Erhöhung der Sicherheit von Public-Key Implementierungen auf Smart Cards gegen Timing-Angriffe

Hagen Ploog, Andreas Ahrens und Dirk Timmermann

1 Einleitung

Public-Key-Kryptographie zeichnet sich u.a. dadurch aus, dass unterschiedliche Schlüssel zum Ver- und Entschlüsseln benutzt werden. Damit der private Schlüssel nicht in Erfahrung gebracht werden kann, wird bei der Implementierung in Smart Cards großer Aufwand betrieben. Statt den Algorithmus selbst zu attackieren, kann ein nicht-invasiver Angriff mit Hilfe der sogenannten Timing-Attacke durchgeführt werden. Der Angriff geht davon aus, dass es dem Angreifer möglich ist, das Verschlüsselungssystem während des Angriffs zu beobachten.

2 Modulo-Exponentierung

Die Modulo-Exponentierung (RSA) ist das am meisten verwendete Public-Key-Verfahren. Dabei wird eine Nachricht M unter Einbeziehung des öffentlichen Schlüssel E und des Moduls N zum verschlüsselten Text C; es ist $C = M^E \bmod N$. Beim Dekodieren wird anstatt des öffentlichen der private Schlüssel D benutzt; es ist $M = C^D \bmod N$. Da in beiden Fällen die gleiche mathematische Operation zu Grunde liegt, kann auch allgemeiner geschrieben werden $M = B^K \bmod N$. Die modulo Exponentierung lässt sich durch wiederholte Ausführung der modulo Multiplikationen ("square and multiply") darstellen:

```
Algorithmus 1:

Input  : B,K,N
Output : B^K mod N
1. Y=1
2. For i = n-1 DOWNTO 0 DO
3.   Y = Y·Y mod N
4.   Y = Y·B mod N IF(K_i==1)
5. END FOR
6. RETURN Y
```

3 Timing-Attacke

Die Berechnungszeiten zur Verschlüsselung der Eingangsdaten hängen von dem geheimen Schlüssel, also dem Ziel des Angreifers, ab. Besteht die Möglichkeit, diese Zeit zu messen, sind Rückschlüsse auf den Schlüssel möglich [1].
Dabei wird davon ausgegangen, dass die Berechnungszeit für die modulo Multiplikation annähernd konstant ist und nur um die Zeit einer evt. auszuführenden Korrekturaddition innerhalb der Modulo-Multiplikation variieren kann.

Diese Addition ist nötig, da das Ergebnis nach dem Reduktionsalgorithmus in Abhängigkeit des Verfahrens entweder im Bereich]-*N* ..*N*[bzw. [0 .. 2*N*[liegt.
Es sei *A* der Algorithmus, der auf eine Nachricht $m \in M$ mit einem Schlüssel $k \in K$ angewendet wird. Das Ergebnis sei die Signatur *S*. Die jeweilige Zeit, die *A*(*m*,*k*) benötigt, sei $t_i \in T$. Weiterhin sei *O*(*m*) das Orakel, das unter gewissen Annahmen bezüglich des Aufbaus des Kryptosystems Aufschlüsse über die Berechnung von *A*(*m*,*k*) zulässt. Es ist:

$$A : M \mathrm{x} K \mapsto S \quad : \quad (m, k,) \mapsto A(m, k)$$
$$B = \{0, 1\}$$
$$T : M \mapsto B \quad : \quad m \mapsto O(m)$$

4 Angriff auf die Quadrierung

In [1] und [2] wird der Multiplikations-Schritt von *Algorithmus 1* angegriffen. Wir attackieren stattdessen die Quadrierung. Für den Angriff auf Bit *i* des geheimen Schlüssels wird der Algorithmus *A*(*m*,*k*) in *L*(*m*,*k*) und *R*(*m*,*k*) aufgeteilt. Dabei stellt *L*(*m*,*k*) die bisherige Berechnung von *Algorithmus 1* bis zum Bit (*i*-1) und die Quadrierung (Zeile 3) für das Bit *i* dar. *R*(*m*,*k*) seien alle noch folgen Operationen. Die Zeit zur Berechung ist damit $T(m)=T^L(m)+T^R(m)$, wobei $T^L(m)$ die Zeit für die Berechnung von *L*(*m*,*k*) und $T^R(m)$ die Zeit für die Berechnung von *R*(*m*,*k*) ist.
Ist $k_i=1$ wird als nächstes die modulo Multiplikation und dann die Quadrierung ausgeführt. Die Aufteilung der Nachrichten in die Mengen M_1 und M_2 erfolgt abhängig davon, ob eine Korrekturaddition während der Quadrierung stattgefunden hat. Ist $k_i=0$ wird nur die Quadrierung ausgeführt. Die Nachrichten werden wiederum in Abhängigkeit von der Korrekturaddition in zwei Gruppen, M_3 und M_4, aufgeteilt. Nur eine dieser beiden Aufteilungen ist sinnvoll. Um nun auf k_i zu schließen werden die Differenzen in der Ausführungszeit zwischen M_1 und M_2 bzw. M_3 und M_4 bewertet.
Das zu diesem Zeitpunkt gültige *y* sei y_{temp}. Die Orakel lassen sich dadurch wie folgt definieren:

