

Diplomarbeit

# Gegenmaßnahmen zu Seitenkanalangriffen

Oktober 2007



von  
Berhane Yoseph  
ብርሃነ ዮሴፍ

Technische Universität Darmstadt  
Fachbereich Informatik  
Fachgebiet Theoretische Informatik  
Kryptographie und Computeralgebra  
Prof. Dr. Johannes A. Buchmann

Betreuer: Dipl.-Inform. Daniel Schepers

# Ehrenwörtliche Erklärung

Hiermit versichere ich, daß ich diese Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe. Diese Arbeit ist in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt worden. Alle Ausführungen, welche wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Darmstadt, den

Berhane Yoseph  
yoseph@web.de

## Danksagung

Ich möchte mich an dieser Stelle bei den Menschen bedanken, die mich, während der Erstellung dieser Arbeit unterstützt haben. Ich danke Herrn Prof. Dr. Johannes Buchmann für seine stets inspirierenden und unterhaltsamen Vorlesungen. Mein besonderer Dank gilt Daniel Schepers, dessen stets hervorragende Betreuung diese Arbeit erst ermöglicht hat.

Ebenso möchte ich mich bei allen, meinen ehemaligen Kommilitonen für eine interessante Studienzeit bedanken.

**ነደይ ወይዘሮ ለተንሴኤ ብርሃነ ኣቶ ኣንዶምኣለም ወልደገርጊሽ ኣብ ጊዜ ናይ ዩኒቨርሲቲ ትምህርቲ ብዓቕሊ ስለዘሰንዩኒን ናይ ትምህርቲ ዕድል ዝለ ዝከፈቱለይ እዛ መጽሓፍ ንመዘከርታ እውፍየሎም። ብዘይ ናቶም ሓገዝ ኣብዚ በጽሑዮ ዘለኹ ደርጃ ኣይ ምስበጸሕኩን።**

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Mathematische Grundlagen</b>	<b>2</b>
2.1	Elliptische Kurven . . . . .	2
2.1.1	Gruppenstruktur . . . . .	3
2.1.2	Operationen auf Elliptische Kurven . . . . .	3
2.1.3	Projektive Darstellung . . . . .	4
2.2	Effiziente skalare Multiplikationen . . . . .	5
2.2.1	Exponentenrekodierung mit NAF . . . . .	6
2.2.2	Exponentenrekodierung mit Width-w NAF . . . . .	7
2.2.3	Exponentenrekodierung mit Fractional Window . . . . .	8
<b>3</b>	<b>Angriffstypen</b>	<b>10</b>
3.1	Power Analysis . . . . .	11
3.1.1	Simple Power Analysis . . . . .	11
3.1.2	Differential Power Analysis . . . . .	14
3.2	Elektromagnetische Abstrahlung . . . . .	17
3.3	Timing attacks . . . . .	18
3.4	Fault attacks . . . . .	19
3.5	Hardwaregegenmaßnahmen . . . . .	20
<b>4</b>	<b>Gegenmaßnahmen bis 2003</b>	<b>21</b>
4.1	Randomisierte m-ary Methode . . . . .	21
4.1.1	Rekodierung in die m-ary Darstellung . . . . .	21
4.1.2	Prinzip der Gegenmaßnahme . . . . .	22
4.1.3	Sicherheitsbetrachtung . . . . .	23
4.1.4	Effizienzbetrachtung . . . . .	24
4.1.5	Abschlußbetrachtung . . . . .	24
4.2	Randomisierte Skalardarstellung mittels NAF . . . . .	25
4.2.1	Grundlage der Gegenmaßnahme . . . . .	25
4.2.2	Umsetzung der Gegenmaßnahme . . . . .	26
4.2.3	Analyse der Gegenmaßnahme . . . . .	27
4.3	Randomisierte Adressmethode . . . . .	28
4.3.1	Grundlage der Gegenmaßnahme . . . . .	28
4.3.2	Beispielanwendungen . . . . .	29
4.3.3	Sicherheitsbetrachtung . . . . .	30
4.3.4	Abschlußbetrachtung . . . . .	31

## Inhaltsverzeichnis

4.4	C-MOS Logik als Hardwaregegenmaßnahme . . . . .	31
4.4.1	Grundlage der Gegenmaßnahme . . . . .	31
4.4.2	Funktionsweise der C-MOS Logikbausteine . . . . .	32
4.4.3	Experimentelle Ergebnisse . . . . .	32
4.4.4	Abschlußbetrachtung . . . . .	33
4.5	Hardwaregegenmaßnahme mittels Einsatz von Kondensatoren . . . . .	34
4.5.1	Grundlage der Gegenmaßnahme . . . . .	34
4.5.2	Schwachstelle . . . . .	35
4.5.3	Vergleich zu Vorgänger . . . . .	35
4.5.4	Abschlußbetrachtung . . . . .	36
<b>5</b>	<b>Neue Gegenmaßnahmen ab 2003</b>	<b>37</b>
5.1	Ununterscheidbarer Code für ECADD und ECDBL . . . . .	37
5.1.1	Grundlage der Gegenmaßnahme . . . . .	37
5.1.2	Vereinheitlichung der Operationen . . . . .	38
5.1.3	Abschlußbetrachtung . . . . .	38
5.2	Parametrisierung in elliptischen Kurven . . . . .	39
5.2.1	Grundlagen der Gegenmaßnahme . . . . .	39
5.2.2	Umsetzung der Gegenmaßnahme . . . . .	39
5.2.3	Gruppenaxiome auf $C_M$ . . . . .	41
5.2.4	Abschlußbetrachtung . . . . .	41
5.3	RPA - ZPA Gegenmaßnahmen . . . . .	41
5.3.1	Refined Power Analysis . . . . .	41
5.3.2	Zero Value Point Attack . . . . .	42
5.3.3	Gegenmaßnahmen . . . . .	42
5.3.4	Effizienzbetrachtung . . . . .	45
5.3.5	Abschlußbetrachtung . . . . .	45
<b>6</b>	<b>Neue Angriffe</b>	<b>46</b>
6.1	Seitenkanäle während der Exponentenrekodierung . . . . .	46
6.1.1	Grundlagen . . . . .	46
6.1.2	Seitenkanal . . . . .	46
6.1.3	Angriffe . . . . .	47
6.1.4	Experimentelle Ergebnisse . . . . .	52
6.1.5	Abschlußbetrachtung . . . . .	52
6.2	Simple Branch Prediction Analysis . . . . .	53
6.2.1	Grundlagen des Angriffs . . . . .	53
6.2.2	Aufbau der Sprungvorhersage Einheit . . . . .	54
6.2.3	Angriff . . . . .	54
6.2.4	Messung . . . . .	55
6.2.5	Heuristik . . . . .	56
6.2.6	Fazit . . . . .	56
<b>7</b>	<b>Vergleich aller Gegenmaßnahmen</b>	<b>57</b>

## *Inhaltsverzeichnis*

7.1	Bewertungsschema . . . . .	57
7.2	Standardalgorithmen und Gegenmaßnahmen . . . . .	59
7.3	Auswertung . . . . .	60
7.4	Erweiterung der Tabelle um die neuen Angriffstechniken . . . . .	62

# Liste der Algorithmen

1	binäre Skalarmultiplikation, LSB . . . . .	6
2	binäre Skalarmultiplikation, MSB . . . . .	6
3	Exponentenrekodierung mit NAF . . . . .	7
4	Exponentenrekodierung mit Width-w NAF . . . . .	7
5	Exponentenrekodierung mit Fractional Window, vorzeichenlos . . . . .	8
6	Exponentenrekodierung mit Fractional Window, vorzeichenbehaftet . . . . .	9
7	Add-and-double-always Methode (MSB) . . . . .	13
8	Randomisierte m-ary Methode, nur DPA-sicher . . . . .	22
9	Randomisierte m-ary Methode, SPA- und DPA-sicher . . . . .	23
10	Addition-Subtraktions Algorithmus . . . . .	25
11	Addition-Subtraktions Algorithmus, SPA sicher . . . . .	28
12	einfache Add-and-double-always Methode . . . . .	29
13	einfacher Montgomery Ladder . . . . .	29
14	modifizierte Add-and-double-always Methode, Adressbit DPA-sicher . . . . .	30
15	modifizierter Montgomery Ladder, Adressbit DPA-sicher . . . . .	30
16	bitabhängige, binäre Skalarmultiplikation . . . . .	37
17	BRIP-Algorithmus . . . . .	42
18	EBRIP-Algorithmus . . . . .	44
19	Angriff auf die Exponentenrekodierung mit Width-w NAF . . . . .	49
20	Angriff auf die vorzeichenlose Fractional Window Methode . . . . .	50
21	Angriff auf die vorzeichenbehaftete Fractional Window Methode . . . . .	51

# Abbildungsverzeichnis

2.1	Operationen auf elliptischen Kurven . . . . .	4
3.1	Spannungsverlaufskurve . . . . .	12
3.2	Fehlerinduktion bei RSA-Berechnung . . . . .	19
4.1	CMOS-Anordnungen . . . . .	31
4.2	Differenzierte und Dynamische Logik . . . . .	32
4.3	Schaltplan der CMOS-Anordnungen . . . . .	33
4.4	Smart Card Schaltplan . . . . .	35
4.5	Energieverbrauch mit dazwischengeschalteten Kondensatoren . . . . .	35
6.1	Pipeline-Architektur . . . . .	53
6.2	BPU Architektur . . . . .	54
6.3	Messergebnisse bei SPBA . . . . .	56

# Tabellenverzeichnis

4.1	Vergleich der Laufzeiten . . . . .	24
4.2	NAF Rekodierungstabelle . . . . .	26
4.3	NAF Rekodierungstabelle, vorzeichenbehaftet . . . . .	27
4.4	Vergleich der NED-Werte . . . . .	33
5.1	Vergleich der Gegenmaßnahmen bzgl. Rechenaufwand . . . . .	45
6.1	Sequenzentsprechungen bei Width-3 NAF . . . . .	47
6.2	Messergebnisse der reproduzierten Bits . . . . .	52
7.1	Vergleich der Gegenmaßnahmen . . . . .	61
7.2	Gesamttabelle der Gegenmaßnahmen . . . . .	65

# 1 Einleitung

In der Literatur finden sich viele Gegenmaßnahmen zu Seitenkanalangriffen. Im wesentlichen bieten die Arbeiten der CHES-Workshops die Basis der Gegenmaßnahmen. Da alle Arbeiten mehr oder weniger für sich alleine stehen, soll eine Übersicht geschaffen werden, die die unterschiedlichsten Abwehrtechniken auflistet und es ermöglicht die verschiedenen Gegenmaßnahmen und deren Kombinationen vergleichen zu können. Die Gegenmaßnahmen werden hinsichtlich ihrer Sicherheit, ihrer Rechengeschwindigkeit und ihres Speicherverbrauchs verglichen. Der Leser soll dadurch einen Eindruck gewinnen, was für seine Bedürfnisse die geeignetere Gegenmaßnahme ist.

Kapitel 2 gibt eine Einführung in die mathematischen Grundlagen elliptischer Kurven und listet die, für diese Arbeit grundlegenden Algorithmen bzgl. der skalaren Multiplikation auf elliptischen Kurven auf.

Kapitel 3 gibt eine detaillierte Übersicht aller bekannten Angriffstechniken und ihren korrespondierenden Gegenmaßnahmen. Weiterhin wird eine zeitliche Entwicklung der bekannten Angriffstechniken und den aus diesen, durch Modifizierungen, entstandenen, verbesserten Angriffstechniken wiedergegeben.

Kapitel 4 beschreibt konkrete Implementierungen von Gegenmaßnahmen gegen Seitenkanalangriffe, die vor 2003 veröffentlicht wurden.

Neue Angriffe hebeln die alten Angriffe aus Kapitel 4 aus. Kapitel 5 beschreibt verbesserte Angriffstechniken und neue Gegenmaßnahmen, die Erweiterungen und Modifikationen der Gegenmaßnahmen aus Kapitel 4 darstellen, die wiederum gegen diese neuen Angriffstechniken absichern.

Kapitel 6 beschreibt einen neuen Angriff „Simple Branch Prediction Analysis“, der alle zuvor beschriebenen Gegenmaßnahmen aushebelt, da er, im Vergleich zu den bisher gängigen Sicherungsverfahren, einem völlig anderem Schema folgt. Dieser Angriff gefährdet alle asymmetrischen Verschlüsselungsverfahren.

In Kapitel 7 wird ein Bewertungsschema formuliert mit dem, die in dieser Arbeit beschriebenen Angriffe und Gegenmaßnahmen verglichen werden, welche in der abschließenden Gesamttabelle aufgelistet werden.

## 2 Mathematische Grundlagen

In diesem Kapitel werden die, für diese Arbeit notwendigen, mathematischen Grundlagen erklärt. Abschnitt 2.1 führt in elliptische Kurven ein. Abschnitt 2.2 beschreibt Algorithmen, mit denen die skalare Multiplikation  $dP$  durchgeführt werden.

### 2.1 Elliptische Kurven

Elliptische Kurven sind ein Forschungsgebiet der mathematischen Zahlentheorie und der Algebraik. Ursprünglich aus dem Problem entstanden den Umfang einer Ellipse zu berechnen, finden sie heute große Anwendung in der Kryptographie. Elliptische-Kurven-Kryptosysteme (engl. Elliptic Curve Cryptosystem, kurz ECC)<sup>1</sup> basieren darauf, dass das sogenannte diskrete Logarithmus-Problem im Kontext elliptischer Kurven schwer lösbar ist [Buc02]. Da elliptische Kurven beim Faktorisieren mit kleineren Schlüsselgrößen, im Vergleich zu anderen asymmetrischen Verfahren, auskommen, werden diese bevorzugt bei Smart Cards eingesetzt, da hier begrenzte Rechen- und Speicherressourcen vorhanden sind. Im folgenden werden die Aspekte von elliptischen Kurven erläutert, die für diese Arbeit relevant sind. Wir beschränken uns auf elliptische Kurven über Körpern  $\mathbb{F}_P$  vom Grad größer als 3.

