BLEICHENBACHER ATTACK

Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1 v1.5

René Behring, Johannes Heßling, Damian Poddebniak

2. Juli 2015

Fachhochschule Münster

EINLEITUNG

- Angreifer hat temporär die Möglichkeit, Ciphertexte seiner Wahl zu entschlüsseln[1]
    - Zugriff auf Hardwaresystem (Einbruch, …)
    - Zugriff auf unvorhergesehene Nebeneffekte (“Side-channels”)
    - …

[1] https://de.wikipedia.org/wiki/Kryptoanalyse#Angriffsszenarien

EINLEITUNG – ADAPTIVE CHOSEN-CIPHERTEXT ATTACK

- Ähnlichkeit zur Chosen-ciphertext attack
  - Längere Zeit Zugang zum System
  - Kann nach jeder Analyse gezielt einen neuen Ciphertext wählen
  - Nutzt vorherige Informationen um neuen Ciphertext anzupassen

Mögliche Ziele:

- Entschlüsselung eines abgefangenen Ciphertext
- Herausfinden von privaten Schlüsseln
- ...

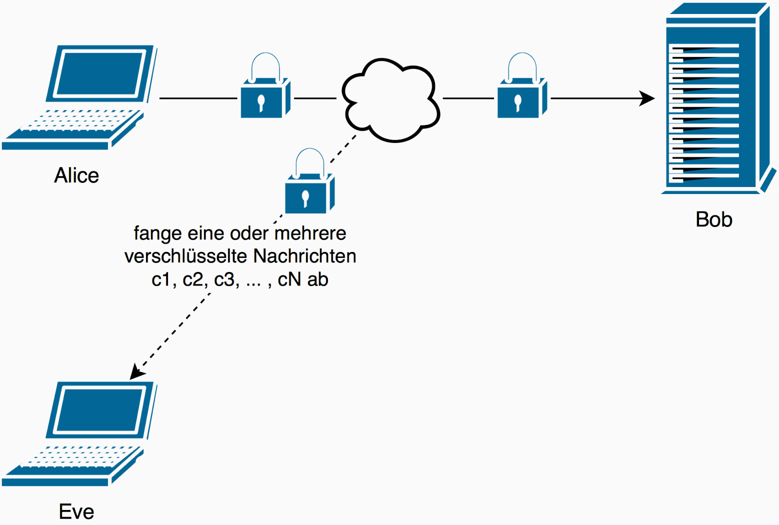

Abbildung: Aufbau und Kontext

EINLEITUNG – ANGRIFFSSZENARIO

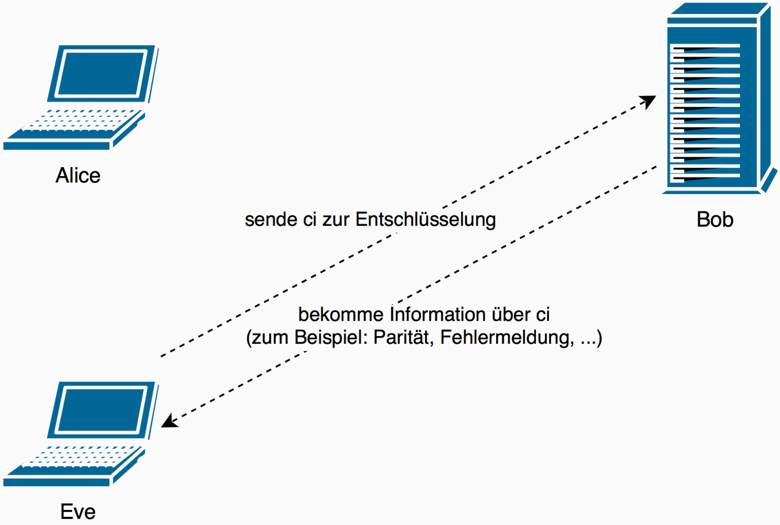

Abbildung: Aufbau und Kontext

Homomorphie-Eigenschaft von RSA:

$$E(m) \cdot E(s) \bmod n = E(m \cdot s \bmod n)$$

Beweis...

$$\begin{aligned} E(m) \cdot E(s) &= m^e \cdot s^e \bmod n \\ &= (m \cdot s)^e \bmod n \\ &= (m \cdot s \bmod n)^e \bmod n \end{aligned}$$

Das Multiplizieren eines Ciphertextes mit s^e entspricht der Multiplikation des Klartextes mit s.

DAS PARITÄTS-ORAKEL

Zunächst ein einfaches Beispiel:

Das “Paritäts-Orakel”.

