Seminar IT Sicherheit
FH Wedel
Sommersemster 2016

Cryptoanalytic Attacks on RSA: Angriffe auf den öffentlichen Exponenten

Nils R. H. van Kan

Eingereicht bei:
Prof. Dr. Gerd Beuster
am
26.06.2016

Inhaltsverzeichnis:

1. Einleitung

Dieses Seminar basiert auf Kapitel 7 aus dem Buch „Cryptanalytic Attacks on RSA“ von Song Y. Yan. [1]

Es wird erläutert warum bei RSA Verschlüsselung in der Parxis häufig kleine öffentliche Exponenten (Schlüssel) verwendet werden und wie sich dadurch das Verschlüsselungsverfahren angreifen lässt. Als Grundlage dient eine Betrachtung des Coppersmith Theorem, gefolgt von drei möglichen Angriffen auf RSA. Alle diese Angriffe beruhen auf öffentlichen Information und sollen den Klartext einer verschlüsselten Nachricht in Polynominalzeit ermitteln.
Als Abschluss folgt eine kurze Betrachtung wie sich diese Angriffe auf die Zukunft von RSA auswirken und welche Schritte nötig sind um weiterhin eine sichere Verschlüsselung zu gewährleisten.

2. Grundlage

2.1 Kleiner öffentlicher Exponent

Das RSA Verfahren verwendet zum ver- bzw. entschlüsseln drei Zahle (e, d, N). Dabei bilden e und N den öffentlichen Schlüssel, welcher nach außen bekannt ist, d bleibt geheim und ist nur dem Schlüsselinhaber bekannt.
Nachrichten werden bei RSA wie folgt verschlüsselt:

$$C \equiv M^e \bmod N$$

Entschlüsselt werden Nachrichten dabei so:

$$M \equiv C^d \bmod N$$

In der Theorie sollten alle Teile des Schlüssels möglichst groß gewählt werden um die Sicherheit des Algorithmus zu erhöhen. In der Praxis wird RSA auf Computern ausgeführt deren Rechenleistung und Speicher begrenzt sind. Daher muss der Algorithmus möglichst effizient ausgeführt werden.
Besonders das Verschlüsseln einer Nachricht sollte schnell erfolgen um eine reibungslose Kommunikation zu ermöglichen. Der größte Rechenaufwand ergibt sich herbei aus der Exponentialfunktion M^e. Ein großer Exponent e führt schnell zu großem Rechenaufwand und somit zu langer Laufzeit des Algorithmus. Zum Vergleich: Ein gebräuchlicher Taschenrechner liefert schon bei einem Exponenten größer 400 einen Fehler.
Daher werden bei RSA häufig e = 3, e = 17 oder e = 65537 gewählt. [1] Um dies genauer zu erklären werden die drei Zahlen zunächst wie folgt zerlegt:

$$3 = 2 + 1 = 2^1 + 1$$
$$17 = 16 + 1 = 2^4 + 1$$
$$65537 = 65536 + 1 = 2^{16} + 1$$

Eine Exponentialfunktion kann wie folgt dargestellt werden:

$$M^x = M \cdot M \cdot M \cdot \ldots$$

Eine Multiplikation ist auf einen Computer eine vergleichsweise teure Operation, sie benötigt relativ viel Rechenzeit. Das Quadrieren einer Zahl ist hingegen relativ günstig und kann schneller durchgeführt werden. Betrachtet man nun e = 3 und 3 = 17 sowie deren Zerlegung oben und bezieht die Regeln für Exponentialrechnung mit ein ergibt sich:

$$M^3 = M^2 \cdot M$$
$$M^{17} = (((M^2)^2)^2)^2 \cdot M$$

Für die Berechnung ist somit nur eine Multiplikation sowie eine Reihe von Quadrierungen nötig. Damit ist die Verschlüsselung bei e = 17 schneller als zum Beispiel bei e = 15, da hier nicht so effizient zerlegt werden kann:

$$15 = 8 + 7 = 2^3 + 7$$
$$M^{15} = (((M^2)^2)^2) \cdot M \cdot M \cdot M \cdot M \cdot M \cdot M \cdot M$$

Dieser Umstand und die Tatsache, dass die Berechnung des privaten Exponenten d bei einem festgelegten öffentlichen Exponenten e schneller durchgeführt werden kann, führt in der Praxis häufig zu der Verwendung der drei oben genannten Zahlen. [1]