**Definition 2.1.1** *Eine elliptische Kurve  $E$  über einem Körper  $K$  ist definiert durch die Menge der Lösungen der allgemeinen Weierstraß Gleichung*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

mit  $a_1, a_2, a_3, a_4, a_6 \in K$ .

Unter der Beschränkung auf elliptische Kurven vom Grad größer als 3 ( $\text{char}(K) \neq 2, 3$ ) kann man die verkürzte Gleichung

$$E : y^2 = x^3 + ax + b$$

mit  $a, b \in K$  und  $\Delta \neq 0$  anwenden.  $\Delta$  wird als Diskriminante bezeichnet und sieht in der verkürzten Form, wie folgt aus:

$$\Delta = -16(4a^3 + 27b^2)$$

---

<sup>1</sup>Um Übersetzungsunstimmigkeiten vorzubeugen und ein leichteres Nachvollziehen der vorwiegend englischen Fachbegriffe zu gewährleisten, werden in dieser Arbeit auch englische Begrifflichkeiten und Akronyme verwendet. Es werden die deutschen Entsprechungen eingeführt und in der Folge die englischen Akronyme benutzt.

## 2 Mathematische Grundlagen

Als neutrales Element beinhaltet jede elliptische Kurve den Punkt  $\mathcal{O}$ , der im Unendlichen auf der projektiven Ebene liegt. Die daraus resultierende Kurvengleichung, sieht wie folgt aus:

$$E(K) = \{(x, y) \in K_2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

**Definition 2.1.2** Eine elliptische Kurve  $E$  heißt *glatt*, genau dann wenn die Diskriminante  $\Delta(E) \neq 0$  ist.

Dies ist gleichbedeutend damit, dass eine der beiden partiellen Ableitungen  $\frac{\partial E}{\partial x}$  oder  $\frac{\partial E}{\partial y}$  ungleich 0 ist.

### 2.1.1 Gruppenstruktur

Für elliptische Kurven gelten folgende Axiome:

**Lemma 2.1.1** Sei  $E$  eine elliptische Kurve über  $\mathbb{F}_P$  und  $P$  und  $Q$  zwei Punkte auf  $E$ , dann gilt [H.S86]:

1.  $P + \mathcal{O} = \mathcal{O} + P = P$
2.  $P + Q \in E(K)$
3. das Inverse von  $P=(x,y)$  ist  $-P=(x,-y)$
4.  $P+(-P) = \mathcal{O}$

Hierbei stellt  $\mathcal{O}$  das neutrale Element bzgl. der Addition in elliptischen Kurven dar.

### 2.1.2 Operationen auf Elliptische Kurven

Die Punkte auf einer elliptischen Kurve bilden zusammen mit der Punktaddition (engl. Elliptic Curve Addition, ECADD) und dem Punkt  $\mathcal{O}$  im Unendlichen als neutrales Element eine additive abelsche Gruppe. Die Punktaddition  $P + Q = R$  lässt sich graphisch, wie folgt beschreiben. Man lege eine Gerade durch die beiden Punkte  $P$  und  $Q$ . Diese Gerade schneidet die elliptische Kurve in einem dritten Punkt  $R^*$ , der den an der x-Achse gespiegelte Punkt  $R$  darstellt (vergl. Abbildung 2.1 (links)).

Für den Fall dass zwei identische Punkte addiert werden ( $P = Q$ ), spricht man von einer Verdopplung, (engl. Elliptic Curve Double, ECDBL). Die Berechnung  $P + P = 2P = R$  entspricht, graphisch betrachtet, einer Tangente durch den Punkt  $P$  an der elliptischen Kurve. Auch hier entspricht der Schnittpunkt  $R^*$  dem an der x-Achse gespiegelten Punkt  $R$  (vergl. Abbildung 2.1 (rechts)).

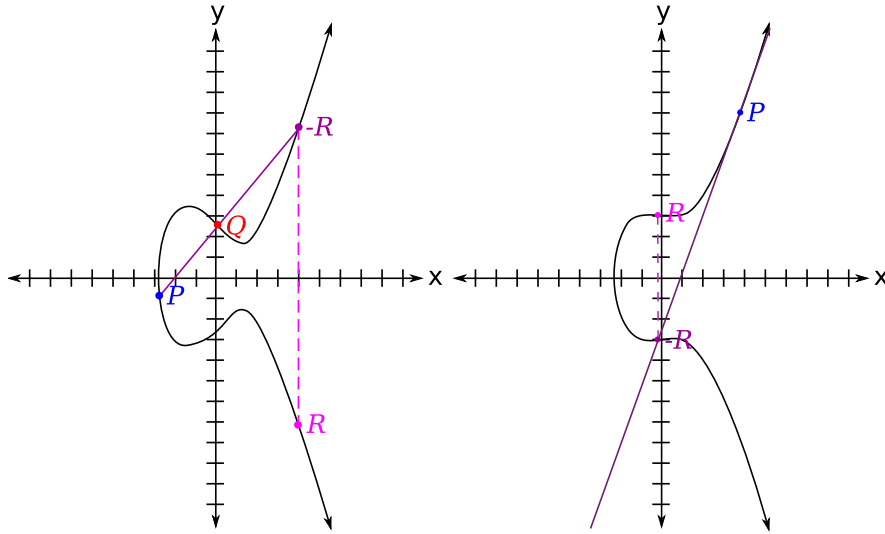


Abbildung 2.1: Addition (links) und Verdopplung (rechts)

Für die Addition (im folgenden verkürzt ECADD) und Verdopplung (im folgenden verkürzt ECDBL) gelten folgende Gleichungen

**Definition 2.1.3 (Operation mit affinen Koordinaten)** Seien  $P_1 = (x_1, y_1)$  und  $P_2 = (x_2, y_2)$  Punkte auf der elliptischen Kurve  $E$ , dann berechnet sich  $P_1 + P_2 = P_3 = (x_3, y_3)$ , wie folgt

$$x_3 = \begin{cases} \lambda_1^2 - x_1 - x_2, & \text{für } P_1 \neq P_2 \\ \lambda_2^2 - 2x_1, & \text{für } P_1 = P_2 \end{cases}$$

$$y_3 = \begin{cases} (x_1 - x_3)\lambda_1 - y_1, & \text{für } P_1 \neq P_2 \\ (x_1 - x_3)\lambda_2 - y_1, & \text{für } P_1 = P_2 \end{cases}$$

$$\text{mit } \lambda_1 = \frac{y_2 - y_1}{x_2 - x_1} \text{ und } \lambda_2 = \frac{3x_1^2 + a}{2y_1}$$

### 2.1.3 Projektive Darstellung

Alternativ zur affinen Darstellung kann man die Punkte einer elliptischen Kurve auch in projektiver Form angeben. Diese Darstellung hat den Vorteil, dass die kostenintensive Berechnung eines inversen Punktes, die sowohl bei ECADD als auch ECDBL vorkommt, wegfällt. Hierbei wird jeder Punkt  $P = (x, y)$  in affiner Darstellung in seine projektive Darstellung  $P = (xZ^\alpha, yZ^\beta, Z)$  mit  $Z \in F_P$  und  $\alpha, \beta \in N$  abgebildet. Die folgende Definition zeigt die Operationen in einer speziellen projektiven Form, der Jacobian Darstellung, hier gilt  $\alpha = 2, \beta = 3$  [CM098].

**Lemma 2.1.2 (Operation mit projektiven Koordinaten)** Seien  $P_1=(x_1, y_1, z_1)$  und  $P_2=(x_2, y_2, z_2)$  Punkte auf der elliptischen Kurve  $E$ , dann berechnet sich  $P_1 + P_2=P_3=(x_3, y_3, z_3)$ , wie folgt [H.S86]:

für ECADD ( $P_1 \neq P_2$ ) gilt:

$$\begin{aligned} x_3 &= T, \\ y_3 &= -8y_1^4 + M(S - T), \\ z_3 &= 2y_1z_1, \end{aligned}$$

$$\text{mit } S=4x_1y_1^2, M = 3x_1 + aZ_1^4, T = -2S + M^2$$

für ECDBL ( $P_1 = P_2$ ) gilt:

$$\begin{aligned} x_3 &= -HY^3 - 2U_1H^2 + R^2, \\ y_3 &= -S_1H^3 + R(U_1H^2 - x_3), \\ z_3 &= z_1z_2H, \end{aligned}$$

$$\text{mit } U = x_1z_2^2, U_2 = x_2z_1^2, S_1 = y_1z_2^3, \\ S_2 = y_2z_1^3, H = U_2 - U_1, R = S_2 - S_1$$

Um mittels ECC eine vergleichbare Sicherheit, wie eine 1024-Bit RSA-Verschlüsselung zu erreichen, ist es notwendig, dass die entsprechende elliptische Kurve eine Anzahl von mindestens  $2^{163}$  Punkten aufweist [Buc02]. Eine erste Näherung erhält man durch das folgende Hasse Theorem:

**Theorem 2.1.1 (Hasse)** Sei  $E(K)$  eine elliptische Kurve über einem endlichen Primkörper  $\mathbb{F}_p$  mit  $p$  Elementen, dann ist die Anzahl der Punkte auf der Kurve  $E$ :  $|E(K)| = p+1-t$  mit  $|t| < 2\sqrt{p}$

## 2.2 Effiziente skalare Multiplikationen

Dieser Abschnitt listet die für diese Arbeit grundlegenden Algorithmen bzgl. der skalaren Multiplikation bzw. der Punktmultiplikation  $dP$  auf elliptischen Kurven auf. Im folgenden stellen

- $d = d_{n-1}d_{n-2} \dots d_0$  ein Skalar und
- $P$  ein Punkt auf einer elliptischen Kurve dar.

Algorithmus 1 beschreibt den einfachen right-to-left Algorithmus [MOV96] zur Berechnung des skalaren Produkts  $dP$ . Der Algorithmus durchläuft den Skalar vom niederwertigsten zum höherwertigstem Bit. Für jedes gesetztes Bit erfolgt eine zusätzliche ECADD-Operation in Schritt 1.3 von Algorithmus 1.

---

**Algorithmus 1** : right-to-left Algorithmus, LSB

---

**Input** :  $P \in E, d = d_{n-1}d_{n-2}\dots d_0$ **Output** : Skalarprodukt  $dP$ 

```

1.1  $A \leftarrow \mathcal{O}, S \leftarrow P$ 
1.2 for  $i = 0$  to  $n - 1$  do
1.3   | if  $d_i = 1$  then  $A \leftarrow A + S$  ;
1.4   | if  $d_i \neq n - 1$  then  $S \leftarrow 2S$  ;
1.5 end
1.6 return  $A$ 

```

---

Algorithmus 1 benötigt  $(n-1)$  ECDBL-Operationen (Schritt 1.4) und so viele ECADD Operationen, wie es gesetzte Bits im Skalar  $d$  (Schritt 1.3) gibt. Der Wert für die Anzahl der gesetzten Bits, d.h. die Anzahl der Einzen einer Binärfolge, bezeichnet man auch als Hamminggewicht der jeweiligen Bitfolge. Das Hamminggewicht einer Bitfolge  $d$  wird im folgenden verkürzt mit  $H(d)$  angegeben. Algorithmus 1 durchläuft anders als Algorithmus 2 den Skalar bitweise vom höherwertigstem zum niederwertigstem Bit. Ein großer Vorteil im Vergleich zu Algorithmus 1 liegt darin, dass eine Speichereinheit weniger benötigt wird, was bei begrenzten Speicher- und Rechenressourcen, wie beispielsweise bei Smartcards von großem Vorteil ist. Algorithmus 2 findet daher größere Anwendung.

---

**Algorithmus 2** : left-to-right Algorithmus, MSB

---

**Input** :  $P \in E, d = d_{n-1}d_{n-2}\dots d_0$ , wobei  $d_{n-1}$  gesetzt ist und  $d > 0$ **Output** : Skalarprodukt  $dP$ 

```

2.1  $A \leftarrow P$ 
2.2 for  $i = n - 2$  to  $0$  do
2.3   |  $A \leftarrow 2A$  ;
2.4   | if  $d_i = 1$  then  $A \leftarrow A + P$  ;
2.5 end
2.6 return  $A$ 

```

---

Algorithmus 2 benötigt  $(n-2)$  ECDBL-Operationen (Schritt 2.3) und  $H(d)$  ECADD-Operationen (Schritt 2.4) [MOV96].

**2.2.1 Exponentenrekodierung mit NAF**

Um die Berechnung der skalaren Punktmultiplikation bzgl. der Laufzeit zu optimieren, wird häufig der Multiplikand statt in binärer Form in eine kompaktere Form mit einer geringeren Anzahl an Nichtnullwerten überführt. Dieser Vorgang wird als „Exponentenrekodierung“ bezeichnet. Die Exponentenrekodierung wird dann vor der eigentlichen skalaren Multiplikation durchgeführt.

