curity GmbH:
 - Penetration tests, security analyses, security workshops...
 - Web, Single Sign-On, SSL, applied crypto
 - www.3curity.de

The logo for 3curity, featuring a large red '3' followed by the word 'curity' in black, with a horizontal line underneath.

Publications

- XML Security:
 - All your Clouds Are Belong to us: Security Analysis of Cloud Management Interfaces (CCSW'11)
 - How to Break XML Encryption (CCS'11)
 - On Breaking SAML: Be Whoever you Want to Be (USENIX'12)
 - On the Insecurity of XML Security (Dissertation)
- Further topics:
 - Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks (USENIX'14)
 - Untrusted Third Parties: When IdPs Break Bad (in submission, by my colleagues Christian Mainka, Vladislav Mladenov and Jörg Schwenk)

About this talk

- Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks
- Paper accepted at Usenix Security 2014
- Authors: Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, Erik Tews
- Describes new side channels in specific TLS implementations

Overview



TLS

- Bleichenbacher's Attack
 - Attack Intuition
 - Oracle Strength
 - Attack Challenges
- Attacks
 - Error Messages in JSSE
 - Additional Random Number Generation
 - Additional Exception in JSSE
 - Unexpected Timing Behavior by Hardware Appliances
- Conclusion

TLS

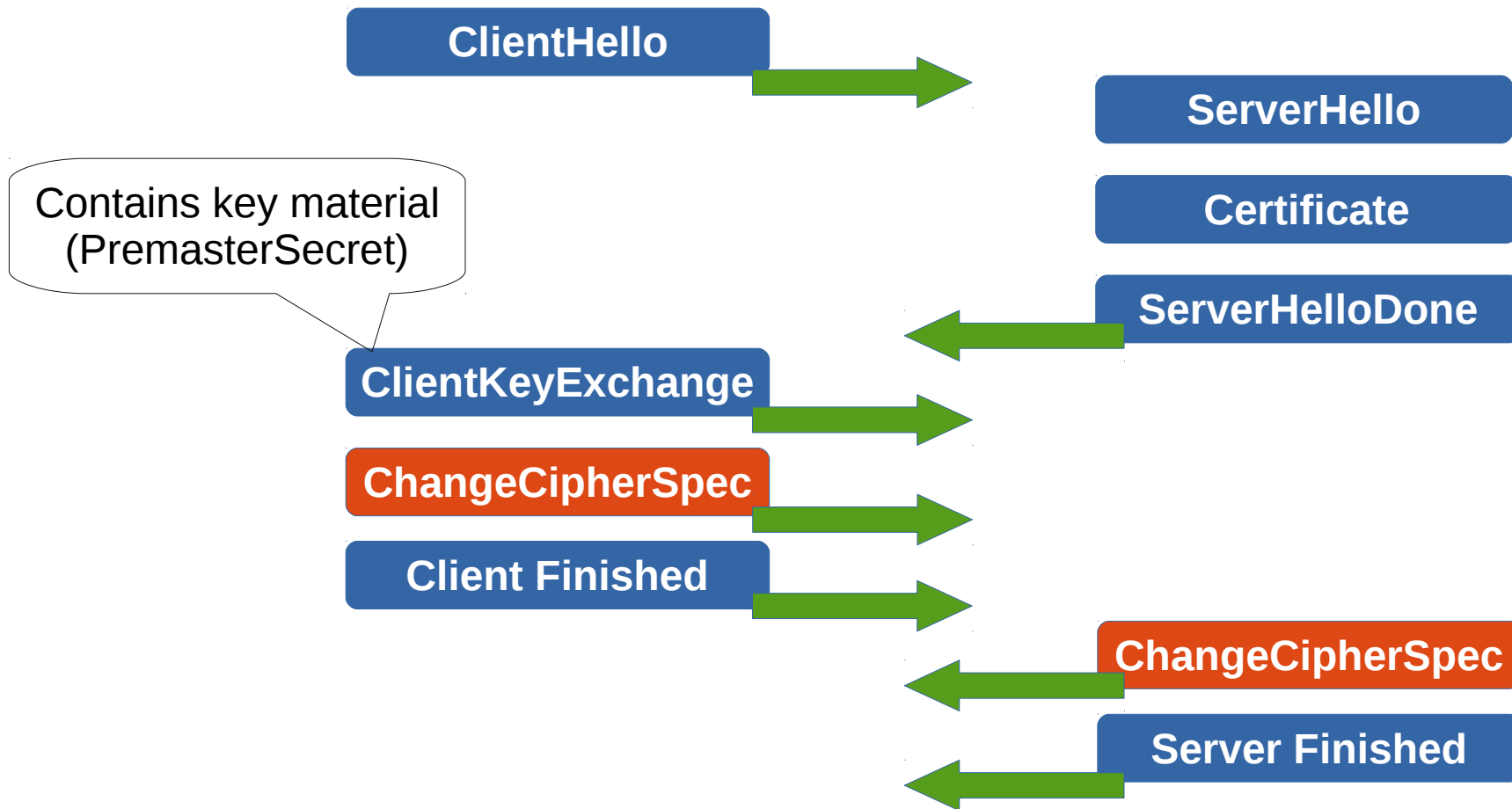
- Invented by Netscape in 1994
 - Name: Secure Sockets Layer
- Adopted by IETF in 1999
 - Renamed to Transport Layer Security
- Versions:
 - SSL 1.0, 2.0, 3.0
 - TLS 1.0, 1.1, 1.2, (1.3 in development)
- Implementations:
 - OpenSSL, GnuTLS, JSSE, Microsoft Schannel, MatrixSSL, LibreSSL, ...

TLS

- Very complex
- Contains various crypto primitives: RSA, EC, AES-CBC, AES-GCM, RC4, 3DES, MD5, SHA1, MACs, Signatures, PRFs, ...
- Can be executed over TCP or UDP (DTLS)
- Contains various extensions
- TLS-Renegotiation

TLS Handshake

- Used for negotiation of cryptographic keys for data transport



ClientKeyExchange

- Contains encrypted PremasterSecret (for example, encrypted using RSA or EC)
- PremasterSecret is used to derive all TLS session keys
- Decryption of PremasterSecret == decryption of the TLS traffic



Snidely Whiplash
(Dudley Do-Right of the Mounties)

Overview

- TLS

-  Bleichenbacher's Attack

- Attack Intuition
- Oracle Strength
- Attack Challenges

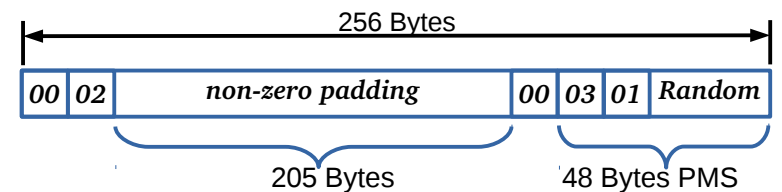
- Attacks

- Error Messages in JSSE
- Additional Random Number Generation
- Additional Exception in JSSE
- Unexpected Timing Behavior by Hardware Appliances