2.2 Mathematische Grundlage

2.2.1 Coppersmith Theorem

Das Coppersmith Theorem wurde 1997 von dem Mathematiker Don Coppersmith (*1950) aufgestellt.[3] Das Theorem beschreibt eine Methode zur effizienten Bestimmung der Nullstellen eines Polynoms modulo N innerhalb einer festgelegten Grenze B. Es bildet dabei die Grundlage für die meisten und wirkungsvollsten Angriffe auf den öffentlichen Exponenten e. [1]
Ausgehend von einem Polynom n-ten Grades

$$p(x)=x^n+a_{n-1}\cdot x^{n-1}+\ldots+a_1\cdot x+a_0$$

werden alle x <= B gesucht die ergeben:

$$p(x)\equiv 0 \, mod \, N$$

Zur Lösung wird der LLL-Algorithmus eingesetzt, der im nächsten Unterkapitel kurz beschrieben wird. Einfach ausgedrückt wird die schwer zu lösende Funktion

$$p(x)\equiv 0 \, mod \, N$$

durch die relativ einfach zu lösende Funktion

$$r(x)=0$$

ersetzt. Nun muss nur noch die Nullstelle der Funktion ermittelt werden, beispielsweise mit Hilfe der Newton Methode.

2.2.2 LLL-Algorithmus

Dieser Abschnitt ist kurz gehalten, da eine genau Betrachtung des Algorithmus über das Thema des Seminars hinausgehen würde.
Der LLL-Algorithmus wurde 1982 von Arjen Lenstra, Hendrik Lenstra und Lázlo Lovász entwickelt. [4] Er berechnet für eine Matrix eine Basis mit minimalen Vektoren. Er war das erste effiziente Gitterreduktionsverfahren und findet vielfältige Anwendung.

Der Algorithmus geht dabei in einer Matrix spaltenweise vor. Er reduziert zunächst die erste Spalte als Initialwert. Im Anschluss wird immer die nächste Spalte reduziert und dann mit der vorangegangenen Spalte verglichen, ob sie gegenüber dieser minimal ist. Trifft dies zu geht der Algorithmus zur nächsten Spalte über, trifft dies nicht zu werden die beiden verglichenen Spalten vertauscht und erneut reduziert. Auf diese Weise kann eine Matrix in Polynominalzeit reduziert werden.

2.2.3 Beispiel

Dieses Beispiel bezieht sich auf die Howgrave-Graham Methode, die kurz nach dem Coppersmith Theorem veröffentlicht wurde, mit diesem aber größtenteils übereinstimmt. Die Howgrave-Graham Methode ist dabei allerdings etwas effizienter. [1]
Grundlage des Beispiels ist das Polynom

$$p(x)=x^2+14x+19\equiv 0(mod\,35)$$

Zunächst wird die 6 x 6 Matrix M gebildet. (Bei Coppersmith wäre eine 10 x 10 Matrix nötig)

```
      1225
      0    1225*2
      665      490*2    35*2^2
M =     0    665*2    490*2^2  35*2^3
      361      532*2    234*2^2  28*2^32^4
      0    361*2    532*2^2  234*2^3    28*2^42^5
```

Mit dem LLL-Algorithmus wird M zu B reduziert:

```
      3    8*2       -24*2^2    -8*2^3       -1*2^4       2*2^5
      49   50*2      0          20*2^30      2*2^5
      115-83*2    4*2^2        13*2^36*2^4        2*2^5
B =61   16*2      37*2^2     -16*2^3     3*2^4        4*2^5
      21   -37*2    -14*2^2    2*2^3       14*2^4-4*2^5
      -201     4*2       33*2^2     -4*2^3       -3*2^4       1*2^5
```

Jede Zeile der Matrix kann wieder als Polynom dargestellt werden, in diesem Beispiel die erste Zeile:

$$r(x)=2x^5-x^4-8x^3-24x^2+8x+3$$

Löst man dieses Polynom ergibt sich:

$$x_0=3$$

3. Angriffe bei gleichen Nachrichten

3.1 Theorie

Wenn der öffentliche Exponent e klein gewählt wird und die selbe Nachricht M an mehrere Empfänger gesendet wird kann M relativ einfach ermittelt werden, wenn die genutzten N jeweils teilerfremd sind.
Die selbe Nachricht M wird an drei Empfänger gesendet mit e = 3.

$$C_1 \equiv M^3 (mod\, N_1)$$
$$C_2 \equiv M^3 (mod\, N_2)$$
$$C_3 \equiv M^3 (mod\, N_3)$$