Funktionsweise:

- Server empfängt einen Ciphertext
- Server entschlüsselt den Ciphertext und leaked das letzte Bit des dazugehörigen Plaintextes

Was kann da schon passieren...?

Öffentlich zugängliche Informationen:

- RSA-Modulus n
- Öffentlicher Exponent e
- …

Zusätzlich:

- Abgehörter Ciphertext c_0.
- Paritäts-Orakel O, welches die Parität von $m_0 = D(c_0)$ leaked.

Vorgehen

- **Multipliziere den <u>Klartext</u> mit** 2 **und prüfe seine Parität**[2]
- Multipliziere den Klartext mit 4 und prüfe seine Parität
- Multipliziere den Klartext mit 8 und prüfe seine Parität
- ...
- Multipliziere den Klartext mit 2^X und prüfe seine Parität

[2] Funktioniert aufgrund der Homomorphie-Eigenschaften von RSA.

Für unser Beispiel nutzen wir einen RSA-Modulus $n = 15$ und einen gegebenen Ciphertext c_0.

Annahme

Der Ciphertext c_0 kann mit $\log_2(n)$ Orakel-Aufrufen entschlüsselt werden.

Vereinfachung

Statt c_0 verwenden wir m_0 – die Funktionsweise ist die gleiche.

PARITÄTS-ORAKEL – VORGEHEN

Verwende zunächst nur abgeleitete Annahmen...

1. Wir wissen: $0 \leq m_0 \leq 14$.
2. Multipliziere m_0 mit 2.

$$\begin{aligned} m_1 &= 2m_0 \bmod n \\ &= 2m_0 - rn \end{aligned}$$

3. Stelle nach m_0 um.

$$\begin{aligned} m_0 &= \frac{m_1 + rn}{2} \\ \Rightarrow m_0 &\in \left[\frac{0 + rn}{2}, \frac{14 + rn}{2}\right] \end{aligned}$$

$$m_0 \in \left[\frac{0 + rn}{2}, \frac{14 + rn}{2}\right]$$

4. Welche Werte kann r annehmen?

$$\begin{aligned} m_1 &= 2m_0 - rn \\ r &= \frac{2m_0 - m_1}{n} \Rightarrow \left\lceil \frac{-14}{15} \right\rceil \leq r \leq \left\lfloor \frac{28}{15} \right\rfloor \end{aligned}$$

5. Die Zahl r kann die Werte 0 und 1 annehmen.

$r = 0$	$r = 1$
$0 \leq m_0 \leq 7$	$\frac{15}{2} \leq m_0 \leq \frac{29}{2}$

$r = 0$	$r = 1$
$0 \leq m_0 \leq 7$	$\frac{15}{2} \leq m_0 \leq \frac{29}{2}$

Was bringt uns das?

Frage das Orakel...

- Wenn $2^e \cdot c_0 \bmod n \xrightarrow{0}$ "gerade" $\Rightarrow m \in [0, 7]$
- Wenn $2^e \cdot c_0 \bmod n \xrightarrow{0}$ "ungerade" $\Rightarrow m \in [8, 14]$

Vorgehen

- Multipliziere den Klartext mit 2 und prüfe seine Parität
- **Multipliziere den Klartext mit 4 und prüfe seine Parität**
- Multipliziere den Klartext mit 8 und prüfe seine Parität
- ...
- Multipliziere den Klartext mit 2^X und prüfe seine Parität

PARITÄTS-ORAKEL – VORGEHEN

$$m_0 \in \left[\frac{0 + rn}{4}, \frac{14 + rn}{4}\right]$$

4. Welche Werte kann r annehmen?

$$m_1 = 4m_0 - rn$$
$$r = \frac{4m_0 - m_1}{n} \Rightarrow \left\lceil \frac{-14}{15} \right\rceil \leq r \leq \left\lfloor \frac{56}{15} \right\rfloor$$

5. Die Zahl r kann die Werte 0, 1, 2 und 3 annehmen.

$r = 0$	$r = 1$	$r = 2$	$r = 3$
$\frac{0}{4} \leq m_0 \leq \frac{14}{4}$	$\frac{15}{4} \leq m_0 \leq \frac{29}{4}$	$\frac{30}{4} \leq m_0 \leq \frac{44}{4}$	$\frac{45}{4} \leq m_0 \leq \frac{59}{4}$

$$m_0 \in \left[\frac{0 + rn}{4}, \frac{14 + rn}{4}\right]$$

4. Welche Werte kann r annehmen?

$$m_1 = 4m_0 - rn$$

$$r = \frac{4m_0 - m_1}{n} \Rightarrow \left\lceil \frac{-14}{15} \right\rceil \leq r \leq \left\lfloor \frac{56}{15} \right\rfloor$$

5. Die Zahl r kann die Werte 0, 1, 2 und 3 annehmen.

$r = 0$	$r = 1$	$r = 2$	$r = 3$
$0 \leq m_0 \leq 3$	$4 \leq m_0 \leq 7$	$8 \leq m_0 \leq 11$	$12 \leq m_0 \leq 14$

PARITÄTS-ORAKEL – VORGEHEN

Frage erneut das Orakel...