- Conclusion

RSA PKCS#1 v1.5 Encryption

- Used e.g. to distribute symmetric keys
- Textbook-RSA: $C_{RSA} = m^e \bmod N$
 - Short messages need padding
 - No randomization
- PKCS#1 adds randomized padding to the PremasterSecret, it works as follows:
 - Take a PremasterSecret PMS
 - Set $m := 00 \parallel 02 \parallel \text{pad} \parallel 00 \parallel \text{PMS}$
 - Compute $C_{PKCS} = m^e \bmod N$
- A ciphertext is “valid”, if its decryption has the correct format



Bleichenbacher's Attack

- 1998: Attack on RSA-PKCS#1 v1.5 (Bleichenbacher, Crypto 1998)
- SSL implementations applied an ad-hoc fix
- Well-noticed in crypto and security community
- PKCS#1 was updated to v2.0 (RSA-OAEP)
 - Still standardized in many applications, including TLS

Attack Applied to ...

- SSL / TLS:
 - D. Bleichenbacher: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1, Crypto'98
- Cryptographic Hardware:
 - Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Graham Steel, and Joe-Kai Tsay. Efficient Padding Oracle Attacks on Cryptographic Hardware, Crypto'12
- XML Encryption:
 - Tibor Jager, Sebastian Schinzel, Juraj Somorovsky: Bleichenbacher's Attack Strikes Again: Breaking PKCS#1 v1.5 in XML Encryption, ESORICS'12

Motivation

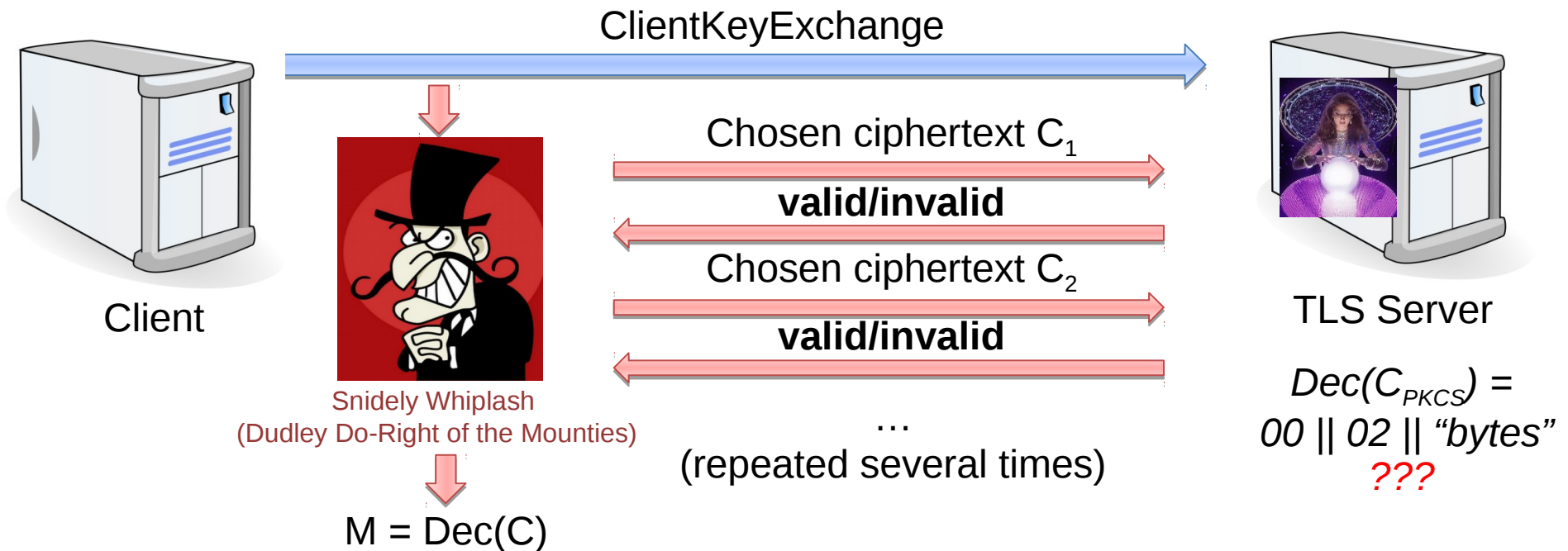
- Attack worked in 1998...
- Is PKCS#1 v1.5 implemented correctly in TLS now?

Overview

- TLS
- Bleichenbacher's Attack
 - Attack Intuition
 - Oracle Strength
 - Attack Challenges
- Attacks
 - Error Messages in JSSE
 - Additional Random Number Generation
 - Additional Exception in JSSE
 - Unexpected Timing Behavior by Hardware Appliances
- Conclusion

Bleichenbacher's Attack

- Requires a “ciphertext validity oracle”
- Adaptive Chosen-ciphertext attack



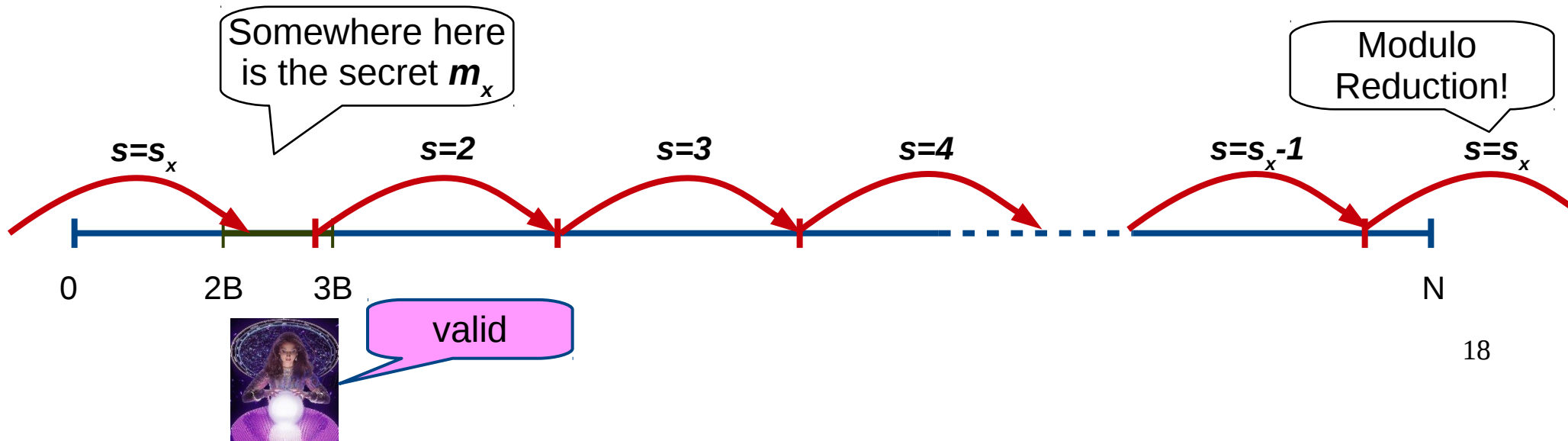
Attack Intuition

- d : private key
- (e, N) : public key
- $m = 00 \parallel 02 \parallel \text{“bytes”}$