Algorithmus 3 zeigt die Umsetzung in eine vorzeichenbehaftete Binärform, die man auch als NAF (engl. non-adjacent form) bezeichnet. Nach der Rekodierung können zwei benachbarte Ziffern nicht gleichzeitig den Wert null haben. Die Ziffern sind aus der Menge

$\{-1,0,1\}$ . Die Schritte 3.3 und 3.4 des Algorithmus kann man auch mittels einer Tabelle ausführen, die alle möglichen Werte auflistet. Durch Anwendung von NAF reduziert sich die Ziffernlänge des Exponenten auf  $1/3$  der ursprünglichen Länge für große Werte von  $d$ .

---

**Algorithmus 3** : Exponentenrekodierung mit NAF

---

**Input** : eine nicht-negative  $t$ -bit lange Zahl  $d = (d_{t-1} \dots d_0)_2$   
**Output** :  $b = (b_t \dots b_0)$ , wobei  $b_i \in \{-1, 0, 1\}$  und  $b_i b_{i+1} = 0 \forall i$

3.1  $c_0 \leftarrow 0, d_{t+1} \leftarrow 0, d_t \leftarrow 0$   
3.2 **for**  $i = 0$  **to**  $t$  **do**  
3.3 |  $c_{i+1} \leftarrow (c_i + d_i + d_{i+1})/2$  ;  
3.4 |  $b_i \leftarrow c_i + d_i - 2c_{i+1}$  ;  
3.5 **end**  
3.6 **return**  $b = (b_t \dots b_0)$

---

### 2.2.2 Exponentenrekodierung mit Width-w NAF

Width-w NAF ist eine verallgemeinerte Version von NAF. Für den Fall, dass  $w = 2$  ist, entspricht dieser dem einfachen NAF Algorithmus. Der Wert  $w$  gibt die Anzahl der Bits an, die zusammengefaßt werden. Man bezeichnet  $w$  auch als Fenstergröße. Algorithmus 4 stellt eine typische Implementierung von Width-w NAF dar. Hier reduziert sich die Binärlänge für große Werte von  $d$  auf  $1/(w+1)$  der ursprünglichen Länge und die Ziffern sind aus einer größeren Menge entnommen, die abhängig vom Wert  $w$  sind.

---

**Algorithmus 4** : Exponentenrekodierung mit Width-w NAF

---

**Input** : eine nicht-negative  $t$ -bit lange Zahl  $d = (d_{t-1} \dots d_0)_2$ , eine Zahl  $w \geq 2$   
**Output** :  $b = (b_t \dots b_0)$ , wobei  $b_i \in \{0, \pm 1, \pm 3e_m, \dots, \pm(2^{w-1} - 1)\}$

4.1  $c_0 \leftarrow d, i \leftarrow 0$   
4.2 **while**  $c > 0$  **do**  
4.3 | **if**  $c$  *ungerade* **then**  
4.4 | |  $b_i \leftarrow c \bmod 2^w$  ;  
4.5 | | **if**  $b_i \geq 2^{w-1}$  **then**  
4.6 | | |  $b_i \leftarrow b_i - 2^w$  ;  
4.7 | | **end**  
4.8 | |  $c \leftarrow c - b_i$  ;  
4.9 | **else**  
4.10 | |  $b_i \leftarrow 0$  ;  
4.11 | **end**  
4.12 |  $c \leftarrow \lfloor c/2 \rfloor, i \leftarrow i + 1$  ;  
4.13 **end**  
4.14 **return**  $b = (b_t \dots b_0)$

---

### 2.2.3 Exponentenrekodierung mit Fractional Window

Für den Fall, dass die Fenstergröße  $w$  bei Width- $w$  NAF eingeschränkt ist, beispielsweise aufgrund begrenzter Speicherressourcen auf Smart Cards, kann die Fractional Window Methode eingesetzt werden. Hier können auch Dezimalzahlen, wie z.B. 2.5 verwendet werden, weiterhin ist eine Beschränkung auf positive Zahlen als Nichtnullwerte für die Ziffern  $b_i$  der Zahlenfolge möglich. Fractional Window stellt eine Annäherung an die Width- $w$  Methode dar. Die durchschnittliche Verkürzung von  $d$  lässt sich, wie folgt angeben. Sei  $w \geq 2$  und  $m$  eine Zahl, so dass  $1 \geq m \geq 2^w - 3$  gilt. Dann lässt sich die Reduzierung der Binärlänge beim vorzeichenlosen Algorithmus für große Werte  $d$  mit

$$1/(w+(m+1)/2^w+2)$$

angeben. Für den vorzeichenbehafteten Algorithmus lässt sich die reduzierte Binärlänge mit

$$1/(w+(m+1)/2^w+1)$$

angeben. Algorithmus 5 und Algorithmus 6 zeigen die vorzeichenbehaftete und die vorzeichenlose Exponentenrekodierung mittels Fractional Window.

---

**Algorithmus 5** : Exponentenrekodierung mit vorzeichenloser Fractional Window

---

**Input** : eine nicht-negative Zahl  $d$ , eine Zahl  $w \geq 2$ , eine gerade Zahl  $m$   
mit  $1 \leq m \leq 2^w - 3em$

**Output** :  $b = (b_{i-1} \dots b_0)$ , wobei  $b_i \in \{0, 1, 3em, \dots, (2^w + m)\}$

```

5.1  $c_0 \leftarrow d, i \leftarrow 0$ 
5.2 while  $c > 0$  do
5.3   if  $c$  ungerade then
5.4      $b_i \leftarrow c \bmod 2^{w+1};$ 
5.5     if  $2^w + m < b_i$  then
5.6        $b_i \leftarrow b_i - 2^w;$ 
5.7     end
5.8      $c \leftarrow c - b_i;$ 
5.9   else
5.10     $b_i \leftarrow 0;$ 
5.11   end
5.12   $c \leftarrow \lfloor c/2 \rfloor, i \leftarrow i + 1;$ 
5.13 end
5.14 return  $b_{i-1} \dots b_0$ 

```

---

---

**Algorithmus 6** : Exponentenrekodierung mit vorzeichenbehafteter Fractional Window

---

**Input** : eine nicht-negative lange Zahl  $d$ , eine Zahl  $w \geq 2$ , eine gerade Zahl  $m$  mit  $1 \leq m \leq 2^w - 3em$

**Output** :  $b = (b_{i-1} \dots b_0)$ , wobei  $b_i \in \{0, \pm 1, \pm 3em, \dots, \pm(2_w + m)\}$

```

6.1  $c_0 \leftarrow d, i \leftarrow 0$ 
6.2 while  $c > 0$  do
6.3   if  $c$  ungerade then
6.4      $b_i \leftarrow c \bmod 2^{w+2};$ 
6.5     if  $2^w + m < b_i < 3em \cdot 2^w - m$  then
6.6        $b_i \leftarrow b_i - 2^{w+1};$ 
6.7     else if  $3em \cdot 2^w - m \leq b_i \leq 2^{w+2}$  then
6.8        $b_i \leftarrow b_i - 2^{w+2};$ 
6.9     end
6.10     $c \leftarrow c - b_i;$ 
6.11  else
6.12     $b_i \leftarrow 0;$ 
6.13  end
6.14   $c \leftarrow \lfloor c/2 \rfloor, i \leftarrow i + 1;$ 
6.15 end
6.16 return  $b_{i-1} \dots b_0$ 

```

---

## 3 Angriffstypen

Kryptographische Verschlüsselungstechniken bzw. kryptographische Primitive stellen in der Regel abstrakte mathematische Modelle dar. Alternativ kann man sie auch als komplexe Funktionen betrachten, die einen Wert aus der Menge der Klartexte auf einen Wert in die Menge der Ciphertexte abbilden. Diese kryptographischen Primitive werden in Softwareprogrammen implementiert, die dann auf einer bestimmten Hardware mit ihren jeweils spezifischen Komponenten laufen. Dies impliziert automatisch bestimmte Charakteristika, die für jedes technische System eigen sind. Diese spezifischen Charakteristika stellen unter Umständen Seitenkanäle dar, durch die ein Angreifer Rückschlüsse über verwendete geheime Schlüssel, die in die kryptographische Verschlüsselung einfließen, erhalten kann. Dieser Ansatz der Kryptoanalyse wird als Seitenkanalangriff bezeichnet (engl. Side Channel Attack, SCA).

Paul Kocher veröffentlichte 1996 die erste Arbeit zu Seitenkanalangriffen bei kryptographischen Berechnungen [Koc96]. Er erkannte, dass es neben der Lösung des mathematischen Problems auch andere Wege zur Entschlüsselung von sensiblen Daten führen können. Seitenkanalangriffe analysieren die Eigenschaften eines Algorithmus. Dazu wird eine statistische Analyse über eine große Anzahl von Messungen durchgeführt. Diese Angriffe beziehen sich auf alle messbaren „Seitengeräusche“, die für die jeweilige technische Plattform charakteristisch sein können. Das Hauptangriffsziel von Seitenkanalangriffen stellen die Berechnungen auf Smart Card Chips dar, worauf sich die vorliegende Arbeit hauptsächlich konzentrieren wird. Die Überlegungen lassen sich jedoch problemlos auf andere PC Plattformen übertragen. Eine SmartCard stellt wohl die größte Angriffsfläche dar. Hier kann man die Berechnungsdauer (Timing Attack), den Energieverbrauch (Power Analysis) und die elektromagnetische Abstrahlung (EMA) einer verwendeten technischen Plattform messen.

Dieses Kapitel soll im folgenden eine Übersicht über alle gängigen Angriffsarten und deren typischen Gegenmaßnahmen auflisten. Wobei stets zu betonen ist, dass es keine ‚perfekte‘ Gegenmaßnahme gibt. Die Entwicklungen sowohl von Angriffstechniken und ihren jeweiligen Gegenmaßnahmen stellt einen Prozeß dar, der sich Hand in Hand parallel entwickelt. Ein realistisches Ziel einer Gegenmaßnahme kann es nur sein, den Aufwand für einen Angreifer dermaßen zu erhöhen, so dass dieser an die Grenzen seiner finanziellen und technischen Mittel stößt.

Die folgende Auflistung [Ava05] stellt eine kurze historische Abfolge von Seitenkanalangriffen dar

- 1996: Timing Attacks

- 1996: Fault Analysis bei RSA
- 1997: Differential Fault Analysis (DFA)
- 1998: Simple Power Analysis (SPA)
- 1998: Differential Power Analysis (DPA)
- 2000: Higher Order DPA
- 2002: EM Seitenkanäle
- 2003: Goubin-Type Analysis

Diese Auflistung stellt keine zwingend chronologische Reihenfolge dar, jedoch spiegelt sie einen kleinen Leitfaden für die im folgenden aufgelisteten Angriffstypen wider.

## 3.1 Power Analysis

Power Analysis misst den Spannungsverlauf während einer kryptographischen Berechnung. Da beispielsweise Smart Cards über eine externe Stromquelle mit Energie versorgt werden, ist dies ohne weiteres möglich. Aus der resultierenden Spannungsverlaufskurve versucht man auf die ausgeführten Operationen des Algorithmus zu schließen. Anders formuliert, ein Angreifer versucht durch Beobachtung des gemessenen Spannungsverlaufs, Rückschlüsse auf verwendete sensible Daten zu erhalten. Wie es zu solchen Seitenkanälen kommt, kann wie folgt erklärt werden. Heutzutage besteht jede technische Einheit in ihrer elementaren Form aus Mikrochips, diese wiederum werden in aufwendigen technischen Prozessen aus Transistoren zusammengestellt, aus denen logische Bausteine geformt werden können. Der Energieverbrauch dieser einzelnen logischen Bausteine hängt eng mit den ausgeführten Operationen, den gespeicherten Daten in den Registern, den momentan abgearbeiteten Befehlen der jeweiligen Befehlsstufe usw. ab. Charakteristisches Merkmal hierbei ist stets das Hamminggewicht [Ava05]. Denn ein gesetztes Bit entscheidet in der Regel, ob eine relativ rechenintensive Operation, wie beispielsweise eine Multiplikation bei RSA oder eine ECADD-Operationen bei elliptischen Kurven, ausgeführt wird (vergl. Abbildung 3.1).

Es wird in der Regel zwischen ‚Simple Power Analysis‘ (SPA) und ‚Differential Power Analysis‘ (DPA) unterschieden.

### 3.1.1 Simple Power Analysis

Bei der SPA-Methode betrachtet man lediglich den Spannungsverlauf einer *einzelnen* kryptographischen Berechnung. Hier versucht man anhand der Spannungsverlaufskurve einzelne Operationen, wie Multiplikationen und Quadrierungen bei RSA bzw. ECDBLs und ECADDs bei elliptischen Kurven zuzuordnen. In der Regel verursachen Multiplikationen und ECADDs höhere Ausschläge bzw. peaks als Quadrierungen und ECDBLs.

### 3 Angriffstypen

Aus den Operationsabfolgen versucht man, die verwendete Zahlenfolge zu rekonstruieren. Zur Verdeutlichung ein Beispiel: Bei der Elliptischenkurvenkryptographie wird ein ECDBL (D) durchgeführt, wenn das jeweilige Bit eine 0 ist. Ist es jedoch eine 1, wird zusätzlich ein ECADD (A) ausgeführt (vergl. Schritt 2.3 und 2.4 von Algorithmus 2). Wenn sich an der Spannungskurve nun die Folge ‚DADDADDDADADADA‘ ablesen lässt, ergibt sich der zugehörige Schlüssel 101001111, denn ‚DA‘ bedeutet eine 1 und die verbleibenden ‚D‘'s repräsentieren jeweils eine 0. Um gegen eine SPA Maßnahme zu sichern, kann eine der 3 folgenden Gegenmaßnahmen verwendet werden.