- Wenn $2^e \cdot c_0 \bmod n \xrightarrow{0}$ "gerade" $\Rightarrow m \in [0, 3]$
- Wenn $2^e \cdot c_0 \bmod n \xrightarrow{0}$ "ungerade" $\Rightarrow m \in [4, 7]$
- Wenn $2^e \cdot c_0 \bmod n \xrightarrow{0}$ "gerade" $\Rightarrow m \in [8, 11]$
- Wenn $2^e \cdot c_0 \bmod n \xrightarrow{0}$ "ungerade" $\Rightarrow m \in [12, 14]$

PARITÄTS-ORAKEL

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Abbildung: Paritäts-Orakel

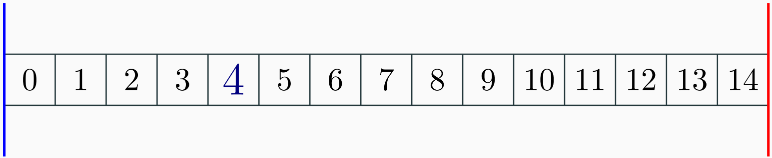

Abbildung: Paritäts-Orakel

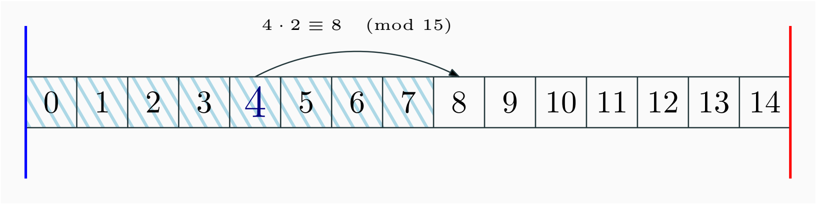

Abbildung: Paritäts-Orakel

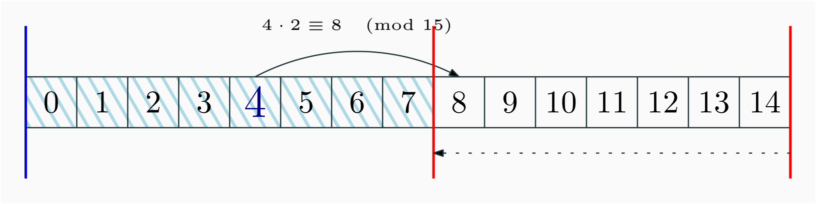

Abbildung: Paritäts-Orakel

PARITÄTS-ORAKEL

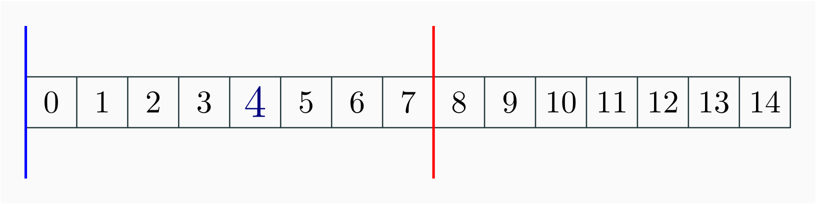

Abbildung: Paritäts-Orakel

PARITÄTS-ORAKEL

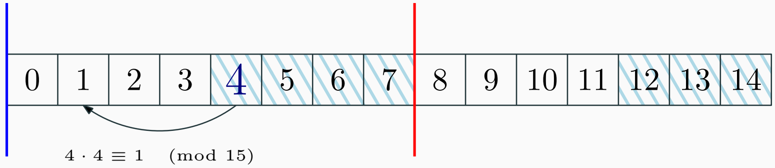

Abbildung: Paritäts-Orakel

PARITÄTS-ORAKEL

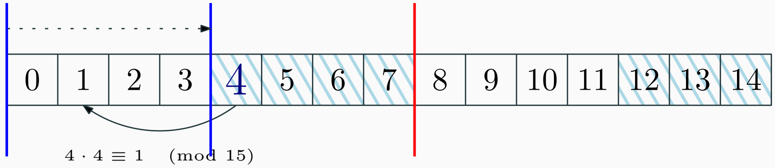

Abbildung: Paritäts-Orakel