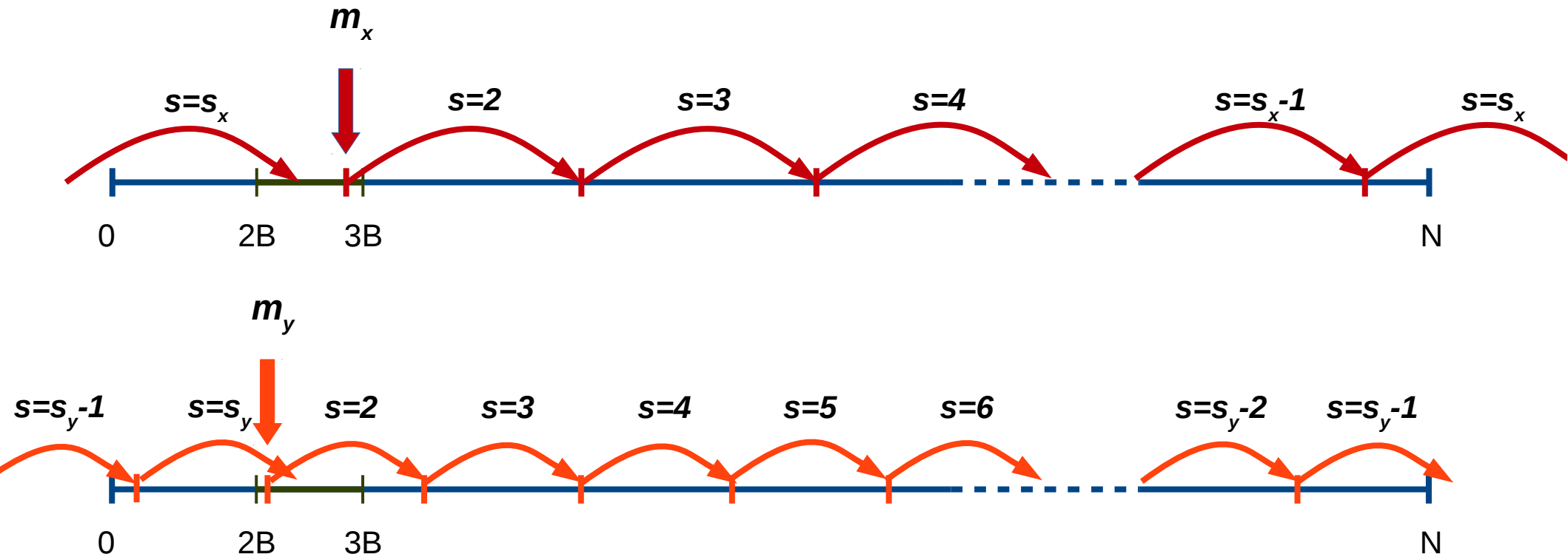
- In RSA we can multiply the encrypted plaintext **without knowing the private key**
- $m = c^d \bmod N$
- $c = m^e \bmod N$
- $c' = (c \cdot s^e) \bmod N \quad s \in Z_N$
- $c' = (ms)^e \bmod N$

Attack Intuition

- OK, so we can multiply a plaintext ...
- We define: $B = 2^{(|N|-2)}$, where $|N|$ is byte length
 - Example: $2B = 00\ 02\ 00\ \dots\ 00$
- Attack Approach:
 - Multiply “plaintext” with s : $c' = (c \cdot s^e) \bmod N$
 - Query oracle if the decrypted plaintext is in interval $<2B, 3B)$



Attack Intuition



- $S_y > S_x$
- Intuition:
 - **Large s** value indicates m is in the near of **2B**
 - **Small s** value indicates m is in the near of **3B**

Attack

- s_x allows us to compute new interval for m :

$$2B \leq m_x s_x - N < 3B$$

- From this follows:

$$(2B + N) / s_x < m_x < (3B + N) / s_x$$

- Full algorithm:
 - Searches for further s values
 - Reduces the interval

Demo Time

Attack Countermeasure

generate a **random** PMS_R

decrypt the ciphertext: $m := \text{dec}(c)$


if ($(m \text{ ? } 00 || 02 || PS || 00 || k)$ OR $(|k| \text{ ? } 48)$) then

 proceed with $PMS := PMS_R$

else

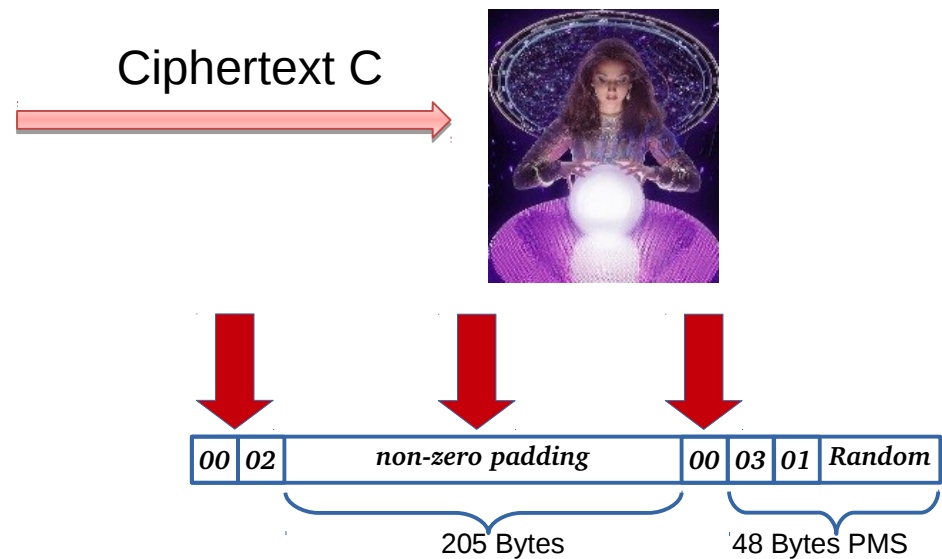
 proceed with $PMS := k$

Overview

- TLS
- Bleichenbacher's Attack
 - Attack Intuition
 -  Oracle Strength
 - Attack Challenges
- Attacks
 - Error Messages in JSSE
 - Additional Random Number Generation
 - Additional Exception in JSSE
 - Unexpected Timing Behavior by Hardware Appliances
- Conclusion