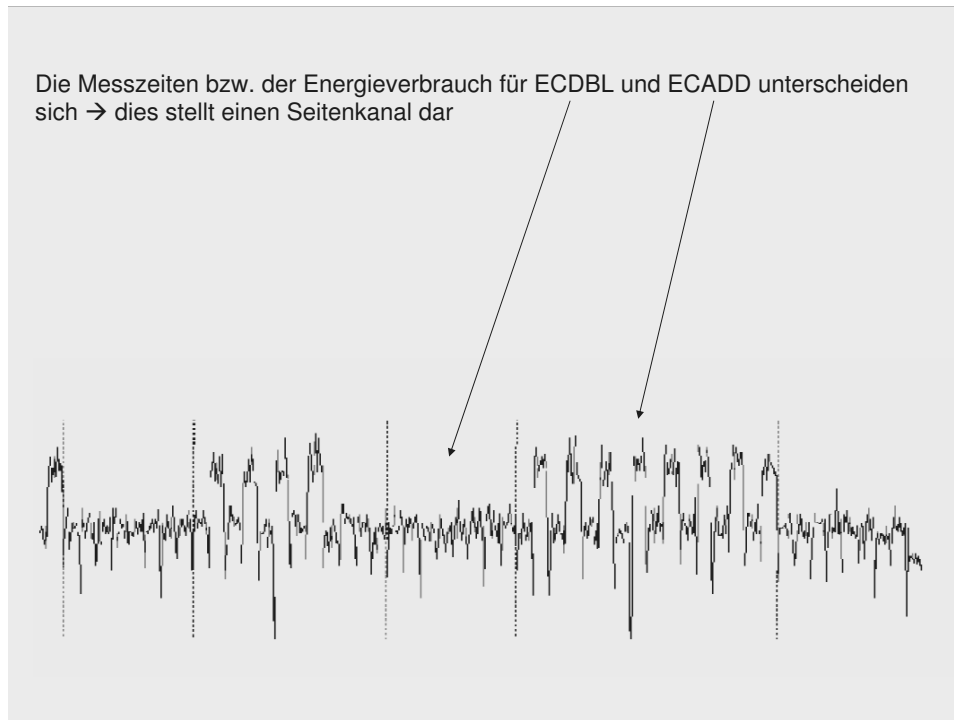


Abbildung 3.1: Spannungsverlaufskurve

#### 3.1.1.1 Dummy Operationen

Die häufigst verwendete Gegenmaßnahme gegen SPA Maßnahmen stellt die Modifikation des zugrundeliegenden Algorithmus durch Einbau von sogenannten „Dummy Operationen“.

---

**Algorithmus 7** : Add-and-double-always Methode (MSB)

---

**Input** :  $P \in E, d = d_{n-1}d_{n-2}\dots d_0$ **Output** : Skalarprodukt  $dP$ 

```

7.1  $T[0] = P$ 
7.2 for  $i = n - 2$  to  $0$  do
7.3    $T[0] \leftarrow \text{ECDBL}(T[0])$ 
7.4    $T[1] \leftarrow \text{ECADD}(T[0], P)$ 
7.5    $T[0] \leftarrow T[d[i]]$ 
7.6 end
7.7 return  $T[0]$ 

```

---

Durch den Einbau von Dummy Operationen wird der iterative Teil der Berechnungen homogener gestaltet, da auf diese Weise stets die gleichen Operationen aufeinander folgen. Algorithmus 7 führt im iterativen Teil stets die gleichen Operationen aus. Der Add-and-double-always Algorithmus stellt eine Modifikation zum einfachen left-to-right Algorithmus (Algorithmus 2 auf Seite 6) dar. Die Modifikation in Algorithmus 7 besteht darin, dass die Schritte 7.4 und 7.5 nun immer ausgeführt werden. Im ursprünglichen Algorithmus entscheidet das jeweilige Bit, welche der beiden Operationen ausgeführt wird. Falls das Bit gesetzt ist, wird lediglich Schritt 7.4, ansonsten Schritt 7.5 ausgeführt. Auf diese Weise kann ein potenzieller Angreifer durch ledigliches Beobachten einer einzigen Messung nicht die Bitfolge des verwendeten Schlüssels detektieren.

Schwachstellen dieser Gegenmaßnahme sind jedoch, dass bei einer Zeitmessung über die gesamte Berechnung, das Hamminggewicht trotzdem festgestellt werden kann und weiterhin eine besonders große Anfälligkeit gegen Fault Attacks besteht, die im folgenden noch erklärt werden.

**3.1.1.2 Ununterscheidbare Formeln für ECADD und ECDBL**

Bei diesem Ansatz wird eine alternative, jedoch äquivalente Darstellung einer elliptischen Kurve, auf denen die Operationen stattfinden sollen, entwickelt. In der transformierten Darstellung finden dann die Operationen auf transformierten Punkten statt, die jedoch keine Rückschlüsse auf den ursprünglichen Punkt und somit auf den verwendeten Schlüssel geben sollen (vergl. Abschnitt 5.1). Das Hauptargument für diese Technik ist, dass der Einsatz von Dummy Operationen nicht mehr notwendig ist, und die damit verbundenen Nachteile vermieden werden. Nachteilig ist jedoch die mangelnde Portabilität auf beliebige elliptische Kurven, d.h. nicht für jede beliebige elliptischen Kurve kann eine Transformation so durchgeführt werden, dass die gewünschten Eigenschaften erhalten bleiben [Ava05].

**3.1.1.3 Skalare Multiplikation mit fixer Sequenz**

Bei diesem Ansatz werden die Operatoren auf eine fixe Sequenz erweitert um deren eigentliche Binärlänge zu verschleiern. Dies kann beispielsweise erfolgen indem man kürzere Bitfolgen mit Nullen auffüllt. Ziel ist es hierbei analog zu Dummy Operationen,

einen stets gleichmäßigen Rechenablauf zu gewährleisten, der unabhängig vom jeweils verwendeten Schlüssel ist. Die Nachteile dieser Gegenmaßnahme sind analog zu Dummy Operationen, dass sie anfällig gegen Fault Attacks sind.

#### 3.1.2 Differential Power Analysis

Die meisten Verfahren, die gegen SPA absichern, stellen sich in der Praxis als unsicher gegenüber DPA Maßnahmen dar. Diese Technik macht sich statistischer Analysetools zunutze, wodurch ein homogenisierter Berechnungsablauf, wie es bei SPA üblich ist, keine hohe Sicherheitsstufe mehr gewährleistet.

Bei diesem Angriffstyp handelt es sich um einen sogenannten Angriff mit gewählttem Klartext (Chosen Plaintext), d.h. dass ein Angreifer zu beliebig gewählten Klartexten die entsprechenden Ciphertexte generieren kann. Im Gegenteil zu SPA, werden hierbei ‚viele‘ Strommessungen einer kryptographischen Berechnung durchgeführt. Man sucht nun ein geeignetes Unterscheidungsmerkmal (engl. Selection Function) [Blu05] für die verschiedenen Messungen, die möglichst stark variierende Eingabeparameter haben sollten. Gemäß diesem Unterscheidungsmerkmal sortiert man die Messungen. Dies kann beispielsweise ein einzelnes Bit der Eingabeparameter sein und das Unterscheidungsmerkmal wäre demnach, ob dieses bestimmte Bit gesetzt ist oder nicht. Die gemessenen Spannungsverläufe der sortierten Mengen werden anschließend gemittelt und von diesen Werten die Differenz gebildet. Je nachdem, ob die resultierende Kurve an der untersuchten Stelle einen extremalen Wert aufweist oder nicht, gibt dies Rückschlüsse auf das entsprechend korrespondierende Bit des geheimen Schlüssels. Wiederholt man dieses Prozedere kann man den geheimen Schlüssel bitweise rekonstruieren. Um gegen DPA abzusichern, werden die folgenden Techniken eingesetzt:

##### 3.1.2.1 Skalare Randomisierung

Die Skalare Randomisierung unterscheidet hier zwei Formen:

a) **Die variierte Darstellung des Skalars**

Hierbei wird eine Zahlendarstellung während einer Berechnung in alternativen Darstellungen verwendet. Diese Gegenmaßnahme wurde jedoch bereits früh angezweifelt sicher zu sein [Ava05], da durch unterschiedliche Darstellungen des gleichen Zahlenwerts Teile des Schlüssels rekonstruiert werden können [Wal02].

b) **Die Variierung des Skalars**

Hier wird in der Regel der geheime Schlüssel mit einem Zufallswert  $R$  verknüpft. Beispielsweise mittels der XOR-Funktion, so dass während der Hauptberechnung, der geheime Schlüssel in randomisierter Form vorliegt und so in die Berechnung einfließt. Am Ende der Hauptberechnung wird der Zufallswert  $R$  mittels der entsprechenden Umkehrfunktion (im Falle von XOR, ist es ebenfalls XOR) wieder herausextrahiert, so dass das exakte Ergebnis zum Schluß vorliegt. Diese Technik

hat den Nachteil, dass sie eine Performanzverschlechterung nach sich zieht, jedoch wird sie häufig in Kombination mit anderen Gegenmaßnahmen verwendet [Ava05].

#### 3.1.2.2 Randomisierung der Hauptgruppenelements (4 Arten)

Diese Technik, die in der Literatur auch als ‚blinding‘ Methode (engl. Verschleierungsmethode) bezeichnet wird, randomisiert entweder den Skalar a) oder benutzt alternative Darstellungen des Elements in anderen Gruppen b) und c) oder randomisiert die ganze Berechnung d).

##### a) Randomisierung des Skalars bei jeder Berechnung

Diese Gegenmaßnahme, die in der Literatur auch als Corons 2.Gegenmaßnahme [Ava05, Cor99] beschrieben wird, ersetzt bei der Hauptberechnung die skalare Multiplikation  $nP$  durch  $n(P + R) - S$ , wobei  $R$  ein geheimes Zufallselement ist und  $S = nR$  gilt. Es wird für jede Berechnung ein neues Paar  $(R, S)$  generiert. Auf diese Weise werden die ursprünglich verwendeten Eingabeparameter, während der Berechnungsphase verschleiert, liegen jedoch am Ende in der gewünschten Form wieder vor. Diese Gegenmaßnahme wird häufig in Kombination mit weiteren Gegenmaßnahmen genutzt um beispielsweise gegen Goubins Angriff (vergl. Absatz 5.3.3) abzusichern.

##### b) Randomisierung mittels alternativer Koordinatendarstellungen

Für die in dieser Arbeit betrachteten elliptischen Kurven vom Grade größer 2 existieren in der Regel alternativer Koordinatensysteme, die es ermöglichen, ein und denselben Punkt in verschiedenen Darstellungen anzugeben (vergl. Abschnitt 2.1.3). Falls es möglich ist, die Operationen ECADD und ECDBL für einen Beobachter von außen ununterscheidbar gestalten zu können, in Bezug auf Energieverbrauch und der daraus resultierenden Stromverlaufskurve, stellt diese Technik eine gute Maßnahme dar um DPA zu erschweren (vergl. Abschnitt 5.2).

##### c) Randomisierung mittels eines isomorphen Körpers

Hierbei wird der zugrundeliegende Körper  $F_p$  mit einer zufällig gewählten Funktion  $f(x)$  faktorisiert, so dass im resultierenden Körper  $F_p[x]/(f(x))$  alle Berechnungen mit veränderten Bitfolgen erfolgen. Joye und Tymen beschreiben diese Gegenmaßnahme in [JT01], bei der sich die extreme Performanzverschlechterung als unpraktikabel erwies.

##### d) Randomisierung mittels einer isomorphen Abbildung

Analog zu c) wird hier der zugrundeliegende Körper randomisiert. Dies erfolgt mittels isomorpher Abbildungen zwischen zwei (hyper-) elliptischen Kurven. Diese Form der Gegenmaßnahmen findet seine Anwendung hauptsächlich bei hyperelliptischen Kurven, welche nicht Gegenstand dieser Arbeit sein werden.

### 3.1.2.3 Spezielle Angriffstypen und Erweiterungen

DPA als größte Angriffsklasse, stellt die mächtigste Technik der Kryptoanalyse in diesem Zusammenhang dar. Dies hat zur Folge, dass darauf aufbauend verbesserte Angriffstechniken entwickelt wurden. Die folgenden 3 Angriffstechniken stellen Beispiele für erweiterte DPA Techniken dar

#### a) Goubin-type Analysis

Goubin beschreibt in seiner Arbeit [Gou03] eine potenzielle Schwachstelle für elliptische Kurven. Diese besteht im invarianten Verhalten einiger Punkte einer elliptischen Kurve nach einer Gegenmaßnahme mittels einer der zu vor genannten Randomisierungstechniken. In der Regel haben diese Punkte einen Nullwert in einer der beiden Koordinatenpunkte und sehen wie folgt aus:

$$P_1 = (0,y) \text{ oder } P_2 = (x,0)$$

Das Fixverhalten dieser Punkte lässt sich arithmetisch leicht erklären. Bei einer Randomisierung werden die Punkte  $P_x$  in der Regel mit einem Zufallswert multipliziert. Ein potenzieller Angreifer macht sich nun hierbei zunutze, dass eine Multiplikation mit einem Nullwert ebenfalls Null ergibt, was sich ebenfalls im Spannungsverlauf bemerkbar macht. Mittels bereits vorgestellter statistischer Analyse-tools lässt sich auf den verwendeten Schlüssel schrittweise schließen. Das Auffinden dieser speziellen Punkte, die das genannte Fixverhalten aufweisen, steht im Mittelpunkt dieses Angriffes. Im Falle einer Randomisierung hin zur Jacobian Darstellung würden die randomisierten Punkte, wie folgt: aussehen

$$P_1^* = (0,yz^3,z) \text{ oder } P_2^* = (xz^2,0,z), z = \text{Zufallswert}$$

Die Bedrohung durch diesen Angriffstyp soll hauptsächlich für hyperelliptische Kurven gelten [Ava03]. Takagi und Akishita beschreiben in ihrer Arbeit [AT03], wie man weitere dieser speziellen Fixpunkte auf elliptische Kurven findet um den beschriebenen Angriff in analoger Form durchzuführen. Dies wird in Kapitel 5.3 nochmals aufgegriffen.