PARITÄTS-ORAKEL

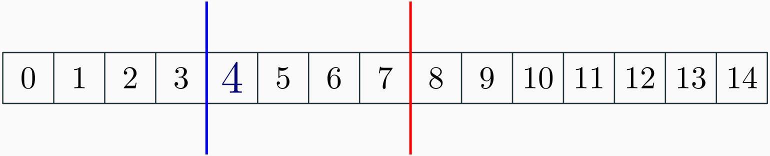

Abbildung: Paritäts-Orakel

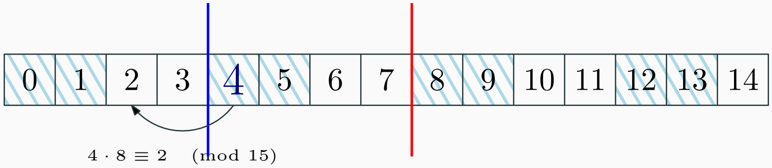

Abbildung: Paritäts-Orakel

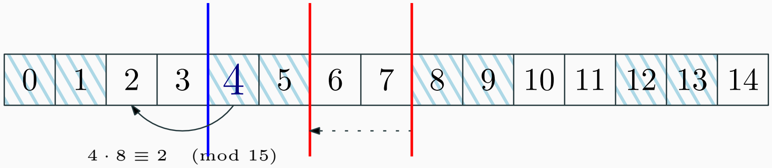

Abbildung: Paritäts-Orakel

PARITÄTS-ORAKEL

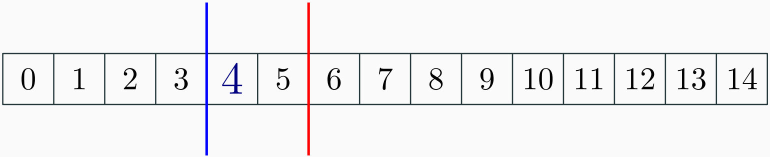

Abbildung: Paritäts-Orakel

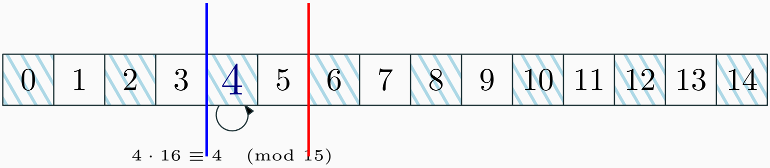

Abbildung: Paritäts-Orakel

PARITÄTS-ORAKEL

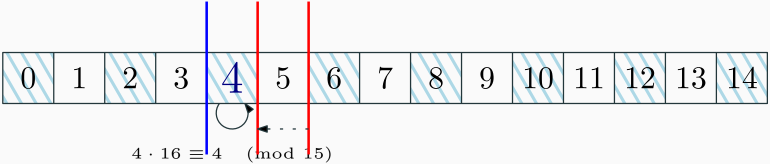

Abbildung: Paritäts-Orakel

PARITÄTS-ORAKEL

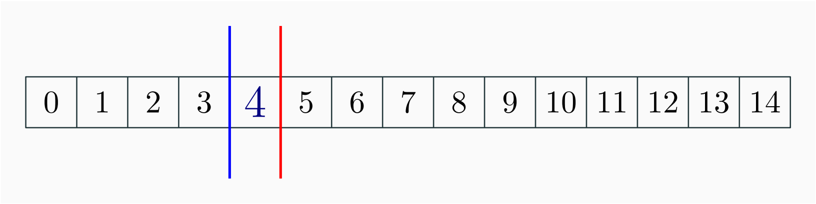

Abbildung: Paritäts-Orakel

DEMONSTRATION

ZUSAMMENFASSUNG PARITÄTS-ORAKEL

1. Nutzt die Homomorphie-Eigenschaften von RSA
2. Verkleinert sukzessive mit Anpassung des Ciphertextes das Intervall für m_0
3. Gutes Beispiel für eine einfache "Adaptive chosen-ciphertext attack".

Aus der Unsicherheit eines Bit folgt, dass der gesamte Klartext unsicher ist.