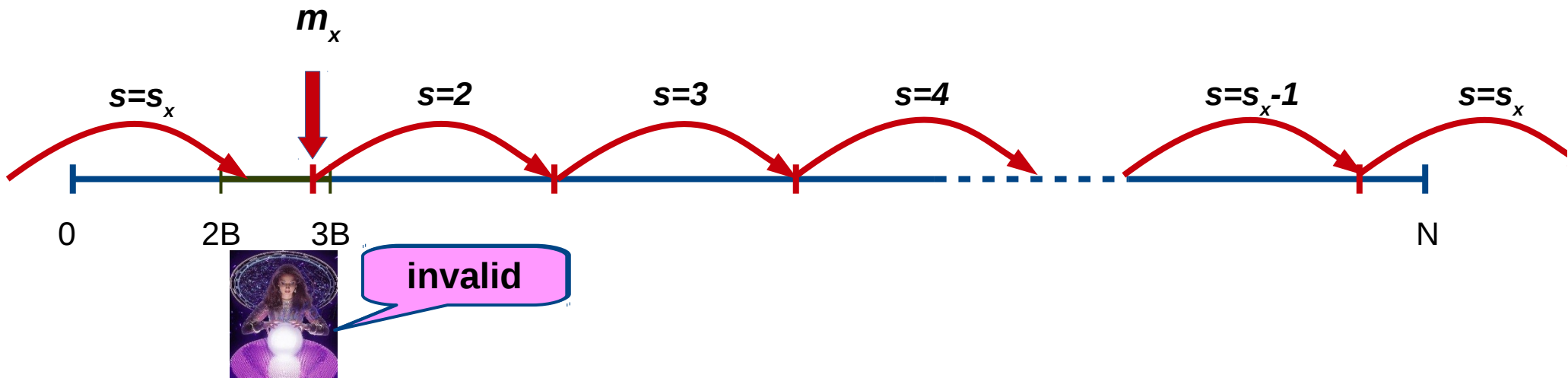
Attack Performance

- Bleichenbacher's attack is also called Million Messages attack
- The attack performance varies: it depends on the oracle message validation
- The oracle responds with “valid” when:
 - The message starts with 00 02
 - (and) the PremasterSecret is of valid length?
 - Further checks?



Oracle Strength

- Oracle with **less checks** brings **better performance**
- Oracle strength: Probability the oracle responds with “valid” when the message starts with 00 02
- Why important?

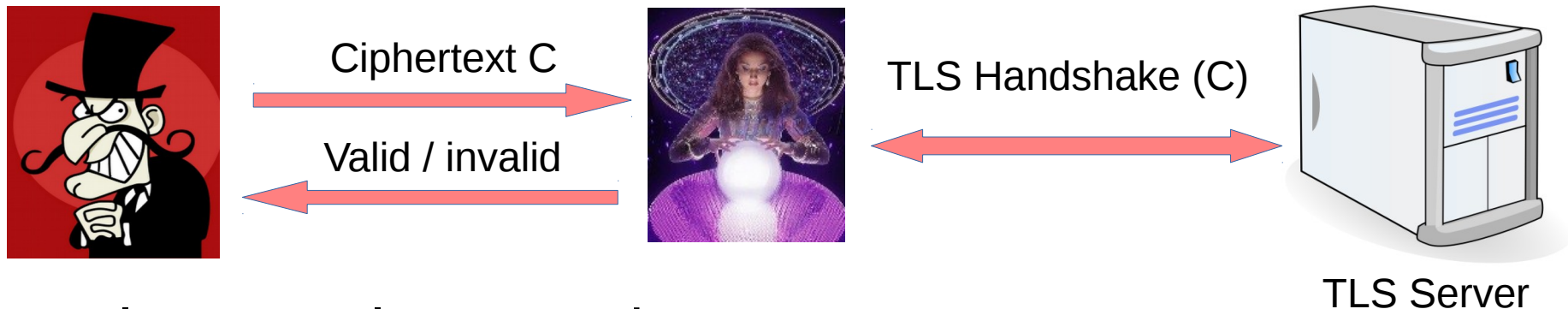


Overview

- TLS
- Bleichenbacher's Attack
 - Attack Intuition
 - Oracle Strength
 - Attack Challenges
- Attacks
 - Error Messages in JSSE
 - Additional Random Number Generation
 - Additional Exception in JSSE
 - Unexpected Timing Behavior by Hardware Appliances
- Conclusion

Attack Challenges

- Implement an oracle based on the server behavior
 - Using different error messages, timing



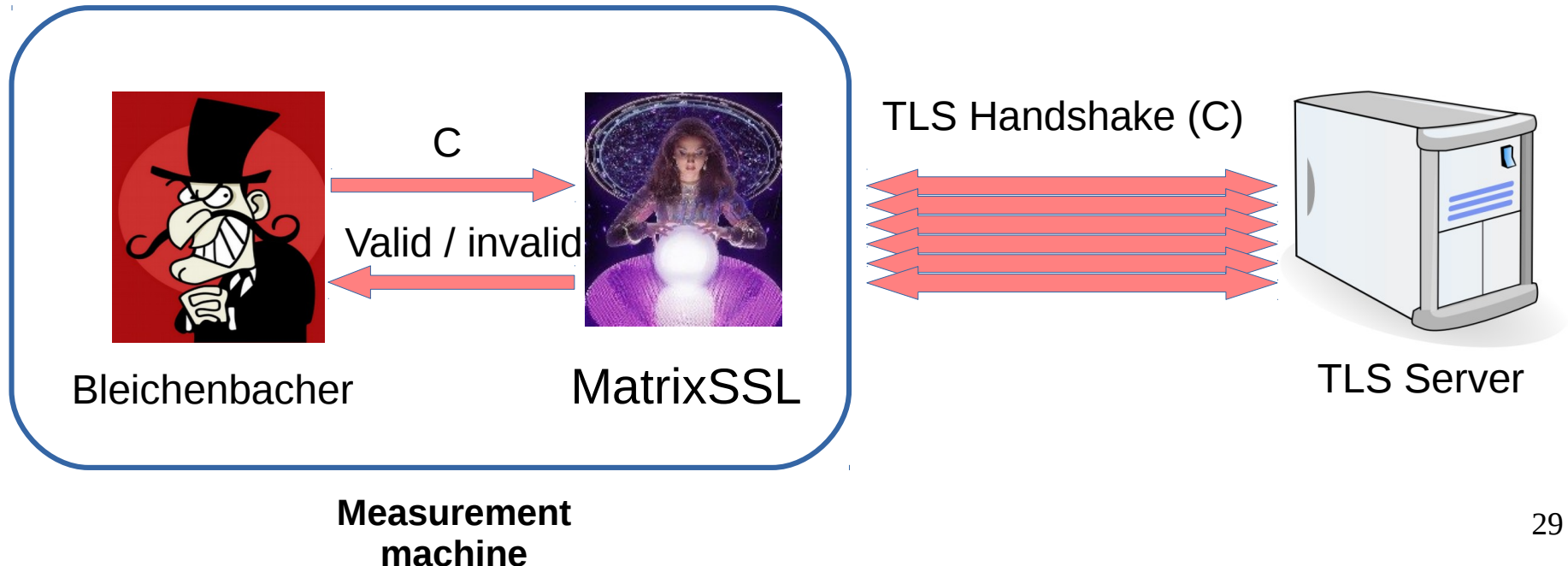
- Analyze oracle strength
 - Probability
 - If timing: how many server requests are needed to respond one oracle request
- Execute Bleichenbacher's attack