#### b) Higher Order DPA

Mehrstufige DPA (engl. Higher Order DPA) stellt eine modifizierte DPA Technik dar. Aufbauend auf den statistischen Messergebnissen von DPA Maßnahmen, welche man in diesem Zusammenhang als Primärstufige DPA (engl. First Order DPA) bezeichnen würde, werden weitere Messverfahren durchgeführt. Dieser Vorgang wird dementsprechend Sekundärstufige DPA (engl. Second Order DPA) genannt [JPS05]. Falls auf den daraus resultierenden Messwerten weitere DPA Maßnahmen ausgeführt werden, verallgemeinert man diese Angriffstechnik zur Mehrstufigen DPA.

Mehrstufige DPA kann zur Sicherheitsanalyse von Systemen herangezogen werden, die man als abgesichert gegen DPA Maßnahmen erachtet. Eine k-stufige DPA Maßnahme nimmt zeitgleich k stichprobenartige Erhebungen, innerhalb der gleichen Menge an Spannungsverläufen und berechnet analog zu einer (Primärstufigen) DPA Maßnahme k Mittelwerte anhand der jeweiligen Selection Funktion. Messerges [Mes00] beschreibt einen solchen Angriff auf ein vermeintlich gegen DPA abgesichertes System, das mittels Datenverschleierung vorgeht. Anders als bei einer einfachen (Primärstufigen) DPA Maßnahme, bei der die gemittelten Messergebnisse keine aussagekräftigen Schlussfolgerungen zuliessen, konnte durch die weiteren Mittelungen der k-stufigen DPA zumindest extremale Kurvenverläufe in Form von peaks detektiert werden. Diese Form der Kryptoanalyse erfordert einen relativ hohen Aufwand und benötigt sehr tiefgreifende Kenntnisse der anzugreifenden Zielplattform [JPS05]. Forschungen im Bereich der Mehrstufigen DPA blieben bisher<sup>1</sup> sehr „empirisch“ [Ava05], d.h. die analytischen Zusammenhänge wurden mathematisch nicht klar beschrieben.

#### c) **Adressbit DPA**

Adressbit DPA stellt eine Angriffstechnik dar, die es nicht unmittelbar auf den Informationswert eines Datums abzielt, sondern auf die Speicheradresse des jeweiligen Datums [JPS05, Mes00]. Die Speicheradresse stellt meistens ein Register auf dem Chip der zugrundeliegenden Recheneinheit dar. Diese Angriffstechnik analysiert die Wechselwirkung zwischen dem verwendeten geheimen Schlüssel und den verwendeten Adressregistern. Eine klassische DPA Maßnahme betrachtet die Wechselwirkung zwischen dem geheimen Schlüssel und dem Informationswert eines verwendeten Register. Für eine bessere Abgrenzung bezeichnet man eine klassische DPA Maßnahme in diesem Zusammenhang auch als Datenbit DPA (engl. Data-bit DPA) [Mes00]. Diese Angriffstechnik wird in Kapitel 4.3 nochmals aufgegriffen.

## 3.2 Elektromagnetische Abstrahlung

Bei der Elektromagnetischen Abstrahlung (engl. electromagnetic radiation, EMA) wird die elektromagnetische Abstrahlung der verwendeten technischen Einheit, hier in der Regel die des Smart Card Chip, gemessen. Gegenmaßnahmen zu EMA werden in der Literatur häufig mit den Gegenmaßnahmen zu Power Analysis zusammengefasst, da die Betrachtungen für Power Analysis, die für EMA in der Regel einschließen [Ava05]. Anders als beim Messen des Strom- bzw. Energieverbrauchs ist bei dieser Angriffstechnik kein direkter Kontakt zur technischen Einheit notwendig, was jedoch ungenaue Messergebnisse nach sich ziehen kann. Weiterhin können physikalische Umgebungsvariablen, wie Temperatur, Oberflächenbeschaffenheit der Smart Card und Leitfähigkeit der verwendeten Drähte Einfluß auf die Messergebnisse haben. Analog zur Power Analysis versucht man eine Korrelation zwischen Stromverbrauch zum einen und Hamminggewicht, Bitse-

---

<sup>1</sup>Stand 2005

quenzen, Operanden oder Speicher- und Adresseinheiten zum anderen festzustellen. Eine typische Gegenmaßnahme gegen EMA, falls sie nicht schon durch eine Gegenmaßnahme gegen DPA bzw. PA abgefangen wurde, ist das Anlegen einer konstanten Spannung über der gesamten technischen Einheit, unabhängig der verwendeten Schlüssel [Ava05]. Auf diese Weise misst ein Angreifer bei jeder Berechnung die gleichen Werte. Jedoch erweist sich diese Gegenmaßnahme als sehr kostspielig. Shamirs Modifikation [Sha00] dieser Gegenmaßnahme sieht zusätzliche Energiequellen in Formen von Kondensatoren, die bei geringerem Energieverbrauch einer EMA Maßnahme entgegenwirken sollen. In Abschnitt 4.5 wird eine konkrete Gegenmaßnahme illustriert.

### 3.3 Timing attacks

Zeitangriffe (engl. Timing attacks) gehören, wie man obiger Auflistung entnehmen kann, zu den ältesten Seitenkanalangriffen und wurden als erstes von Kocher [Ava03] vorgestellt. Timing attacks stellen Chosen plaintext Angriffe dar, d.h. dass ein Angreifer sich zu selbstgewählten Klartexten den Ciphertext erzeugen lässt und, für diese Angriffstechnik sehr wichtig, die für den *gesamten* Berechnungsvorgang benötigte Zeit misst. Ein Zeitangriff ähnelt prinzipiell einer DPA Maßnahme, ist jedoch in der Grundauführung an einigen Stellen komplexer [Ava03]. Der Grund dafür liegt darin, dass ein Zeitangriff die gesamte Zeit der Berechnung misst und nicht nur die Zeitspanne während der Teilberechnung, in die die sensible, kryptographische Berechnung fällt. Da die Algorithmen standardisiert sind, wird der konstante Teil solange abgearbeitet, bis der variable Teil, der von den sensiblen Daten abhängig ist, in der Regel ein geheimer Schlüssel, beginnt. Daraufhin folgt, analog zu einer DPA Maßnahme, eine statistische Analyse. Auch hierbei geht man bitweise vor und trifft Annahmen, ob das folgende Bit gesetzt (das Bit hat den Wert 1) oder ungesetzt (das Bit hat den Wert 0) ist. Bei den folgenden Teilberechnungen sollte man nun Unterschiede in der gemessenen Zeit feststellen, die nur vom zuvor variablen Teil der Berechnung abhängen können. Unter der Annahme, dass für den Rest der Berechnung die gleiche Berechnungszeit gilt, kann man nun auf das Teilgeheimnis schließen. Die Zeitunterschiede in den Teilberechnungen werden dazu genutzt die Messungen zu gruppieren. Für den Fall, dass die Annahmen korrekt gewählt wurden, sollten unterschiedliche Zeiten für die jeweiligen Gruppierungen der Messungen festgestellt werden. Andernfalls sollten keine unterschiedlichen Zeitmessungen festgestellt werden und man korrigiert die falsche Annahme, d.h. aus einer falsch angenommenen 0 wird eine 1 und umgekehrt. Somit erhält man das jeweilige Bit des variablen Teils und setzt diesen Vorgang bitweise fort.

Diese Angriffstechnik kann nur erfolgreich sein, falls kontinuierlich alle aufeinanderfolgende Bits korrekt ermittelt wurden. Sobald ein Bit falsch ist, sind alle darauffolgenden Bits fehlerhaft. In der Folge eines fehlerhaften Bits würden sich die gemessenen Zeitunterschiede zu keiner konkreten Aussagen mehr heranziehen lassen.

Wie schon angedeutet, sind die Gegenmaßnahmen gegen Zeitangriffe, die gleichen, die man gegen DPA bzw. PA Maßnahmen wählt. Spezifische Gegenmaßnahmen für Zeitangriffe, wie beispielsweise das randomisierte Taktverhalten [Ava03] haben sich als untaug-

lich erwiesen, da man den Einfluß dieser Maßnahme durch eine ausreichend große Anzahl an Messungen heraussubtrahieren kann.

### 3.4 Fault attacks

Fehlerattacken<sup>2</sup> (engl. Fault attacks) beschreiben Angriffe, deren hauptsächliche Ziele schwache Implementierungen sind, die es ermöglichen, mittels eines von außen induzierten Fehlers, Rückschlüsse auf sensible Daten zu ziehen. Erste Forschungsergebnisse wurden 1996 von Bellare Forschern [Rel96, BDL97] veröffentlicht.

Folgende Unterscheidungen werden bei Fault Attacks getroffen:

a) **Simple Fault Attacks**

Hier probiert ein Angreifer einen Fehler in die Berechnung zu *induzieren*, um danach Teile des Schlüssels zu rekonstruieren. Als Beispiel dienen häufig RSA Realisierungen, bei denen die Gesamtberechnung in zwei Teilberechnungen gesplittet wird. Bei den Teilberechnungen wird statt mittels des Moduls  $n$ , mit einem der beiden Primfaktoren  $p$  oder  $q$  die Modulodivision ausgeführt (vergl. Abbildung 3.2<sup>3</sup>).

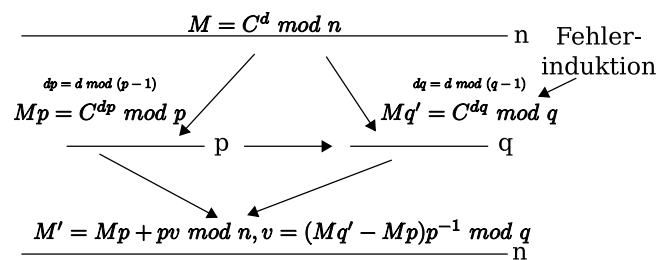


Abbildung 3.2: Fehlerinduktion bei RSA-Berechnung mittels chinesischem Restsatz

Falls es nun einem Angreifer gelingen sollte, wie in Abbildung 3.2 einen der Werte  $p$  oder  $q$  zu manipulieren, kann aus dem resultierendem manipuliertem Ergebnis mittels chinesischem Restsatzes rückwirkend der andere Primfaktor berechnet werden und der geheime Schlüssel wäre bekannt.

b) **Differential Fault Attacks**

Analog zu DPA stellt die Differential Fault Analysis die Modifikation und Erweiterung um statistische Analysetools dar um mögliche Dummy Operationen aufzudecken, die man mittels einer einzelnen Simple Fault attack nicht aufspüren würde. Einem solchem Angriff müssten viele Einzelmessungen vorausgehen.

<sup>2</sup>Fault attacks werden in dieser Arbeit sehr knapp beschrieben, da diese in der Diplomarbeit von Anja Becker am gleichen Fachgebiet ausführlichst behandelt wurden

<sup>3</sup>aus Vorlesung: Effiziente Kryptographie, bei Tsuyoshi Takagi, Technische Universität Darmstadt

### 3 Angriffstypen

Eine typische Gegenmaßnahme für Fault Attacks besteht im regelmäßigen Kontrollieren der benutzten Operanden auf Wohlgeformtheit, das hieße beispielsweise im Fall von Elliptischer Kurvenkryptographie, man kontrolliert, ob die Punkte auf einer vorgegebenen elliptischen Kurve liegen. Sollte einer der Punkte, der in die Berechnung einfließt dies nicht erfüllen, würde das Ergebnis nicht wohlgeformt sein, d.h. der resultierende Punkt wäre nicht mehr auf der vorgegebenen elliptischen Kurve. In diesem Falle würde die Berechnung komplett neu gestartet werden müssen, was den Kostenfaktor für einen Angreifer nach oben treiben würde. Versuche die Integrität der Daten mittels Signaturen [Sha97, Sha99] zu gewährleisten, bleiben im Falle der Elliptischen Kurvenkryptographie bedeutungslos, da hier anders als beim obigen RSA Beispiel das Induzieren eines Fehlers in eine Teilberechnung nicht zwangsläufig zur Enthüllung des Skalars oder eines anderen relevanten Teilergebnisses führen würde. Andere Ansätze bestehen im Einsatz von Fehlerkorrekturcodes zum Aufspüren von Dummy Operationen um diese von den wesentlichen Operationen abzugrenzen. Auch gibt es hardwareseitige Versuche die Stromversorgung für die jeweilige technische Plattform, in der Regel Smart Cards, konstant zu halten (vergl. Abschnitt 4.5) um eine Fehlerinduktion durch absichtlich verursachte Spannungsschwankungen zu vermeiden. Jedoch erweisen sich diese Gegenmaßnahmen ohne Kombination mit anderen Gegenmaßnahmen als sehr unsicher und eignen sich lediglich den Aufwand für einen Angreifer zu erhöhen.

### 3.5 Hardwaregegenmaßnahmen gegen Seitenkanalangriffe

Hardwaregegenmaßnahmen basieren häufig auf dem Prinzip „Sicherheit durch Unklarheit“ (engl. security through obscurity) [Eck04]. D.h. dass konkrete Realisierungen von Hardwareimplementierungen geheim gehalten werden um potenziellen Angreifern das Entdecken von Schwachstellen im System zu erschweren. Diese Gegenmaßnahmen gelten solange als sicher, bis ihre Schwachstellen entdeckt werden, daher sind kaum konkrete Hardwarerealisierungen veröffentlicht. In Abschnitt 4.4ff werden zwei bekannte Hardwaregegenmaßnahmen vorgestellt.