Bleichenbacher's Attacke

- Daniel Bleichenbacher (Bell Laboratories) veröffentlichte 1998 einen Angriff, der schnell als der “Million Question“-Angriff bekannt wurde.
- Ermöglicht es, einen mit PKCS#1 V1.5 verschlüsselten Cyphertext zu entschlüsseln
  $\Rightarrow$ Premaster-Secret
- Klassische Seitenkanal-Attacke
- Angreifbar sind RSA-basierende TLS cipher suites
- ECC und Diffie Hellman suites sind nicht angreifbar

1. Ähnliche Vorgehensweise wie beim Paritätsorakel, aber deutlich schwieriger auszunutzen
2. Funktioniert, wenn die Implementierung Informationen über ein korrektes oder inkorrektes Padding (PKCS#1 v1.5) leaked
3. Nutzt die Tatsache aus, dass das Padding mit `0x00 0x02` anfangen muss. Ist dies der Fall, kann das Intervall verkleinert werden

Gegeben sei ein Orakel, welches eine verschlüsselte Nachricht c entgegen nimmt. Das Orakel gibt `True` zurück, wenn die entschlüsselte Nachricht PKCS#1 v1.5 konform ist, sonst `False`.

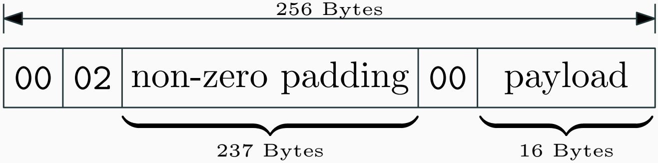

Abbildung: PKCS#1 v1.5 Encryption Format

RSA Modulus

Gegeben sei ein 128 Bit langer RSA-Modulus.

Verschlüsselt werden Octetstrings der Länge k Byte.

Bei einem 128-Bit Modulus sind die zu verschlüsselnden Octetstrings demnach 16 Byte lang.

AB	03	0D	F1	44	12	…

PKCS#1 v1.5 Padding Intervall

Welche Octetstrings fangen mit `0x00 0x02` an?

$$
\begin{array}{lccc}
k = 4 & \boxed{00}\boxed{02}\boxed{00}\boxed{00} & \cdots & \boxed{00}\boxed{02}\boxed{\mathrm{FF}}\boxed{\mathrm{FF}} \\
k = 6 & \boxed{00}\boxed{02}\boxed{00}\boxed{00}\boxed{00}\boxed{00} & \cdots & \boxed{00}\boxed{02}\boxed{\mathrm{FF}}\boxed{\mathrm{FF}}\boxed{\mathrm{FF}}\boxed{\mathrm{FF}} \\
 & \vdots & & \vdots
\end{array}
$$

Mit $B = 2^{8(k-2)}$ liegen alle Octetstrings, welche mit `0x0002` anfangen im Intervall

$$M = [2B, 3B - 1]$$

Beispiel für $k = 4$:

$$[2B, 3B - 1] = [2 \cdot 2^{8(k-2)}, 3 \cdot 2^{8(k-2)}] = [2^{17}, 2^{17} + 2^{16} - 1]$$

Abbildung: Bleichenbacher-Orakel – siehe dazu auch
`https://vimeo.com/116972878`

Abbildung: Bleichenbacher-Orakel – siehe dazu auch `https://vimeo.com/116972878`

Abbildung: Bleichenbacher-Orakel – siehe dazu auch `https://vimeo.com/116972878`

Abbildung: Bleichenbacher-Orakel – siehe dazu auch `https://vimeo.com/116972878`

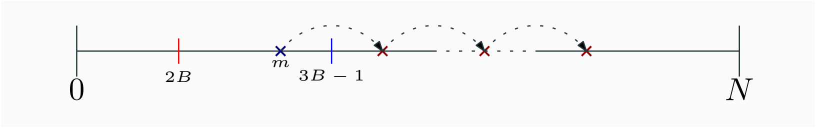

Abbildung: Bleichenbacher-Orakel – siehe dazu auch `https://vimeo.com/116972878`

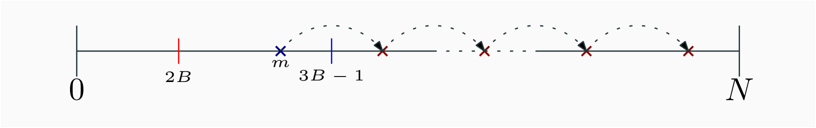

Abbildung: Bleichenbacher-Orakel – siehe dazu auch
`https://vimeo.com/116972878`

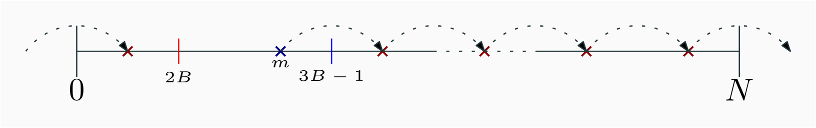

Abbildung: Bleichenbacher-Orakel – siehe dazu auch `https://vimeo.com/116972878`

Abbildung: Bleichenbacher-Orakel – siehe dazu auch `https://vimeo.com/116972878`

BLEICHENBACHER SCHRITT 1

> Jeder PKCS#1 v1.5 konforme Klartext liegt im Intervall $[2B, 3B-1]$.