With the help of T.I.M.E.

- T.I.M.E.: TLS Inspection Made Easy
- Automatic scanning of TLS implementations
- Written (mainly) by Christopher Meyer:
 - <http://www-brs.ub.ruhr-uni-bochum.de/net/html/HSS/Diss/MeyerChristopher/diss.pdf>
- Supports further features like TLS fingerprinting

For Timing Measurements...

- T.I.M.E. was not appropriate, caused too much noise
- We used our Bleichenbacher attack module with a patched MatrixSSL library
- NetTimer for response times evaluation:
 - <http://sebastian-schinz.de/nettimer>



Overview

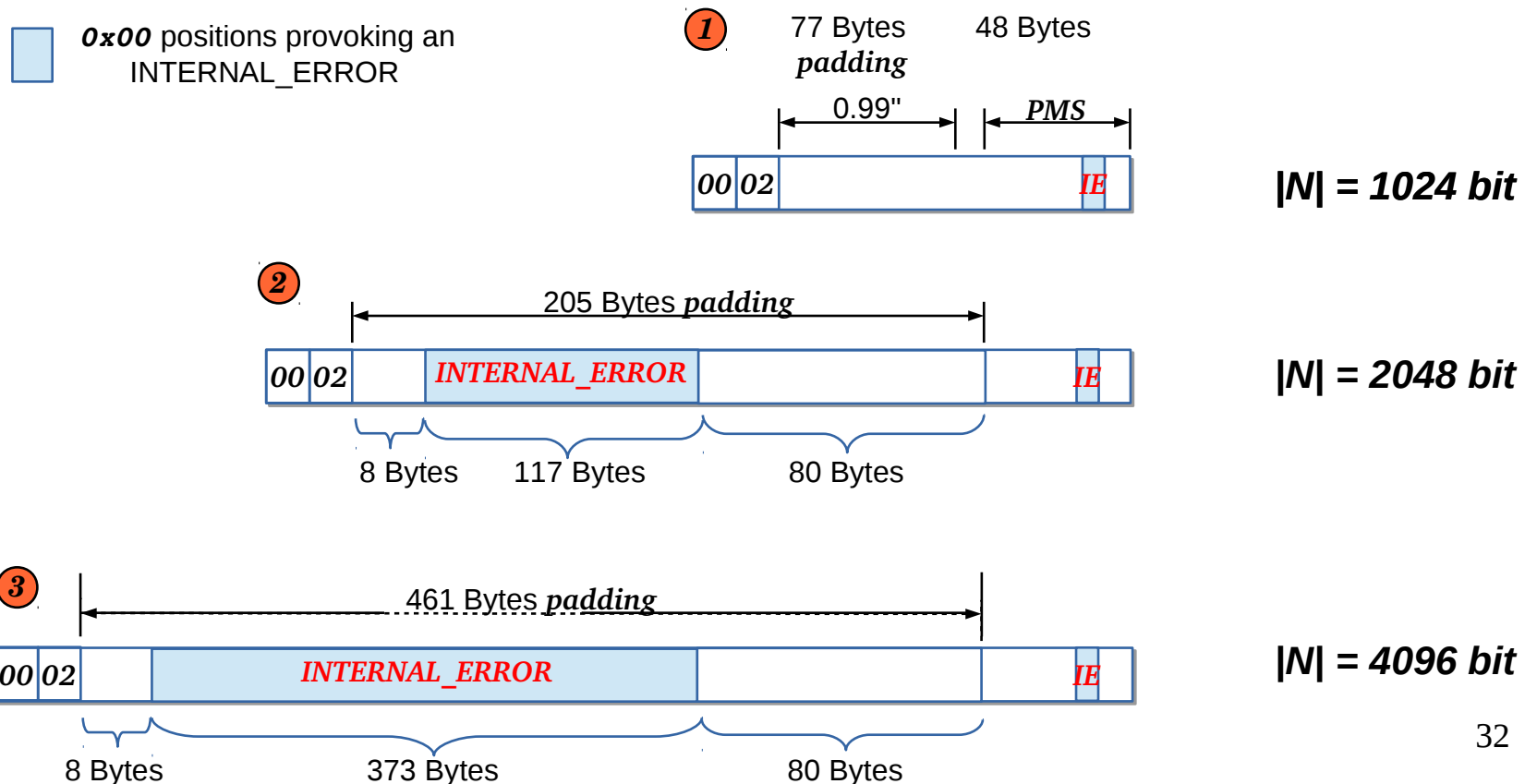
- TLS
- Bleichenbacher's Attack
 - Attack Intuition
 - Oracle Strength
 - Attack Challenges
- Attacks
 - Error Messages in JSSE
 - Additional Random Number Generation
 - Additional Exception in JSSE
 - Unexpected Timing Behavior by Hardware Appliances
- Conclusion

Error Messages in JSSE

- With T.I.M.E. we sent differently formatted PKCS#1 messages to a JSSE server
- Server responded with:
 - INTERNAL ERROR and
 - HANDSHAKE FAILURE

Analysis

- 0x00 bytes inserted at specific positions cause an internal **ArrayIndexOutOfBoundsException**
- Lead to a different TLS alert message



Oracle Strength

- We were able to construct an oracle:
 - INTERNAL_ERROR: message **valid**, starts with 00 02
 - HANDSHAKE FAILURE: message **invalid**
- What is the probability for triggering INTERNAL_ERROR?