Dies lässt sich umstellen zu:

$$M^3 \equiv C_1 (mod\, N_1)$$
$$M^3 \equiv C_2 (mod\, N_2)$$
$$M^3 \equiv C_3 (mod\, N_3)$$

Mit Hilfe des Chinesischen Restsatzes ergibt sich

$$M \equiv C_1 \cdot (N_2 \cdot N_3) \cdot ((N_2 N_3)^{-1} mod\, N_1) +$$
$$C_2 \cdot (N_1 \cdot N_3) \cdot ((N_1 N_3)^{-1} mod\, N_2) +$$
$$C_3 \cdot (N_1 \cdot N_2) \cdot ((N_1 N_2)^{-1} mod\, N_3) (mod\, N_1 \cdot N_2 \cdot N_3)$$

Nach dem Ziehen der Kubikwurzel ist die Nachricht M wieder hergestellt. Dafür musste weder N faktorisiert werden, noch musste d oder andere Informationen der Falltürfunktion bekannt sein.[1]

Unsicher wird RSA durch diesen Angriff allerdings nicht, da viele Voraussetzungen gleichzeitig erfüllt sein müssen. Zunächst ist es ungewöhnlich exakt die selbe Nachricht an mehrere Empfänger zu senden, was immer potentiell unsicher ist, da ein Angreifer so leichter an Informationen gelangen kann.
Des weiteren müssen alle N teilerfremd sein. Die Überprüfung dieses Faktors ist relativ aufwendig, vor Allem bei großen N und/oder bei einer großen Anzahl N. Außerdem hat der Angreifer im Allgemeinen keinen Einfluss auf die Wahl der N, sodass er die Teilerfremdheit auch nicht erzwingen kann.
Zuletzt muss die Anzahl der gesendeten Nachrichten >= e sein. Somit steigt bei einem größeren öffentlichen Exponenten auch die Sicherheit stark an. Spätestens bei e = 65537 ist es äußerst unwahrscheinlich, dass die selbe Nachricht M an 65537 Empfänger gesendet wird.[1]

3.2 Beispiel

e = 3

N_1 =
160742267780887390193498154036432277706853678172635349431669799546987155543930704061491272423912871491087467233437624891628418 29

N_2 =
114381625757888867669235779976146612010218296721242362562561842935706935245733897830597123563958705058989907514759929002687954354 1

N_3 =
180708208868740480595165616440590556627810251676940134917012702145005666254024404838734112759081230337178188796656318201321488055 7

M =
200805001301070903002315180419000118050019172105011309190800151919090618010170 5

Zunächst werden C1, C2 und C3 berechnet und anschließend mit Hilfe des Chinesischen Restsatzes M^3 ermittelt.

M^3 =
809698949404799825175318829599280800701404889564329888804732359014720253220847686394738491562155963332021601075233618284262592613313570410096477344103527394431017424522039758939871510237521097572264320228702391835756584586111115262 5

Nach dem Ziehen der Kubikwurzel entsteht wieder

M =
200805001301070903002315180419000118050019172105011309190800151919090618010170 5

4. Angriffe bei zusammenhängenden Nachrichten

4.1 Theorie

Sollten zwei oder mehr Nachrichten mit dem selben e und N verschlüsselt werden ist es möglich die originale Nachricht wieder zu ermitteln wenn die beiden Nachrichten M_1 und M_2 zusammenhängen:

$$M_2 = aM_1 + b$$

Die RSA Verschlüsselung ergibt:

$$C_1 \equiv M_1^3 (mod\ N)$$
$$C_2 \equiv (aM_1 + b)^3 (mod\ N)$$

Daraus folgt:

$$M_1 \equiv \frac{b(C_2 + 2a^3 C_1 - b^3)}{a(C_2 - a^3 C_1 + 2b^3)}$$
$$M_1 \equiv \frac{3a^3 bM_1^3 + 3a^2 b^2 M_1^2 + 3ab^3 M_1}{3a^3 bM_1^2 + 3a^2 b^2 M_1 + 3ab^3} (mod\ N)$$