$$O_1(m) \mapsto \begin{cases} 1, \text{ mit Korrekturaddition in } \left(\mathrm{y}_{\mathrm{temp}} \cdot \mathrm{y}\right)^2 \\ 0, \text{ohne Korrekturaddition in } \left(\mathrm{y}_{\mathrm{temp}} \cdot \mathrm{y}\right)^2 \end{cases}$$

$$O_2(m) \mapsto \begin{cases} 1, \text{ mit Korrekturaddition in } \left(\mathrm{y} \cdot \mathrm{y}\right)^2 \\ 0, \text{ohne Korrekturaddition in } \left(\mathrm{y} \cdot \mathrm{y}\right)^2 \end{cases}$$

Damit sind die Mengen festgelegt:

$M_1=\{m \in M : O_1(m)=1\}$
$M_2=\{m \in M : O_1(m)=0\}$
$M_3=\{m \in M : O_2(m)=1\}$
$M_4=\{m \in M : O_2(m)=0\}$

$F_j=M_j \mapsto R : F_j(m)=T(m),\ 1 \le j \le 4$

Für k_i=1 gilt:

$$F_1(m) = T^L + T^R$$
$$F_2(m) = T^R$$
$$F_3(m) = F_4(m)$$

Also gilt für die arithmetischen Mittelwerte: $\overline{F_1} > \overline{F_2}$ und $\overline{F_3} = \overline{F_4}$. Ist k_i=0, dann gilt:

$$F_1(m) = F_2(m)$$
$$F_3(m) = T^L + T^R$$
$$F_4(m) = T^R$$

Und damit $\overline{F_1} = \overline{F_2}$ und $\overline{F_3} > \overline{F_4}$.

Durch die Attackierung der Quadrierung braucht nicht mehr unterschieden zu werden, ob eine Aufteilung sinnvoll ist, sondern welche der beiden Aufteilungen signifikanter ist. Abbildung 1 zeigt die Ergebnisse des Algorithmus' am Beispiel eines 16 Bit Schlüssels. Der geheime Schlüssel lässt sich direkt aus dem Diagramm ablesen. Bit 0 ist auf diese Weise nicht mehr bestimmbar, da nach der Multiplikation keine weitere Quadrierung mehr erfolgt. Da aber die Nachrichten *M* und die Signaturen *S* bekannt sind, kann eine zweite Signatur *S´* unter der Annahme das k_0=0 ist, erzeugt werden. Ist nun *S = S´* war die Annahme richtig und der Schlüssel liegt vor, ansonsten ist k_0=1. Ein robusteres Verhalten des Algorithmus kann u.a. dadurch erreicht werden, dass nicht mehr mit einem festen, sondern mit einem adaptiven Wert verglichen wird. Wird in einem Schritt *i* sowohl die Hypothese k_i=1 also auch k_i=0 abgelehnt, deutet dies auf einen Fehler in der Bestimmung vorangegangener Bits hin.