- 2048 bit key:

- Number of bytes provoking INTERNAL_ERROR: 117
- Probability:

$$P_{2048} = (255/256)^8 (1 - (255/256)^{117}) = 35 \%$$

- 4096 bit key:

$$P_{4096} = 74 \%$$

- 1024 bit key:

$$P_{1024} = 0,2 \%$$


Evaluation

	Mean	Median
2048 bit RSA key	177 000	37 000
4096 bit RSA key	73 000	28 000

```
INFO [main] 26 Sep 2012 19:35:50,388 - Step 2c: Searching with one interval left
INFO [main] 26 Sep 2012 19:35:50,726 - Found s2015:
2353474280691358986415452724171276939586296506243294697426729369327166239800462513683715910529286402005811714103895479929288033060730341292865411
INFO [main] 26 Sep 2012 19:35:50,726 - Step 3: Narrowing the set of solutions.
INFO [main] 26 Sep 2012 19:35:50,726 - # of intervals for M2015: 1
INFO [main] 26 Sep 2012 19:35:50,726 - Step 4: Computing the solution.
INFO [main] 26 Sep 2012 19:35:50,726 - // Total # of queries so far: 456355
INFO [main] 26 Sep 2012 19:35:50,727 - Step 2: Searching for PKCS conforming messages.
INFO [main] 26 Sep 2012 19:35:50,727 - Step 2c: Searching with one interval left
INFO [main] 26 Sep 2012 19:35:51,305 - Found s2016:
4706948561382717972830905448342553879172593012486589394853458738654332479600925027367431821058572804011623428207790959858576066121460682585730823
INFO [main] 26 Sep 2012 19:35:51,305 - Step 3: Narrowing the set of solutions.
INFO [main] 26 Sep 2012 19:35:51,306 - # of intervals for M2016: 1
INFO [main] 26 Sep 2012 19:35:51,306 - Step 4: Computing the solution.
INFO [main] 26 Sep 2012 19:35:51,306 - =====> Solution found! -> PreMasterSecret
00 02 f5 a7 9f cd b1 27 f9 39 15 21 49 71 65 97 33 99 6d 9b cd 6d 4b e3 f5 fd b5 71 d5 69 71 91 b9 39 c9 6d f5 59 f1 b9 97 b7 6b ff 33 d1 9b 85 13 d
39 1b 00 03 01 06 26 a6 40 57 4b 50 d6 a3 d0 8a 70 16 0a 0d af 33 2a 7f 9b c8 65 a7 b5 54 e7 48 9f 57 da c9 bf 34 8b 8d d4 84 ed c9 63 2b 16 6f 2
INFO [main] 26 Sep 2012 19:35:51,306 - // Total # of queries so far: 456370
```

- Attack on server with 1024 bit keys not practical because of the weak oracle
- Patched in October 2012 – JDK 6, Update 37 (JDK 6u37): CVE-2012-5081

Overview

- TLS
- Bleichenbacher's Attack
 - Attack Intuition
 - Oracle Strength
 - Attack Challenges
- Attacks
 - Error Messages in JSSE
 -  Additional Random Number Generation
 - Additional Exception in JSSE
 - Unexpected Timing Behavior by Hardware Appliances
- Conclusion

Additional Random Number Generation

- **Recommended Countermeasure:**

```
generate a random PMSR
decrypt the ciphertext: m := dec(c)
if ( (m ? 00||02||PS||00||k) OR
      (|k| ? 48) ) then
    proceed with PMS := PMSR
else
    proceed with PMS := k
```

- **Countermeasure in OpenSSL, GnuTLS, ...:**

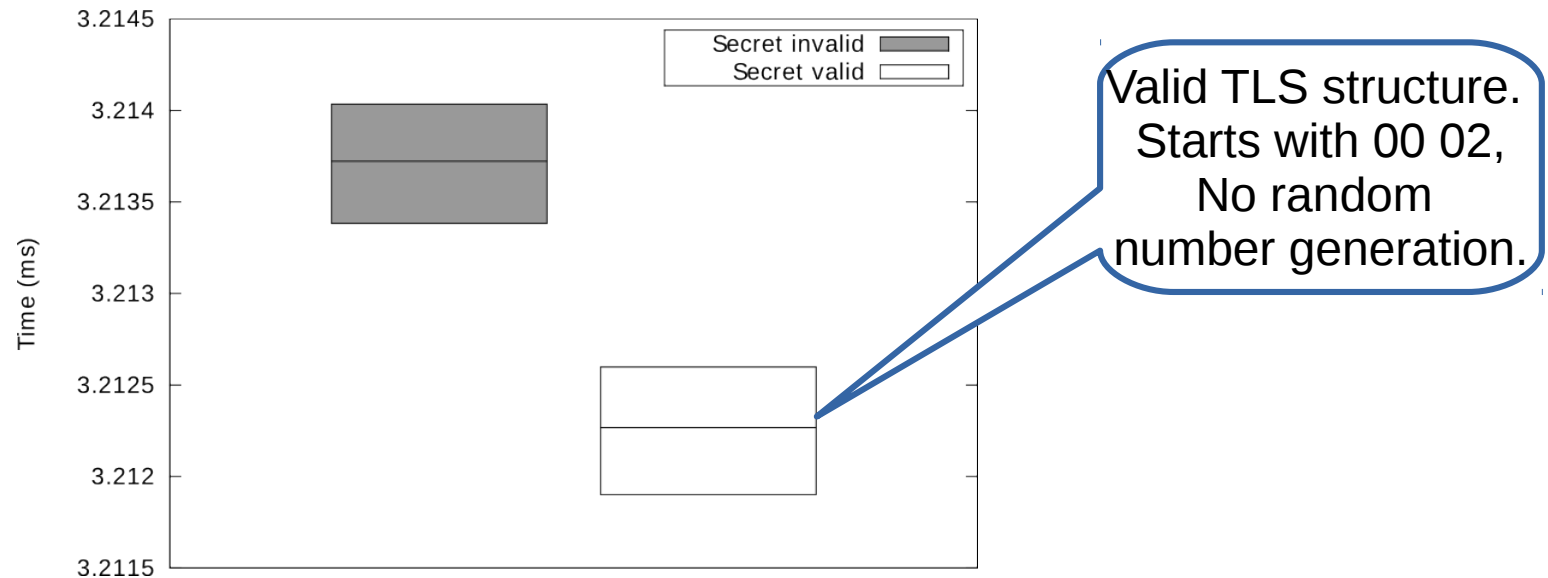
```
decrypt the ciphertext: m := dec(c)
if ( (m ? 00||02||PS||00||k) OR
      (|k| ? 48) ) then
    generate a random PMSR
    proceed with PMS := PMSR
else
    proceed with PMS := k
```

Analysis

- We saw this in more implementations
- Important observation: Random PMS generated only in case of invalid decryption step
- Does this misbehavior allow us to execute practical attacks?