Sind dem Angreifer e, N, C_1 und C_2 bekannt müssen die Werte nur noch eingesetzt werden um M zu erhalten.
Im Gegensatz zu dem Angriff im vorherigen Kapitel ist die hier beschriebene Methode schon deutlich praktikabler. Wird davon ausgegangen, dass mehrmals mit dem selben Empfänger kommuniziert wird ist es plausible, das mehrere Nachrichten mit dem gleichen Schlüssel (e,N) verschlüsselt werden. Mehrmals die selbe Nachricht zu senden ist potentiell unsicher, daher muss der Sender ein Paddingverfahren nutzen, nicht nur wenn er mehrfach die selbe Information verschlüsselt, sondern allgemein beim Einsatz von RSA. Ein einfaches Paddingverfahren entspricht dabei genau dem hier beschriebenen Muster:

$$M_2 = aM_1 + b$$

Allerdings ist auch hier anzumerken, dass es ungewöhnlich ist mehrmals die exakt selbe Nachricht zu verschicken und es ansonsten reiner Zufall wäre, wenn zwei Nachrichten nach diesem Muster zusammenhängen.[1]

4.2 Beispiel

$$
\begin{aligned}
\alpha &= 3 \\
\beta &= 5 \\
N &= 77903022885101595423624756547055783624857676209739839410844022221357287251170999858504838764813194434051093226513681516857411993477558685427409422564450008791272325857493370618539583402784340582088810854850787372421498307209138098905430816127795973382486506868759421313982662205554370007455229369350394035118720326674091105680617088067997846221222823129257533392400699736209686208403812308104374465892532943045181265208185871222090592859132787427488883517622574112296645299299833541045392916173339389220473000267483895528757240045575727185403212 (\mod N)
\end{aligned}
$$

Then

$$
\begin{aligned}
M_1 &\equiv \frac{5(C_2 + 2 \cdot 3^3 \cdot C_1 - 5^3)}{3(C_2 - 3^3 \cdot C_1 + 2 \cdot 5^3)} \\
&\equiv 200805001301070903002315180419000118050019172105011309190800151919090618010705 \ (\mod N) \\
M_2 &\equiv 3M_1 + 5 \ (\mod N) \\
&\equiv 602415003903212709006945541257000354150057516315033927572400455757271854032120 \ (\mod N)
\end{aligned}
$$

5. Angriffe auf stereotype Nachrichten

5.1 Theorie

Eine stereotype Nachricht M besteht aus einem bekannten Teil B und einem unbekannten Teil x.

$$M = B + x$$

Ein Beispiel wäre eine Bank die einem Kunden eine neue Pin zuschicken möchte. Die Bank sendet die Nachricht „Ihre neue Pin ist ****“. Dabei bildet „Ihre neue Pin ist“ den bekannten Teil B und „****“ die unbekannte Information, die der Angreifer ermitteln möchte.

Mit der RSA Verschlüsselung ergibt sich bei e=3:

$$C \equiv M^3 \pmod N$$
$$C \equiv (B + x)^3 \pmod N$$

Dies lässt sich nach 0 umstellen:

$$(B + x)^3 - C \equiv 0 \pmod N$$

Mit Hilfe des oben beschriebenen Coppersmith Theorem lässt sich eine Lösung finden:

$$x_0 \equiv 0 \pmod N$$
$$x_0 = (B + x)^3 - C$$

Da alle Variablen außer x bekannt sind lässt sich x und somit die geheime Information einfach ermitteln.
Im Gegensatz zu den eher theoretischen Anwendungsmöglichkeiten der beiden zuvor vorgestellten Angriffe bietet diese Methode ein größeres Potential für praktische Anwendungen. Es werden nicht mehr mehrere gleiche oder zusammenhängende Nachrichten benötigt um die Information zu erhalten. Außerdem sind stereotype Nachrichten nach dem Muster M = B + x denkbar, auch wenn aufmerksamen RSA Nutzern klar sein sollte, dass das Versenden ähnlicher Nachrichten ohne Padding oder anderer Verfahren immer unsicher ist und deshalb niemals geschenhen sollte. Und wie schon bei den anderen Angriffen ist es auch hier weder nötig N zu faktorisieren noch Informationen der Falltürfunktion zu ermitteln.[1]

Die Bank dient hier natürlich nur als Beispiel und weicht stark von der Realität ab. Eine echte Bank würde niemals eine Pin auf die Weise verschicken und bei der Übermittlung sensibler Daten immer ein Paddingverfahren verwenden.

5.2: Beispiel

N = 549574648413583142768645428985 51

Wie in der Theorie beschrieben gehen wir von einer Nachricht nach dem Muster M = B + x aus, wobei dem Angreifer B bekannt ist, x allerdings nicht.

B = 25152118001609140014150009190000

Zusätzlich ist dem Angreifer bekannt, das x zwischen 0 und 10000 liegt.