Wir nutzen nun die Homomorphie-Eigenschaft von RSA und berechnen einen neuen Ciphertext:

Gegeben sei ein PKCS#1 v1.5 konformer Chiffretext c; gesucht ist:

$$m = c^d \bmod n$$

Berechne $ms_1 \bmod n$ mit verschiedenen Werten für $s_1 \geq \frac{n}{3B}$,[3] sodass das Orakel einen nach PKCS# v1.5 konformen Klartext entschlüsselt.

Da das Format stimmt, folgern wir:

$$ms_1 \in [2B, 3B-1]$$

[3] Herleitung gerne im Anschluss

Für das ausgewählte s_1 gilt: $ms_1 \in [2B, 3B-1]$

$$\begin{array}{rrcll} & 2B & \leq & ms_1 - rn & \leq & 3B-1 \\ \Leftrightarrow & 2B - ms_1 & \leq & -rn & \leq & 3B-1-ms_1 \\ \Leftrightarrow & ms_1 - 3B + 1 & \leq & rn & \leq & ms_1 - 2B \\ \Leftrightarrow & \frac{ms_1-3B+1}{n} & \leq & r & \leq & \frac{ms_1-2B}{n} \end{array}$$

Da $2B \leq m \leq 3B-1$ folgt:

$$R_i = \frac{2Bs_1 - 3B + 1}{n} \leq r \leq \frac{(3B-1)s_1 - 2B}{n}$$

Wir erhalten eine Lösungsmenge für r, die alle möglichen ganzen Zahlen r enthält.

Aus der Beschränkung von $ms_1 \in [2B, 3B-1]$ ergibt sich eine neue Beschränkung für m.

- Intuitiv gibt jedes r an, wie oft wir “zurück springen” müssen, um wieder im Intervall $[0, n-1]$ zu landen.
- Für jede Zahl r aus der Lösungsmenge können wir analog ein mögliches Intervall bestimmen:

$$\frac{2B + rn}{s_1} \leq m \leq \frac{3B - 1 + rn}{s_1}$$

Wir erhalten eine Menge von Intervallen in denen m liegen könnte.

Da m sowohl im Intervall $[2B, 3B-1]$, als auch in einem der Intervalle aus $[\frac{2B+rn}{s_1}, \frac{3B-1+rn}{s_1}]$ liegen muss, liegt m in der Schnittmenge dieser Intervalle.

$$M_1 = \bigcup_r \left\{ \left[\max\left(2B, \frac{2B+rn}{s_1}\right), \min\left(3B-1, \frac{3B-1+rn}{s_1}\right) \right] \right\}$$

M_1 besteht nun aus kleineren Teilintervallen von $[2B, 3B-1]$, wodurch die möglichen Werte für m drastisch reduziert werden können.

- Suche die kleinste Zahl $s_2 > s_1$, sodass $c \cdot (s_i)^e$ mod n PKCS-konform ist.
- Analog zu Schritt 1 erhält man eine neue Lösungsmenge für r'.
- Ebenso die neuen möglichen Intervalle für m:

$$\frac{2B + r'n}{s_1} \leq m \leq \frac{3B - 1 + r'n}{s_1}$$

- Nun wird jedes Intervall aus der Menge M_i mit jedem der neu berechneten Intervalle geschnitten. Ist diese Schnittmenge nicht leer, so wird dieses Schnittintervall zur Menge $M_i + 1$ hinzugefügt.

Fall 1: Die neue berechnete Menge an Intervallen M_i enthält mehrere Intervalle:

- Wiederhole Schritt 2, bis nur noch ein Intervall übrig ist.

Fall 2: Enthält M_i genau ein Intervall $[a, b]$ mit Länge > 1:

- Wähle zwei kleine natürliche Zahlen r_i, s_i, sodass

$$r_i \geq 2\frac{bs_{i-1} - 2B}{n} \quad \text{und} \quad \frac{2B + r_i n}{b} \leq s_i < \frac{3B + r_i n}{a}$$

bis $c \cdot (s_i)^e \bmod n$ PKCS-konform ist.

Enthält M_i nur noch ein einziges Intervall der Länge 1, z.B. $M_i = [a, a]$, dann gilt:

$$m = a$$

DEMONSTRATION

ORACLE-STRENGTH

- Stärke beschreibt die Wahrscheinlichkeit, dass eine Nachricht als PKCS#1 konform bestätigt wird.
- $P = \frac{\#\text{conform}}{\#\text{total}}$
- Weniger Prüfungen => stärkeres Oracle
- Idealerweise wird nur auf 0x0002 am Anfang geprüft.