## 4 Gegenmaßnahmen bis 2003

In diesem Kapitel werden konkrete Implementierungen von Gegenmaßnahmen beschrieben, die im vorherigen Kapitel vorgestellt wurden, und die Angriffstechniken, gegen die sie eingesetzt werden können. Es werden 3 algorithmische- und 2 Hardwaregegenmaßnahmen beschrieben.

### 4.1 Randomisierte m-ary Methode [AHLM03]

Die Randomisierung der skalaren Multiplikation stellt eine der häufigsten Gegenmaßnahmen gegen Seitenkanalangriffe in der Elliptischen Kurvenkryptographie dar. Die folgende Gegenmaßnahme basiert auf einer transformierten Darstellung der m-ary (Fenster-) Methode, durch die ein potenzieller Angreifer keinen direkten Zusammenhang des Stromverbrauchs zu einem verwendeten geheimen Schlüssel detektieren soll.

#### 4.1.1 Rekodierung in die m-ary Darstellung

In einer Vorberechnungsphase wird der Skalar  $d$ , je nach gewählter Fenstergröße  $r \in \{1, 2, 3\}$  in eine Folge von m-ary Nummern, im folgenden als ‚*trans*‘ bezeichnet, transferiert, so dass folgendende Beziehung gilt:

- $d = \sum_{i=0}^{k-1} d_i * 2^{ri}$
- $d_i \in \{0, 1, 2, \dots, 2^r - 1\}$
- $r \in \{1, 2, 3\}$

Daraus ergibt sich eine Menge von Paaren, die zum einen aus der  $trans[i]$  Nummer und zum anderen aus der zugehörigen Fenstergröße  $r_i$  bestehen. Eine Zufallszahl  $c$  wird zusätzlich generiert um gegen SPA abzuwehren. Sollte eine Zufallszahl  $P_i=0$  sein, dann ist  $c_i = 0$ , andernfalls ist  $c_i = 1$ . Es gelten folgende Beziehungen:

- $r = \sum_{i=0}^{k-1} r_i * 8^i, r \in \{1, 2, 3\}$
- $trans = \sum_{i=0}^{k-1} trans[i] * 8^i, trans[i] = P_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$
- $c = \sum_{i=0}^{k-1} c_i * 2^i, r \in \{0, 1\}$

### 4.1.2 Prinzip der Gegenmaßnahme

Eine potenzielle Angriffsfläche in der Elliptischen Kurvenkryptographie bieten die unterschiedlichen Formeln für ECADD und ECDBL (vergl. Abschnitt 2.1.2). Hierbei macht sich ein Angreifer zunutze, dass bei zwei identischen Summanden, eine andere Formel verwendet werden muss (ECDBL statt ECADD), und sich dieser Sachverhalt im Energieverbrauch bemerkbar macht. Algorithmus 8 zeigt die randomisierte m-ary Methode, die nicht gegen SPA absichert, da sie in den Schritten 8.11 bis 8.13 konditionelle Sprünge aufweist, in Abhängigkeit der jeweiligen Fenstergröße  $r$ . D.h. diese Schritte werden ausgeführt, falls das jeweilige Bit eine 1 ist, ansonsten werden sie nicht ausgeführt. Algorithmus 9 stellt die modifizierte Version von Algorithmus 8 dar. Hierbei werden in den Schritten 9.11 bis 9.14 von Algorithmus 9 die Berechnungen stets ausgeführt, unabhängig von der jeweiligen Fenstergröße  $r$ . Diese Modifikationen stellen ebenfalls eine Dummy Operation (vergl. Abschnitt 3.1.1.1) dar.

Die Verwendung der Zufallsbits  $c_i$  in Schritt 9.15 stellt eine sogenannte blinding Maßnahme (vergl. Abschnitt 3.1.2.2) dar. Durch den Einbau des Zufallswert  $c$  für jede Berechnung wird erfolgreich gegen DPA Maßnahmen abgewehrt. Weiterhin hat das Bit  $c_i$  die Funktion die Registerfolge zu inkrementieren, so dass am Ende eines Schleifendurchlaufes der richtige der zuvor berechneten 3  $Q[x]$  Werte im Zielregister  $Q[0]$  landet. Der modifizierte Algorithmus 9 durchläuft sequentiell eine DDDA Folge, d.h. es erfolgen in jedem Schleifendurchlauf der Hauptberechnung stets 3 ECDBL und eine ECADD Berechnung.

---

**Algorithmus 8** : Randomisierte m-ary Methode, nur DPA-sicher

---

**Input** :  $P[0]$ ,  $r$ ,  $\text{trans}[k]$   
**Output** :  $Q = dP$

8.1 Vorberechnungsphase  
8.2  $P[1] \leftarrow P[0]$   
8.3 **for**  $P[1] = 2$  **to** 6 **do**  
8.4      $P[i] \leftarrow 2P[i/2]$   
8.5      $P[i + 1] \leftarrow P[i - 1] + P[2]$   
8.6      $i = i + 2$   
8.7 **end**

8.8 Hauptberechnungsphase  
8.9  $Q = \mathbb{O}$ , wobei  $\mathbb{O}$  der Punkt im Unendlichen bei ECC ist  
8.10 **for**  $i = 0$  **to**  $k - 1$  **do**  
8.11      $Q = 2Q$   
8.12     **if**  $r_i \geq 2$  **then**  $Q = 2Q$   
8.13     **if**  $r_i \equiv 3$  **then**  $Q = 2Q$   
8.14     **if**  $\text{trans}[i] \neq 0$  **then**  $Q = Q + P[\text{trans}[i]]$   
8.15 **end**  
8.16 **return**  $Q$

---

---

**Algorithmus 9** : Randomisierte m-ary Methode, SPA- und DPA-sicher

---

**Input** :  $P[0]$ ,  $r$ ,  $\text{trans}[k]$   
**Output** :  $Q[0] = dP$

**9.1** Vorberechnungsphase  
**9.2**  $P[1] \leftarrow P[0]$   
**9.3** **for**  $P[1] = 2$  **to**  $6$  **do**  
**9.4**      $P[i] \leftarrow 2P[i/2]$   
**9.5**      $P[i + 1] \leftarrow P[i - 1] + P[2]$   
**9.6**      $i = i + 2$   
**9.7** **end**

**9.8** Hauptberechnungsphase  
**9.9**  $Q[0] = \mathbb{O}$ , wobei  $\mathbb{O}$  der Punkt im Unendlichen bei ECC ist  
**9.10** **for**  $i = 0$  **to**  $k - 1$  **do**  
**9.11**      $Q[1] = 2Q[0]$   
**9.12**      $Q[2] = 2Q[1]$   
**9.13**      $Q[3] = 2Q[2]$   
**9.14**      $Q[1 + r_1] = Q[r_i] + P[\text{trans}[i]]$   
**9.15**      $Q[0] = Q[c_i + r_i]$   
**9.16** **end**  
**9.17** **return**  $Q[0]$

---

### 4.1.3 Sicherheitsbetrachtung

Die hier vorgestellte Gegenmaßnahme hatte bei ihrer Entwicklung zwei Hauptziele:

- a) Die Unabhängigkeit der sensiblen Daten vom Berechnungsprozeß zu gewährleisten, d.h. es sollte bei der Berechnung keine Korrelation zwischen den verwendeten Rechendaten, hier hauptsächlich der geheime Schlüssel  $d$ , und etwaigen Seitenkanalinformationen, hier hauptsächlich der Stromverbrauch nachvollziehbar sein
- b) Die Randomisierung der benutzten bzw. der zu ermittelnden Rechendaten.

Die 1. Bedingung scheint äquivalent zur Absicherung gegen SPA Maßnahmen zu sein. Da dieser Algorithmus unabhängig vom geheimen Schlüssel  $d$  stets die gleichen Berechnungsschritte (DDDA) in jedem Schleifendurchlauf ausführt, ist dieser Aspekt schon abgedeckt. Ferner wird eine randomisierte Darstellung für den geheimen Schlüssel durchgeführt, die bei jeder Berechnung mit unterschiedlichen Zufallswerten  $r_i$  und  $c_i$  erfolgt, womit auch die 2. Bedingung erfüllt scheint. Es ist nicht davon auszugehen, dass ein Angreifer in der Lage ist, den geheimen Schlüssel zu detektieren UND einen entsprechenden Wert inmitten des Berechnungsprozesses zu erraten mittels einer ‚Selection Function‘ (vergl. Abschnitt 3.1.2), welches die Grundlage einer DPA Maßnahme ist. Jedes Paar aus einer  $\text{trans}[i]$  Zahl und der zugehörigen Fenstergröße  $r$  ist zufällig gewählt.

#### 4.1.4 Effizienz Betrachtung

Tabelle 4.1 vergleicht die hier beschriebenen Gegenmaßnahmen mit anderen binären Methoden und Fenstermethoden in der Standard Weierstraßform bzgl. ihrer Laufzeit. Für eine Bitgröße  $n = 160$  weist die randomisierte m-ary Methode durchschnittlich 69 Paare auf. Die randomisierte m-ary Methode und die Ha-Moon Methoden [hamoon] sind in der Tabelle mit zwei Versionen vertreten. Die jeweils erste sichert nicht gegen DPA ab, die zweite sichert gegen DPA ab. Dies hängt mit den bedingten Sprüngen nach Bitabfragen zusammen (vergl. Abschnitt 4.1.2). Um die unterschiedlichen Anzahlen an ECDBLs und ECADDs besser zu vergleichen wurde (Spalte ‚Laufzeit‘ in Tabelle 4.1) folgende Approximation vorgenommen:

$$\text{ECADD} \approx 0.7 \text{ ECADD}^*$$

d.h. eine ECDBL Operation benötigt lediglich 70% der Laufzeit einer ECADD Operation. Die, hier vorgestellte, Gegenmaßnahme reduziert die Laufzeit auf  $1.33n + 5.1$  ECADD Operationen, was eine enorme Effizienzsteigerung darstellt, wenn man dies beispielsweise mit Corons Methode [Cor99] vergleicht, die lediglich gegen SPA absichert. Im Durchschnitt braucht die randomisierte m-ary Methode 55% weniger ECADDs, jedoch 30% mehr ECDBLs im Vergleich zur Ha-Moon Methode [HM02], die eine randomisierte NAF Implementierung darstellt. Vergleicht man jedoch die Gesamtlaufzeit beider Methoden, fällt die randomisierte m-ary Methode mit 20% effizienter aus. Zieht man nur die DPA abgesicherten Methoden heran, stellt man immerhin noch eine Effizienzsteigerung um 12 % fest. Die randomisierte m-ary Methode ist effizienter und zusätzlich sicherer als die bis 2003 gängigen Gegenmaßnahmen gegen SPA und DPA [Ava05].

Algorithmen	Bitgröße n	SPA-sicher	DPA-sicher	ECADD	ECDBL	Laufzeit*
binäre Methode	160	nein	nein	$0.5n$	$n$	$1.2n$
Coron Methode	160	ja	nein	$n$	$n$	$1.7n$
Ha-Moon Methode	160	nein	ja	$0.5n$	$n+1$	$1.2n+0.7$
		ja	ja	$n+1$	$n+1$	$1.7n+1.7$
randomisierte m-ary Darstellung	69 Paare	nein	ja	$0.33n+3$	$n+3$	$1.03n+5.1$
		ja	ja	$0.43n+3$	$n+3$	$1.33n+5.1$

Tabelle 4.1: Vergleich der Laufzeiten

#### 4.1.5 Abschluß Betrachtung

Die randomisierte m-ary Darstellung transferiert den geheimen Schlüssel  $d$  in eine Folge von trans-Zahlen, die wiederum jeweils mit einer Zufallszahl korrelieren. Dadurch wird eine DPA Maßnahme ausgeschlossen. Ferner wurde durch die sich wiederholende DDDA Struktur innerhalb der Schleife verhindert, dass ein Angreifer mittels einer SPA

Maßnahme Rückschlüsse auf den geheimen Schlüssel erhält. Analog verhält es sich für einen Timing Angriff. Die durchschnittliche Anzahl an ECCADs und ECDBLs bei einer Schlüsselgröße  $n=160$  Bit ist  $1.33n + 5.1$  [Ava05].

## 4.2 Randomisierte Skalardarstellung mittels NAF [HM02]

Analog zur Randomisierten m-ary Methode (vergl. Abschnitt 4.1) versucht die „Randomisierte Skalardarstellung mittels NAF“ gegen DPA Maßnahmen zu sichern, indem in jedem Berechnungsschritt der skalaren Multiplikation eine randomisierte, vorzeichenbehaftete Darstellung des geheimen Schlüssels in NAF gewählt wird. Um zusätzlich gegen SPA abzusichern, wird eine Addition-Subtraktion Operationskette zur Berechnung hinzugefügt.

### 4.2.1 Grundlage der Gegenmaßnahme

Mittels der NAF-Rekodierung läßt sich ein binärer Schlüssel in kompakterer Form wiedergeben. Dann jedoch muss eine skalare Multiplikation in modifizierter Form durchgeführt werden. Algorithmus 10 zeigt eine mögliche Umsetzung.