C = 373933230960876657639221068571 01

Um x zu ermitteln wird zunächst ein Polynom aufgestellt:

$$f(x) \equiv (B+x)^3$$
$$f(x) \equiv B^3 + 3B^2 x + 3Bx^2 + x^3 - C$$
$$f(x) \equiv x^3 + 3Bx^2 + 3B^2 x + (B^3 - C)$$
$$f(x) \equiv x^3 + a_2 x^2 + a_1 x + a_0 (mod\ N)$$

Wobei:

$$a_2 \equiv 3B (mod\ N)$$
$$a_2 \equiv 3B^2 (mod\ N)$$
$$a_2 \equiv B^3 - C (mod\ N)$$

Es wird ein y festgelegt, in diesem Beispiel y = 10000 und eine Matrix gebildet:

$$v_1 = (N, 0, 0, 0)$$
$$v_2 = (0, yN, 0, 0)$$
$$v_3 = (0, 0, y^2 N, 0)$$
$$v_4 = (a_0, ya_1, y^2 a_2, y^4)$$

Anschließend wird der LLL-Algorithmus verwendet um die Matrix zu reduzieren. Jede Zeile der reduzierten Matrix kann nun in ein Polynom der Form

$$r(x) = e_3 x^3 + e_2 x^2 + e_1 x + e_0$$

eingesetzt werden.
Wird r(x) = 0 gesetzt und das Polynom so gelöst ergibt sich als Lösung x = 1379.

6. Ausblick

„Based on recent results in this area the public exponent for RSA must be sufficiently large. Values such as 3 or 17 can no longer be recommended, but commonly used values such as […] 65537 still seem to be fine. If one prefers to stay on the safe side one may select an odd 32-bit or 64-bit public exponent at random.“

Lenstra, Verheul [5]

Kleine öffentliche Exponenten e sind potentiell unsicher, da sie Raum für Angriffe bieten. Daher wird empfohlen e = 3 und e = 17 nicht mehr zu nutzen sondern größere Exponenten zu verwenden. Also sollte e immer so groß wie Möglich gewählt werden. Allerdings darf auch nicht vernachlässigt werden, dass RSA um effizient zu bleiben keine zu großen Schlüssel verwenden kann. Es ist ein schmaler Grad zwischen Sicherheit und Effizienz. Da davon auszugehen ist, dass die Rechenleistung der Angreifer, aber auch die der Nutzer, stetig steigt, sollte RSA beobachtet werden und zu gegebener Zeit eine Empfehlung zur Nutzung größerer Schlüssel herausgegeben werden. Allerdings darf RSA niemals ohne Paddingverfahren verwendet werden. Durch Padding wiederum wird aber auch die Sicherheit bei kleinen Exponenten deutlich erhöht, da viele Angriffe sehr viel schwerer oder gar unmöglich werden. Somit kann argumentiert werden das auch kleine Exponenten weiterhin sicher sind. Damit hängt die Sicherheit der Verschlüsselung aber nicht mehr nur von der Verlässlichkeit von RSA sondern auch von der des verwendeten Paddingverfahrens ab. Sollte eine einfache Methode zur Umgehung oder Zurückrechnung des Padding gefunden werden, könnten Angriffe bei der Verwendung kleiner öffentlicher Exponenten sehr viel einfacher gelingen als bei größeren Exponenten. Daher sollte neben RSA auch immer die Sicherheit des verwendeten Paddingverfahrens kritisch beobachtet werden und nach Möglichkeit ein größerer öffentlicher Exponent gewählt werden.
Zusätzlich sollten niemals gleiche Nachrichten oder Nachrichten mit teilweise bekannten Informationen zu versenden werden.

7. Quellen

[1] Song Y. Yan, Cryptanalytic Attacks on RSA, Springer Science+Business Media, 2008

[2]Durfee, G.: „Cryptanalysis of RSA using algebraic and lattice methods“, Juni 2002 unter: http://theory.stanford.edu/~gdurf/durfee-thesis-phd.pdf (abgerufen am 12.07.2016)

[3] Coppersmith, D.: The Data Encryption Standard (DES) and its strength against attacks, IBM J. Res. Dev., Band 38, 1994, S. 243

[4] Phong Q. Nguyen, Brigitte Valée (Hrsg.): The LLL algorithm. Survey and applications. Springer 2010, ISBN 978-3-642-02294-4.

[5] A. K. Lenstra, R. R. Verheul, „Selecting Cryptographic Key Sizes“, Journal of Cryptology, 14, 2001, p 265