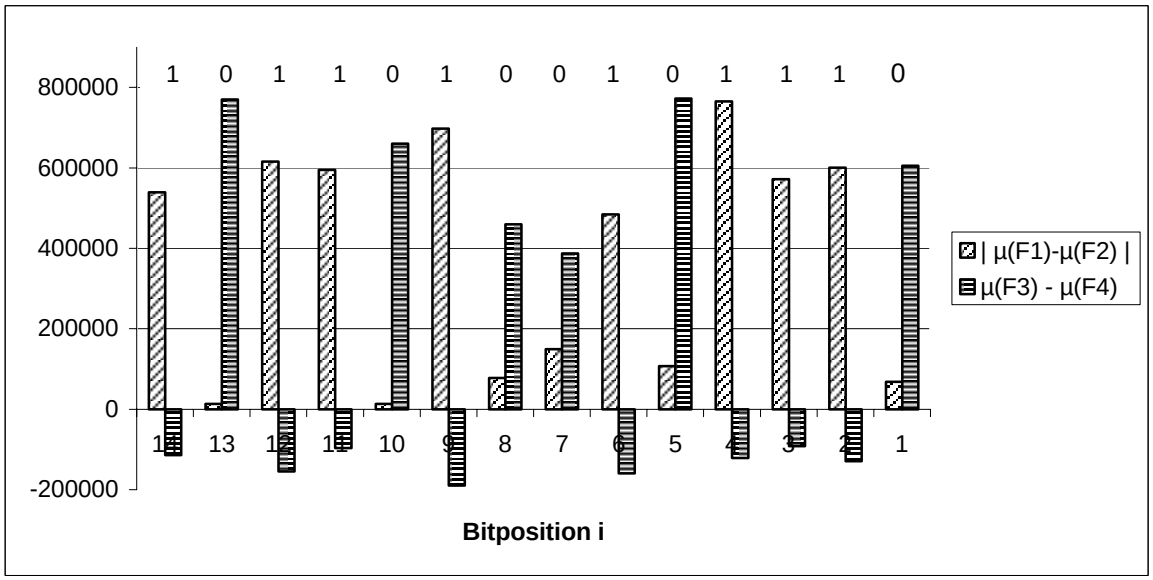

Abbildung 1 : Auswertung der Timing-Analyse für einen 16 bit Schlüssel (2000 Signaturen)

5 Gegenmaßnahmen

Häufig wird sogenanntes Blinding zum Verhindern der Timing-Analyse einsetzt [3]. Dabei wird ein zufälliges Zahlenpaar (v_i, v_f) mit $v_f^{-1} \equiv v_i^x \bmod N$ gewählt. Vor jeder modulo Exponentierung wird der Eingangswert mit v_i multipliziert. Um das Ergebnis zu korrigieren, wird es mit v_f multipliziert. Allerdings bietet auch diese Methode Möglichkeiten zur Timing-Analyse, weshalb das Zahlenpaar häufig neu berechnet werden sollte.
Unser Ansatz ergibt sich ebenfalls auf den Gesetzen der modularen Mathematik. Es gilt:

$$A \bmod N = (A + z \cdot N) \bmod N = (A + 2^x \cdot N) \bmod N$$

Es müssen lediglich Vielfache des Moduls N addiert werden. Leicht zu Implementieren sind Vielfache zur Basis 2, die sich durch Linksschieben des Moduls erzeugen lassen. Dadurch wird das Ergebnis nicht verändert. Die Zeit hingegen, die für die Signatur benötigt wird, variiert und lässt somit keine Schlussfolgerungen mehr auf den Schlüssel zu.
Um die Sicherheit weiter zu erhöhen, erscheint es sinnvoll, quasi-zufällig ausgewählte Vielfache zu addieren, was z.B. durch rückgekoppelte Schieberegister erfolgen kann.

6 Zusammenfassung

Wir haben in diesem Beitrag demonstriert, dass mit einem relativ einfachen Angriff der geheime Schlüssel in Erfahrung gebracht werden kann, ohne dabei die Hardware der Chipkarte zu zerstören. Bei der Implementierung von Public-Key-Kryptosystemen muss deshalb auch dieser Aspekt immer Berücksichtigung finden.
Wir haben weiterhin gezeigt, mit welchen Maßnahmen ein, zumindest vorläufig wirksamer, zusätzlicher Schutz realisiert werden kann.

Literatur

[1] P.C. Kocher, "Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems", Advances in Cryptology - CRYPTO '96, LNCS 1109, 1996, S. 104-113.
[2] Dehm et al, "A practical implementation of the timing attack", Crypto Group Technical Report Series CG--1998/1, Universit'e Catholique de Louvain
[3] T.S. Messerges, "Investigations of power analysis attacks on smartcards", USENIX Workshop on Smartcard Technology, 1999. CG--1999/1

Verfasser

Dipl.-Ing. Hagen Ploog
Universität Rostock
Fachbereich Elektrotechnik und Informationstechnik
Institut für Angewandte Mikroelektronik und Datentechnik
Richard-Wagner-Str. 31
18119 Rostock