Oracle Strength

- We were able to measure different timing responses, however the timing difference was very small (cca. 2 microseconds)




- Probability of returning a valid message small:

$$P = 2,7 * 10^{-8}$$

Evaluation

- Attack not practical
- Too many oracle queries
- The timing difference too small

Overview

- TLS
- Bleichenbacher's Attack
 - Attack Intuition
 - Oracle Strength
 - Attack Challenges
- Attacks
 - Error Messages in JSSE
 - Additional Random Number Generation
 -  Additional Exception in JSSE
 - Unexpected Timing Behavior by Hardware Appliances
- Conclusion

Additional Exception in JSSE

- PKCS#1 unpadding function in Java:

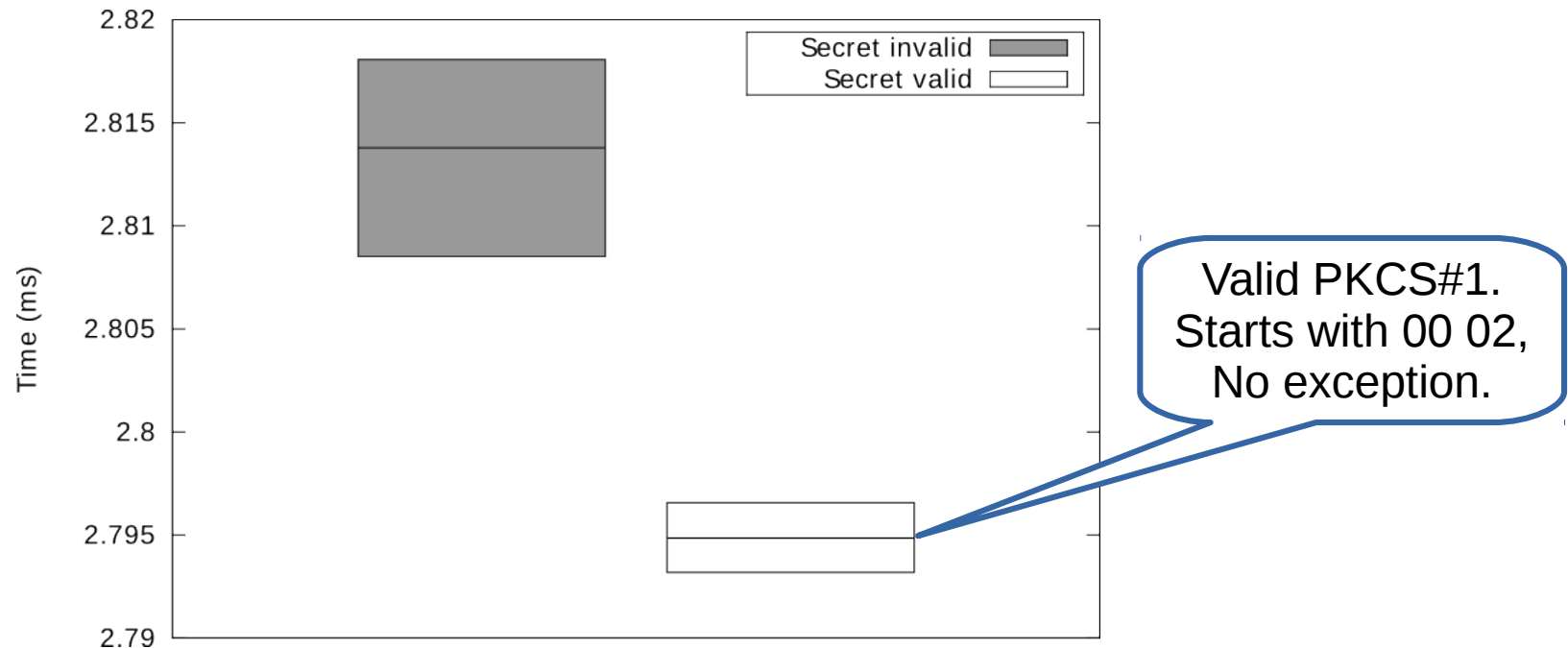
```
private byte [] unpadV15 (byte[] padded) throws
BadPaddingException {
    if (not PKCS compliant) {
        throw new BadPaddingException();
    } else {
        return unpadding text;
    }
}
```

Analysis

- We tested the JSSE server with different valid and invalid PKCS#1 messages
- We were not able to trigger a different alert...
- ...but we saw **an additional exception** in case of invalid message

Oracle Strength

- We evaluated that an additional exception consumes about 20 microseconds!

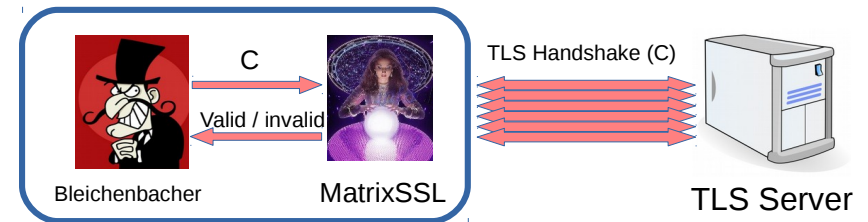


- Enough to measure over LAN

Oracle Strength

- We were able to construct an oracle:
 - Shorter time: message **valid**, PKCS#1 compliant
 - Longer time: message **invalid**, additional exception produced
- Large probability of about 60%

Evaluation



- Attack evaluation:
 - About 20 000 oracle queries to decrypt a PMS
 - Each oracle query takes about 500 server queries
 - 20% false negatives, no false positive
 - 20 hours, over LAN
 - Executed against OpenJDK and Oracle JDK
- Patched in January 2014 – JDK 7, Update 45: CVE-2014-411
- Similar behavior found in Bouncy Castle (Java and C#)
 - Reported, not fixed

Overview

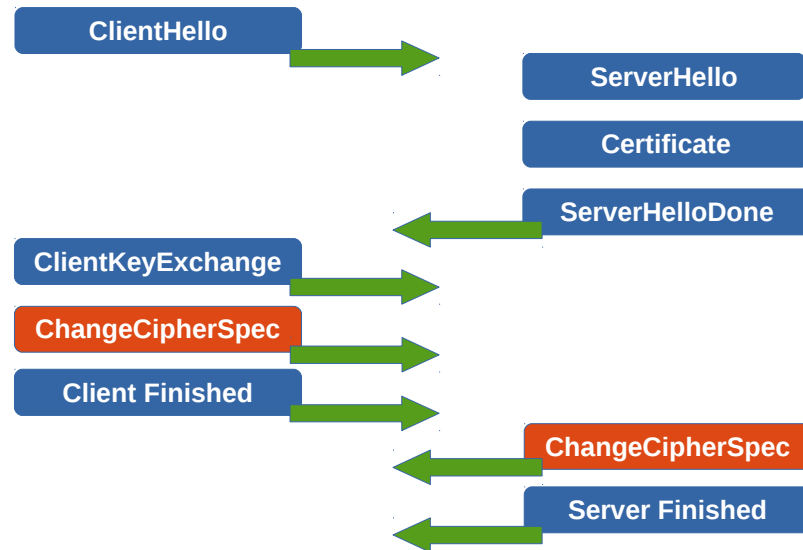
- TLS
- Bleichenbacher's Attack
 - Attack Intuition
 - Oracle Strength
 - Attack Challenges
- Attacks
 - Error Messages in JSSE
 - Additional Random Number Generation
 - Additional Exception in JSSE
- Unexpected Timing Behavior by Hardware Appliances
- Conclusion