---

**Algorithmus 10** : Addition-Subtraktions Algorithmus

---

**Input** :  $P, (n + 1)$  bit lange Zahl  $d = \sum_{i=0}^n d_i 2^i, d_i \in \{-1, 0, 1\}$

**Output** :  $Q = dP$

```

10.1  $Q = 0$ 
10.2 for  $i = n - 1$  to  $0$  do
10.3    $Q = 2Q$ 
10.4   if  $d_i = 0$  then
10.5      $Q = Q + P$ 
10.6   end
10.7   if  $d_i = -1$  then
10.8      $Q = Q - P$ 
10.9   end
10.10 end
10.11 return  $Q$ 

```

---

Anders als bei binären Schlüssel, bei denen für jedes gesetzte Bit eine Addition ausgeführt wird, treten bei Schlüsseln in NAF-Darstellung zusätzlich auch Subtraktionen (Schritt 10.8 ) auf, falls eine vorzeichenbehaftete Darstellung (vergl. Abschnitt 2.2.3) eines Schlüssels gewählt wird.

Z.B. kann der gleiche binäre Schlüssel, wie folgt auf zweierlei Arten dargestellt werden:

- $k = 11110 \equiv 2^4 + 2^3 + 2^2 + 2^1 = 30$  in der typischen binären Form
- $d = 1000 - 10 \equiv 2^5 - 2^1 = 30$  rekodiert in der NAF Form

Tabelle 4.2 zeigt, wie eine solche Rekodierung in NAF umgesetzt werden kann. Die Werte  $k_{i+1}$  und  $k_i$  stehen für aufeinanderfolgende Bits im ursprünglichen Schlüssel  $k$ .  $d$  steht für den entsprechenden Schlüssel in NAF. Die Werte für  $c_{i+1}$  und  $c_i$  stehen für einen entstehenden Übertrag an der jeweiligen Bitstelle  $i$ .

Input			Output	
$k_{i+1}$	$k_i$	$c_i$	$c_{i+1}$	$d_i$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0

Tabelle 4.2: NAF Rekodierungstabelle

#### 4.2.2 Umsetzung der Gegenmaßnahme

Um eine DPA Maßnahme abzuwehren, wird nun der Prozess der NAF Rekodierung randomisiert. Hierzu wird ein  $n$ -bit langer Zufallswert  $r = (r_{n-1}r_{n-2}\dots r_0)$  generiert, der für die Randomisierung der NAF Rekodierung verwendet wird, wie in Tabelle 4.3 illustriert. Sei  $k_{i+1}k_i c_1 = 001$  und das Zufallsbit  $r_i = 0$ , dann entspricht dies der NAF Rekodierung  $c_{i+1} d_i = 01$  (vergl. Tabelle 4.3). Falls jedoch das Zufallsbit  $r_i = 1$  ist, käme es zur NAF Rekodierung  $c_{i+1} d_i = 1(-1)$  die jedoch äquivalent zu  $01$  ist. D.h. für unterschiedliche Zufallsbit entstehen bei der NAF Rekodierung für äquivalente Werte unterschiedliche Darstellungen. Dies gilt für die Kombinationen  $k_{i+1} k_i c_1 \in \{001, 010, 101, 110\}$  (vergl. Tabelle 4.3). In allen anderen Kombinationen sind die jeweiligen Darstellungen identisch, unabhängig vom Wert für  $r_i$ .

Das folgende Beispiel soll dies nochmals verdeutlichen.

- Sei  $k = (111011110) = 2^8 + 2^7 + 2^6 + 2^4 + 2^3 + 2^2 + 2^1 \equiv 478$
- $c = (1111111000)$ ,  $r = (101010011)$
- $d = (1000(-1)00(-1)10) = 2^9 - 2^5 - 2^2 + 2^1 \equiv 478$

Mit einer anderen Zufallszahl  $r = (110101001)$  sähe die Berechnung für  $d$ , wie folgt aus

- $c = (1110111100)$ ,  $r = (110101001)$
- $d = (100(-1)1000(-1)0) = 2^9 - 2^6 + 2^5 - 2^1 \equiv 478$

Es ist also möglich für eine bestimmte Bitfolge mehrere verschiedene Alternativdarstellungen angeben zu können, die dem selben Wert entsprechen.

Input				Output		
$k_{i+1}$	$k_i$	$c_i$	$r_i$	$c_{i+1}$	$d_i$	Bemerkung
0	0	0	0	0	0	NAF
0	0	0	1	0	0	NAF
0	0	1	0	0	1	NAF
0	0	1	1	1	-1	AF
0	1	0	0	0	1	NAF
0	1	0	1	1	-1	AF
0	1	1	0	1	0	NAF
0	1	1	1	1	0	NAF
1	0	0	0	0	0	NAF
0	0	0	0	0	0	NAF
0	0	0	0	1	-1	NAF
0	0	0	0	0	1	AF
0	0	0	0	1	-1	NAF
0	0	0	0	0	1	AF
0	0	0	0	1	0	NAF
0	0	0	0	1	0	NAF

Tabelle 4.3: randomisierte, vorzeichenbehaftete NAF Rekodierungstabelle

Der Eintrag AF soll verdeutlichen, dass die NAF Bedingung (= keine benachbarten 1en) nicht mehr erfüllt ist [AHLM03].

### 4.2.3 Analyse der Gegenmaßnahme

Die Absicherung gegen DPA basiert darauf, dass bei jeder Berechnung der verwendete Schlüssel  $d$  in einer alternativen, jedoch äquivalenten, Darstellung vorliegt. Dadurch wird eine statistische Analyse mittels einer Selection Function, dem Kern einer DPA Maßnahme unbrauchbar. Auch bei dieser Gegenmaßnahme, analog zur Randomisierten m-ary Methode, müssen zusätzliche Maßnahmen getroffen werden um gegen SPA abzusichern. Algorithmus 4.3 wurde um eine Dummy Operation erweitert. Bei Algorithmus 4.3 wird unabhängig davon, ob das jeweilige Bit gesetzt ist ( $d_i \in \{1, -1\}$ ) oder nicht gesetzt ist ( $d_i=0$ ) eine Berechnung ausgeführt, so dass dieser Seitenkanal geschlossen ist. Auch hier entsteht eine feste Folge von Multiplikationen, Additionen und Zuweisungen in den Schleifendurchgängen. Durch diese Vorgehensweise wird auch eine Absicherung gegen Timing Angriffe erzielt, da die Dauer der Berechnung durch die Modifikationen in Algorithmus 4.3 keine Korrelation zum verwendeten Schlüssel aufweist. Experimentelle

Versuche kommen zu einer durchschnittlichen Laufzeit von  $1.5n + O(1)$  [AHLM03] bei einer Schlüssellänge  $n$ .

---

**Algorithmus 11** : Addition-Subtraktions Algorithmus, SPA sicher

---

**Input** :  $P$ ,  $(n + 1)$  bit lange Zahl  $d = \sum_{i=0}^n d_i 2^i$ ,  $d_i \in \{-1, 0, 1\}$   
**Output** :  $Q[0] = dP$

```

11.1  $Q[0] = 0$ 
11.2  $P[0] = P, P[1] = P, P[-1] = -P$ 
11.3 for  $i = n - 1$  to  $0$  do
11.4    $Q[0] = 2Q[0]$ 
11.5    $Q[1] = Q[0] + P[d_i]$ 
11.6    $Q[-1] = Q[1]$ 
11.7    $Q[0] = Q[d_i]$ 
11.8 end
11.9 return  $Q$ 

```

---

## 4.3 Randomisierte Adressmethode [IIT03]

Adressbit DPA stellt eine DPA Maßnahme dar, durch die Angreifer versucht, anhand der Wechselwirkung der verwendeten Adressregister und des Stromverbrauchs, den geheimen Schlüssel zu erfahren.

### 4.3.1 Grundlage der Gegenmaßnahme

Adressbit DPA (vergl. Abschnitt 3.1.2.3) basiert auf der Abhängigkeit der Zielspeicherregister auf dem geheimen Schlüssel. So gibt es in nahezu allen Algorithmen eine bitabhängige Zuweisung, die wie folgt aussehen könnte:

$$\begin{aligned} T[0] &\leftarrow T[0] \text{ if } d_i = 0 \\ T[0] &\leftarrow T[1] \text{ if } d_i = 1 \end{aligned}$$

Mit anderen Worten, eine Änderung von  $d_i$  impliziert eine Änderung des jeweiligen Zielspeicherregister. Um diesem Seitenkanal entgegenzutreten, randomisieren wir die Zielspeicherregister mit einer zufallsgenerierten Bitfolge

$$r = r_{n-1}2^{n-1} + r_n2^n + \dots + r_12^1 + r_0, \text{ mit } r_i \in \{0, 1\}$$

Der geheime Schlüssel  $d$  wird mit dem Zufallswert  $r$  wie folgt verknüpft:

$$d_i^* = d_i \oplus r_i$$

Für die  $\oplus$  Verknüpfung können Operationen oder Transformationen gewählt werden, wie beispielsweise die XOR-Funktion oder eine projektive Darstellung. Dieses Prinzip stellt

die „Randomisierte Adressmethode“ dar. Jedoch schützt diese Gegenmaßnahme hauptsächlich gegen Adressbit DPA, so dass eine Kombination mit anderen Gegenmaßnahmen zwingend erforderlich ist. Jedoch hat die hier beschriebene Gegenmaßnahme den Vorteil, dass sie auf reine DPA Maßnahmen mit geringen Modifikationen adaptierbar ist.

### 4.3.2 Beispielanwendungen

Algorithmus 12 zeigt den Add-and-always-Algorithmus und Algorithmus 13 den Montgomery Ladder. Beide Algorithmen sind zunächst in ihrer unveränderten Form angegeben.

---

**Algorithmus 12** : einfache Add-and-double-always Methode

---

**Input** :  $d, P$   
**Output** :  $Q[0] = dP$

12.1  $T[0] = P$   
12.2 **for**  $i = n - 2$  **to**  $0$  **do**  
12.3      $Q[0] = 2Q[0]$   
12.4      $Q[1] = Q[0] + P$   
12.5      $Q[0] = Q[d[i]]$   
12.6 **end**  
12.7 **return**  $Q[0]$

---



---

**Algorithmus 13** : einfacher Montgomery Ladder

---

**Input** :  $d, P$   
**Output** :  $Q[0] = dP$

13.1  $T[0] = P, T[1] = 2P$   
13.2 **for**  $i = n - 2$  **to**  $0$  **do**  
13.3      $Q[2] = 2Q[d[i]]$   
13.4      $Q[1] = Q[0] + Q[1]$   
13.5      $Q[0] = Q[2 - d[i]]$   
13.6      $Q[1] = Q[1 + d[i]]$   
13.7 **end**  
13.8 **return**  $Q[0]$

---

Algorithmus 14 und 15 zeigen die um die Randomisierte Adressmethode modifizierten, und dadurch gegen Adressbit DPA gesicherten, Algorithmen. Bei beiden Algorithmen kann der Zufallswert  $r$  zur Laufzeit (= on the fly) generiert werden, und muss nicht zwingend im voraus erzeugt und im begrenzten Speicher abgelegt werden, was einer Optimierung der Ressourcen gleich kommt. Am Ende der Berechnung muss die inverse Operation zur Verknüpfungsoperation ausgeführt um den Zufallswert zu extrahieren.

---

**Algorithmus 14** : modifizierte Add-and-double-always Methode,  
Adressbit DPA-sicher

---

**Input** :  $d, P$   
**Output** :  $Q[0] = dP$

14.1  $Q[2] = P$   
14.2  $Q[r[n-1]] = Q[2]$   
14.3 **for**  $i = n-2$  **to**  $0$  **do**  
14.4      $Q[r[i+1]] = 2Q[r[i+1]]$   
14.5      $Q[1-r[i+1]] = Q[r[i+1]] + Q[2]$   
14.6      $Q[r[i]] = Q[d[i] \oplus r[i+1]]$   
14.7 **end**  
14.8 **return**  $\text{invR}(Q[r[0]])$

---



---

**Algorithmus 15** : modifizierter Montgomery Ladder, Adressbit DPA-sicher

---

**Input** :  $d, P$   
**Output** :  $Q[0] = dP$

15.1  $Q[r[n-1]] = P$   
15.2  $Q[1-r[i+1]] = 2Q[r[i+1]]$   
15.3 **for**  $i = n-2$  **to**  $0$  **do**  
15.4      $Q[2] = 2(Q[d[i] \oplus r[i+1]])$   
15.5      $Q[1] = Q[0] + Q[1]$   
15.6      $Q[0] = Q[2 - (d[i] \oplus r[i])]$   
15.7      $Q[1] = Q[1 + (d[i] \oplus r[i])]$   
15.8 **end**  
15.9 **return**  $\text{invR}(Q[r[0]])$

---

### 4.3.3 Sicherheitsbetrachtung

Durch diese Modifikationen konnten Algorithmen, die bereits gegen SPA und DPA absicherten, nun zusätzlich gegen Adress Bit DPA abgesichert werden. Bei Algorithmus 14 und 15 werden die Zielspeicherregister durch die Verknüpfung  $d_i \oplus r_i$  bestimmt. Mittels einer Adress Bit DPA kann ein Angreifer zwar detektieren, ob die Zielspeicherregister

$$d_i \oplus r_i = d_i \oplus r_j$$

gleich sind, jedoch kann ein Angreifer nicht feststellen, ob

$$d_i = d_j$$

sind, weil die Werte für  $r_i, r_j$  zufällig gewählt sind. Somit sichert die Erweiterung mittels der Randomisierten Adressierungsmethode gegen Adress Bit DPA ab.