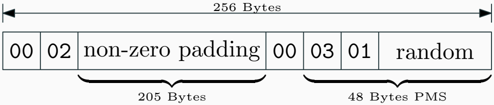

Abbildung: PKCS#1 Format für 2048-Bit Key[4]

[4] https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-meyer.pdf

GEGENMASSNAHMEN

BLEICHENBACHER GEGENMASSNAHMEN

- Alle Fehlermeldungen vereinheitlichen
- PKCS#1 v2.1 mit RSA-OAEP (wirklich sicher?!)[5]
- TLS 1.0: Falls die Nachricht nicht PKCS#1 konform ist, generiere zufälligen Key und fahre damit fort. Problem?
- TLS 1.2: Generiere immer zufälligen Key und nutze ihn, falls die Nachricht nicht PKCS#1 konform ist.

[5] A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS#1 v2.0, James Manger

- Alte TLS/SSL oder PKCS#1 Versionen können nicht einfach ausgeschlossen werden um Abwärtskompatibilität zu gewährleisten
- Sichere Implementierung ist wichtig, oftmal aber nicht der Fall

SCHWACHSTELLEN

BETROFFENE SOFTWARE (BEISPIELE)

- CVE-2001-0361
    - ssh-1 bis 1.2.31
    - OpenSSH bis 2.2.0
    - AppGate
- CVE-2006-4339, CVE-2012-0884
    - OpenSSL bis 1.0.0h
- CVE-2011-2487, CVE-2015-0226
    - Apache WSS4J bis 1.6.17 und 2.0.2
    - JBossWS
- CVE-2012-5081
    - Java Secure Socket Exentension (JSSE)

INTERNAL ERROR IN JSSE

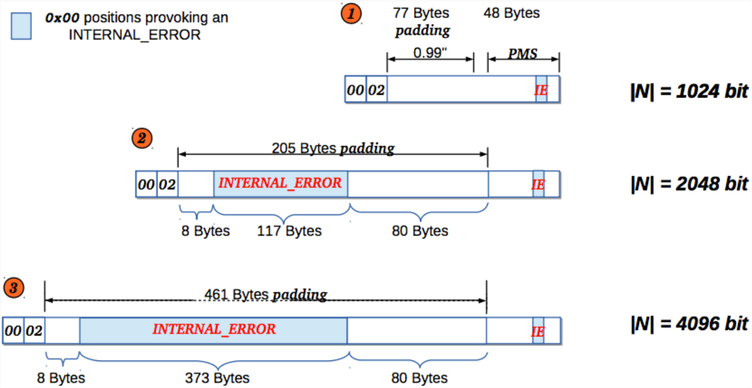

Abbildung: JSSE Internal Error `https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-meyer.pdf`

- Oracle Strength
  - 1024-bit Key: ~0.02%
  - 2048-bit Key: ~36%
  - 4096-bit Key: ~76%
- Performance
  - 1024-bit Key: hunderte Millionen Requests
  - 2048-bit Key: 176797 Requests, ca. 12 Stunden
  - 4096-bit Key: 73710 Requests, ca. 6 Stunden

```
<Envelope>
 <Header>
  <Security>
   <EncryptedKey Id="EncKeyId">
    <EncryptionMethod Algorithm="...xmlenc#rsa-1_5"/>
    <KeyInfo>...</KeyInfo>
    <CipherData>
     <CipherValue>Y2bh...fPw==</CipherValue>
    </CipherData>
    <ReferenceList>
     <DataReference URI="#EncDataId-2"/>
    </ReferenceList>
   </EncryptedKey>
  </Security>
 </Header>
 <Body>
  <EncryptedData Id="EncDataId-2">
   <EncryptionMethod Algorithm="...xmlenc#aes128-cbc"/>
   <CipherData>
    <CipherValue>3bP...Zx0=</CipherValue>
   </CipherData>
  </EncryptedData>
 </Body>
</Envelope>
```

c_{key}

c_{data}

Abbildung: Beispiel einer SOAP-Nachricht [6]

[6] https://www.nds.ruhr-uni-bochum.de/media/nds/veroeffentlichungen/2012/12/19/XMLencBleichenbacher.pdf

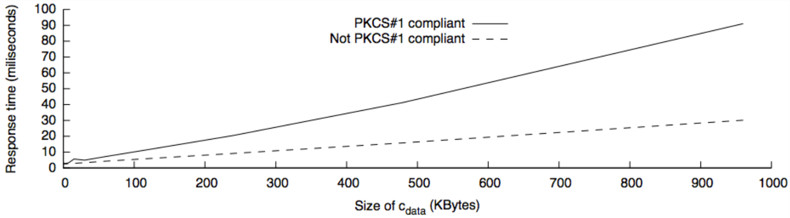

Abbildung: Zeitdifferenz zwischen validem und nicht validem Key in Raltion zur Ciphertext Länge
`https://www.nds.ruhr-uni-bochum.de/media/nds/veroeffentlichungen/2012/12/19/XMLencBleichenbacher.pdf`