Unexpected Timing Behavior by Hardware Appliances

- We used T.I.M.E. to execute TLS handshakes using malformed PKCS#1 messages
- Our Hardware Appliance accepted malformed PKCS#1 formatted PremasterSecrets:
 - 01 02 ... 00 PMS
 - 02 02 ... 00 PMS
 - 03 02 ... 00 PMS
- The first byte was not checked at all and we could execute valid TLS handshakes

Analysis

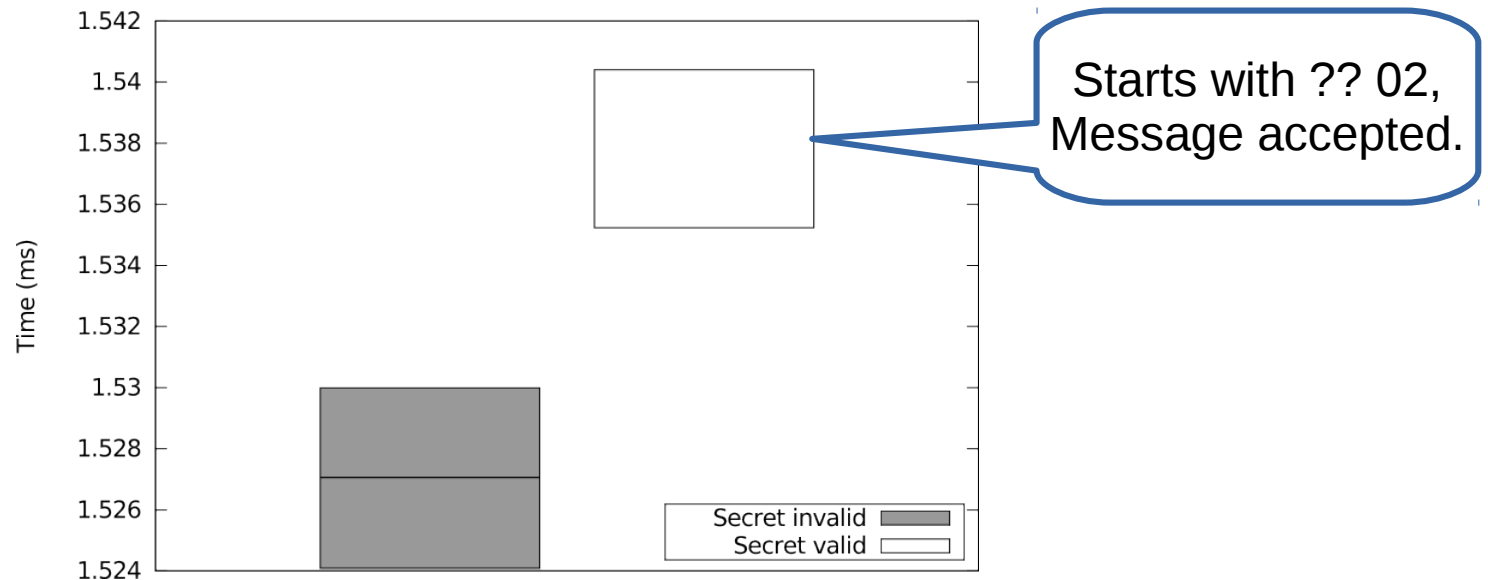
- It was not directly exploitable
 - the attacker is not able to produce valid ClientFinished messages



- ... but we smelled a timing leakage in the PKCS#1 processing
- Black box analysis

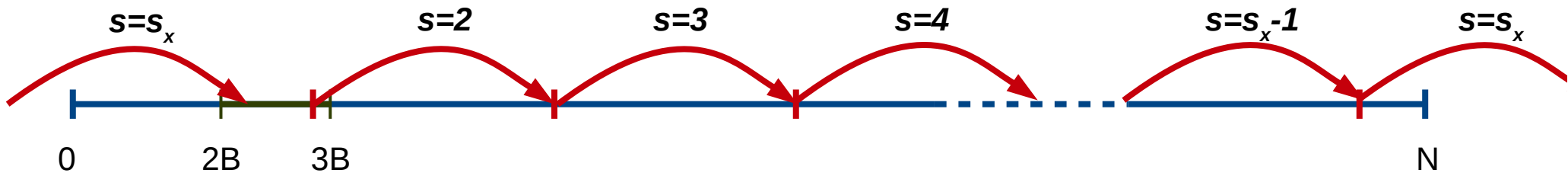
Oracle Strength

- We found a timing difference of about 15 microseconds between messages starting with **?? 02** and other messages (?? indicates an arbitrary byte)



Oracle Strength

- We were able to construct an oracle:
 - Longer time: message **valid**, starts with **?? 02**
 - Shorter time: message **invalid**, different second byte
- The oracle is not “Bleichenbacher” compliant



Evaluation

- We extended Bleichenbacher's attack to work with our oracle
- Performance improvement:
 - About 4700 oracle queries to decrypt a PMS
- Real attack:
 - 7371 oracle queries
 - 4 000 000 server queries at total
 - 40 hours
 - 1290 false negatives, no false positive
- Developers notified, be prepared to update your appliances
- Public disclosure in August

Overview

- TLS
- Bleichenbacher's Attack
 - Attack Intuition
 - Oracle Strength
 - Attack Challenges
- Attacks
 - Error Messages in JSSE
 - Additional Random Number Generation
 - Additional Exception in JSSE
 - Unexpected Timing Behavior by Hardware Appliances

 Conclusion

Conclusion and Outlook

- We showed first practical timing Bleichenbacher attacks on TLS
- A tiny side channel can lead to catastrophic results

TLS impl.	Type	Queries	Time
OpenSSL	timing		NA
JSSE	direct	177 000	12 h
JSSE	timing	18 600	20 h
Hardware	timing	7 400	41 h

- Crypto code should be handled with care, especially when assuming local attackers: e.g., crypto in browser
- We motivate for the usage of secure cryptographic primitives
- Future Work:
 - Analysis of further crypto standards
 - Development of TLS penetration tools