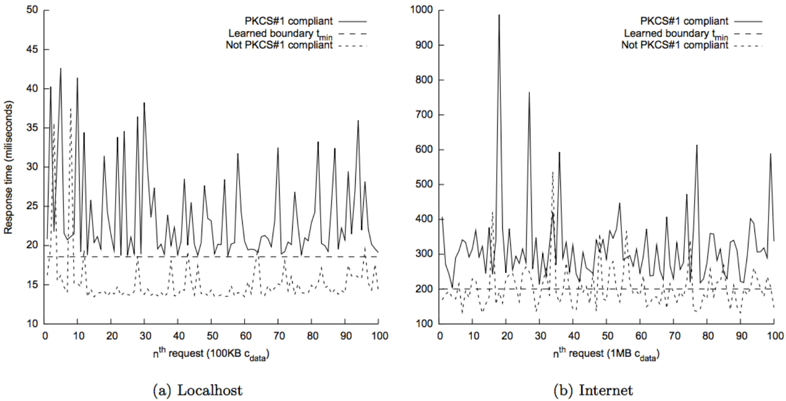

Abbildung: Reaktions Zeit des Servers für valide und nicht valide Keys
`https://www.nds.ruhr-uni-bochum.de/media/nds/veroeffentlichungen/2012/12/19/XMLencBleichenbacher.pdf`

- OpenSSL
  - Erzeugte nur den Random Key, wenn die Nachricht nicht PKCS#1 konform war
  - ~1.5 Mikrosekunden Unterschied
  - $P = 2.7 \cdot 10^{-8} \Rightarrow$ ~2^{40} Requests
- JSSE, OpenJDK
  - Schön Objektorientiert mit Exceptions
  - ~20 Mikrosekunden Unterschied
  - $P_{2048\text{bit}} = 60\% \Rightarrow 18600$ Requests und 19.6 Stunden
- Cavium, TLS Beschleunigungs-Hardware
  - Prüft nur 0x??02 $\Rightarrow$ schwächt das Oracle
  - Anpassen des Bleichenbacher Angriffs nötig
  - 4000000 Requests benötigt und 41 Stunden

- Zeitbedarf des Algorithmus' sollte kein Rückschluss auf den Ablauf geben
- Random sleep(t_{rand})
  - Erhöht nur das Rauschen, beseitigt nicht das Problem
- Statisches sleep($t_{max} - t_{used}$)
  - Verlangsamt den Prozess stark, aber keine Unterscheidung mehr erkennbar
- Statisches sleep($t_i - t_{used}$) mit Zeitintervallen
  - wenn $t_{used} < t_1 \Rightarrow$ sleep($t_1 - t_{used}$)
  - wenn $t_{used} < t_2 \Rightarrow$ sleep($t_2 - t_{used}$)
  - ...
- Deterministisches und unvorhersagbares Delay

FAZIT

- Der Bleichenbacher Angriff kann sehr mächtig sein und leicht ausgenutzt werden
- Über 15 Jahre alt und trotzdem immer wieder Thema
- Der interne Ablauf der Algorithmen sollte nach Außen nicht sichtbar werden, um Side-Channel-Attacks zu verhindern
- Sicherheit kommt vor Performance
- Encrypt-then-MAC

Fragen?

- Daniel Bleichenbacher. “Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1”. In: Advances in Cryptology—CRYPTO’98. Springer. 1998, S. 1–12.
- Jörg Schwenk. “Sicherheit und Kryptographie im Internet”. In: Auflage, Friedr. Vieweg & Sohn Verlagsgesellschaft, Braunschweig/Wiesbaden (2002).
- Christopher Meyer u. a. “Revisiting ssl/tls implementations: New bleichenbacher side channels and attacks”. In: 23rd USENIX Security Symposium (USENIX Security 14). USENIX Association. 2014, S. 17.

- Riccardo Focardi. Practical Padding Oracle Attacks on RSA. EN. URL: https://secgroup.dais.unive.it/wp-content/uploads/2012/11/Practical-Padding-Oracle-Attacks-on-RSA.html (besucht am 28. 06. 2015).
- B. Kaliski. PKCS#1: RSA Encryption Version 1.5. EN. URL: http://tools.ietf.org/html/rfc2313 (besucht am 28. 06. 2015).