#### 4.3.4 Abschlußbetrachtung

Die Randomisierte Adressmethode schließt einen weiteren Seitenkanal und ist mit geringem Aufwand auf andere Gegenmaßnahmen adaptierbar. Ferner braucht sie keine Hardwareimplementierung und bewirkt durch ihre Modifikationen keine nennenswerten Effizienzeinbußen [IIT03], im Vergleich zu ihren ursprünglichen Algorithmen.

### 4.4 C-MOS Logik als Gegenmaßnahme gegen DPA [TAV02]

Die bisher beschriebenen Gegenmaßnahmen hatten bisher alle einen algorithmischen Ansatz. Um gegen Seitenkanalangriffe abzusichern wurden auch Hardwarelösungen angestrebt. Eine dieser Ansätze wird mittels C-MOS Logikbausteinen umgesetzt. Ziel ist es, durch eine dynamische Anordnung von C-MOS Transistoren, während der sensiblen Rechenoperation eine Korrelation zwischen dem Energieverbrauch und dem geheimen Schlüssel zu vermeiden.

#### 4.4.1 Grundlage der Gegenmaßnahme

Von den 4 möglichen Zuständen eines C-MOS Bauteils (vergl. Abbildung 4.1) stellen lediglich die 0-1 und die 1-0 Transition akzeptable Zustände dar. Bei diesen beiden Zuständen redet man von einer logischen 1 bzw. einer logischen 0 [Kat93], da hier entweder ein Stromfluss von Masse zum Schalter fließt, im Falle der 0-1 Transition, bzw. ein Stromfluß vom Schalter zur Erde, im Falle der 1-0 Transition. In den beiden anderen Zuständen erfolgt kein signifikant messbarer Stromfluss, daher sind diese beiden für folgende Betrachtungen vernachlässigbar.

Dieser Aufbau von C-MOS Bauteilen ermöglicht es einem Angreifer zu unterscheiden, ob zu einem gegebenen Zeitpunkt eine logische 1, im Falle der 0-1 Transition, oder eine logische 0, im Falle der 1-0 Transition, übertragen wird. Hier versucht die folgende Gegenmaßnahme entgegenzusteuern, indem zum einen

- mittels veränderter Anordnung von Logikbausteinen, nach außen hin eine stets konstante Spannung suggeriert wird
- und weiterhin am Ausgangsgatter stets eine Spannung anliegt, so dass es von außen nicht mehr möglich ist, zwischen einer logischen 1 und einer logischen 0 als Eingangssignal zu unterscheiden

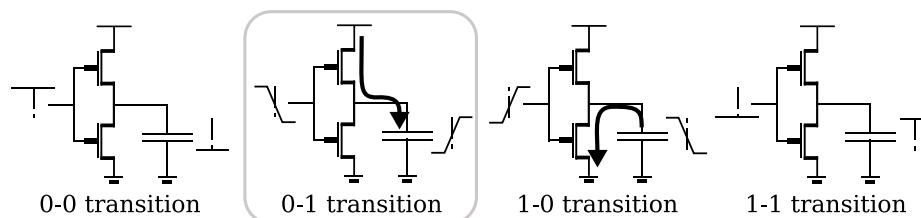


Abbildung 4.1: CMOS-Anordnungen

#### 4.4.2 Funktionsweise der C-MOS Logikbausteine

Um die beiden gewünschten Punkte umzusetzen, wird bei der Anordnung der logischen Bausteine eine Kombination einer „Differenzierten Logik“ und einer „Dynamischen Logik“ gewählt, die mit SABL bezeichnet wird [TAV02]. Abbildung 4.2 zeigt die Umsetzungen dieser beiden Logiken in vereinfachter Form, da lediglich die Eingangs- und Ausgangssignale, Masse und Grund angezeigt werden. Die Differenzierte Logik maskiert die Eingangssignale, d.h. unabhängig vom Eingangssignal, wird ein Energieverbrauch gemessen, falls über genau einen Ausgangsknoten Strom fließt. Dies ist genau dann der Fall bei einer 0-1 und einer 1-0 Transition eines C-MOS Transistors. In den beiden anderen Fällen sperrt das Ausgangsgatter. Somit wurde eine Ununterscheidbarkeit zwischen den beiden leitenden Fällen bewirkt, jedoch können diese beiden Fälle von den beiden nicht leitenden Fällen differenziert werden. Mittels der Dynamischen Logik wird erreicht, dass ein Energieverbrauch gemessen wird, wenn sich am betreffenden C-MOS Teil eine Spannung aufgrund der Kondensatorfähigkeit aufbaut. Dadurch wird eine Ununterscheidbarkeit, zwischen der 0-0 und der 1-0 Transition einerseits und der 1-1 und der 0-1 Transition andererseits, erreicht. Kombiniert man diese beiden Logiken, hat man die beiden gewünschten Punkte erreicht. Hier muss jedoch angemerkt werden, dass nun auch weiterhin unterschiedliche Energieverbräuche gemessen werden können, die je komplexer die Anordnung der logischen Bauteile ist, sich vergrößert. Diesem Umstand wird entgegengetreten, indem stets die gleiche Spannung abgeführt wird, sowohl am Ausgangsgatter als auch bei den einzelnen Gattern in den Zwischenstufen.

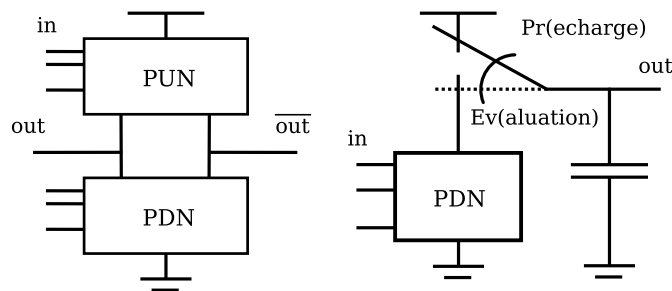


Abbildung 4.2: Differenzierte (links) und Dynamische(rechts) Logik

Für konkrete Umsetzungen von Kombinationen Differenzierter und Dynamischer Logiken sei auf [Rab] und [BNL] verwiesen.

#### 4.4.3 Experimentelle Ergebnisse

Abbildung 4.3 zeigt die Schaltbilder einer herkömmlichen Hardwareanordnung mittels C-MOS Bauteilen (links) und der SABL-Anordnung (rechts), die die zuvor beschriebenen Modifikationen beinhaltet. Um die beiden Versionen bzgl. ihres Energieverbrauchs zu untersuchen, wurde folgende Normierungsenergieabweichungsdifferenz (NED) [TAV02] definiert:

#### 4 Gegenmaßnahmen bis 2003

$$\text{NED} = \frac{\text{Max}(\text{energy/cycle}) - \text{min}(\text{energy/cycle})}{\text{Max}(\text{energy/cycle})}$$

die einen Indikator darstellen soll, wie viele Seitenkanalinformationen durch den Energieverbrauch preisgegeben werden können. NED ist ein Wert zwischen 0 und 1. Je kleiner der Wert für NED umso weniger Seitenkanalinformationen werden preisgegeben, und ein möglicher Angriff wird extrem erschwert.

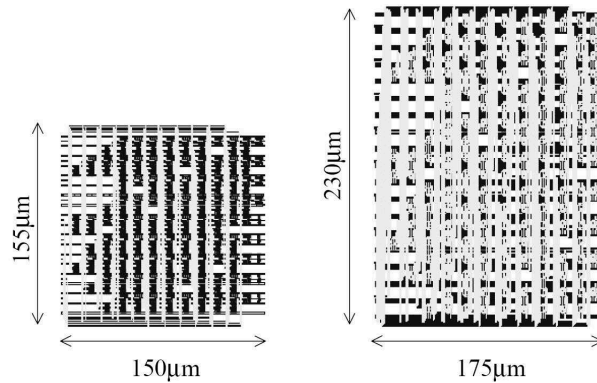


Abbildung 4.3: Standard (links), modifizierte Anordnung (rechts) [TAV02]

Tabelle 4.4 vergleicht die NED Werte der S9 Implementierung, als auch von Standardgattern, wie beispielsweise NAND und XOR, aus denen u.a. die modifizierte Hardwareanordnung besteht. Die Hauptinformation besteht jedoch darin, dass die modifizierte Implementierung im Vergleich zu ihrer herkömmlichen CMOS-Implementierung nur noch einen sehr kleinen NED Wert aufweist. Dies gilt sowohl für die Standardgatter als auch für eine vollständige Implementierung, wie der S9-Anordnung. Letztere weist nach der modifizierten Hardwareanordnung einen NED Wert auf, der nur 3% des ursprünglichen Wertes beträgt.

	INV	NAND	XOR	FF	S9-box
	NED	NED	NED	NED	NED
CMOS	1.000	1.000	1.000	0.821	1.000
SABL	0.009	0.032	0.016	0.020	0.032

Tabelle 4.4: Vergleich der NED-Werte

#### 4.4.4 Abschlußbetrachtung

Die Umsetzung einer SABL-Anordnung, die eine Kombination einer Dynamischen und Differenzierten Logik darstellt, verbraucht einen konstanten Wert pro Berechnung und dies unabhängig von benutzten Eingabewerten. Sie leitet dynamisch den Stromfluß zwischen Anfangs- und Endgatter, so dass zwischen diesen beiden Gatter keine unterschied-

lichen Spannung gemessen werden kann. Diese Hardwaregegenmaßnahme stellt eine mögliche Abwehrtechnik gegen SPA Maßnahmen dar. Da diese Realisierung speziell bei komplexeren Anordnungen eine nur sehr schwache Sicherheitsmaßnahme darstellt, sollte sie lediglich als zusätzliche Maßnahme verwendet werden.

### 4.5 Einsatz von Kondensatoren zur Verschleierung des Energieverbrauchs [CAM<sup>+</sup>02]

Falls ein Angreifer genaue Kenntnisse der zugrundeliegenden Hardware besitzt und in der Lage ist, jeden (Maschinen-)Rechenschritt nachzuvollziehen, kann es ihm gelingen anhand des Energieverbrauchs zu erkennen, ob gerade ein gesetztes Bit oder ein ungesetztes Bit an der jeweiligen Recheneinheit durchlaufen ist. Eine mögliche Hardwaregegenmaßnahme besteht darin, durch den Gebrauch zweier Kondensatoren den Energieverbrauch für Beobachter von außen stets konstant erscheinen zu lassen.

#### 4.5.1 Grundlage der Gegenmaßnahme

Um diese Gegenmaßnahme zu realisieren werden zwei Kondensatoren verwendet, die die Aufgabe haben abwechselnd die Energieversorgung der verwendeten Hardware, in der Regel handelt es sich um Smart Cards, zu gewährleisten. Die beiden Kondensatoren befinden sich in maximal 2 zueinander gegensätzlichen Zuständen. Während einer der beiden Kondensatoren sich mit der externen Energieversorgung lädt, entlädt sich der andere Kondensator, indem er den Smart Card Chip mit Energie versorgt. Anschließend wechseln die beiden Kondensatoren in den jeweils anderen Zustand. Das Verhalten der Kondensatoren wird durch eine simple Schaltkontrolleinheit (engl. switch control unit) bestimmt, die mittels logischer Bausteine, wie in Abbildung 4.4 [CAM<sup>+</sup>02] implementiert sein könnte. Für diese Umsetzung werden 4 Transistoren hinzugefügt um die gewünschten Zustände zu realisieren. Die empfohlene Abfolge der Zustandsübergänge, sieht wie folgt aus

- a) der 1.Kondensator ist nicht verbunden mit der externen Energieversorgung
- b) der 1.Kondensator ist verbunden mit dem Smart Card Chip
- c) der 2.Kondensator ist nicht verbunden mit dem Smart Card Chip
- d) der 2.Kondensator ist verbunden mit der externen Energieversorgung

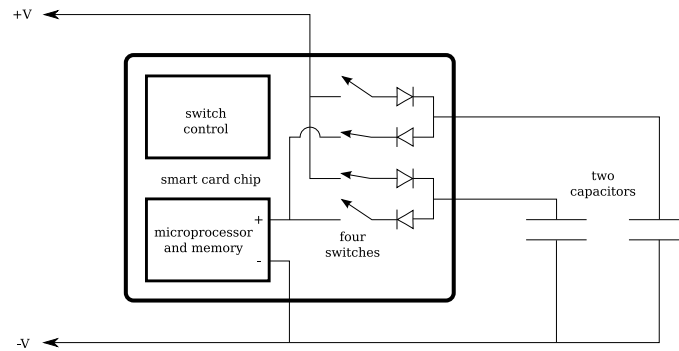


Abbildung 4.4: Smart Card mit konstantem Energieverbrauch

Auf diese Art wird erreicht, dass stets ein Kondensator den Smart Card Chip mit Energie versorgt, jedoch die externe Energieversorgung nie eine direkte Verbindung zum Smart Card Chip hat. Folglich kann ein Angreifer keine direkte Wechselwirkung zwischen dem Energieverbrauch während einer Rechenoperation und dem verwendeten geheimen Schlüssel erkennen. Der Energieverbrauch über die Zeit ist in Abbildung 4.5 illustriert.

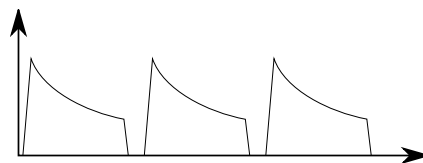


Abbildung 4.5: Energieverbrauch mit dazwischengeschalteten Kondensatoren

#### 4.5.2 Schwachstelle

Ferner muß bedacht werden, dass Seitenkanalinformationen nicht nur über die Stromverbrauchslinien verhindert werden müssen, sondern auch über I/O Sequenzen des Smart Card Chips, der in serieller Weise Daten empfängt und abschickt. Diese Schwachstelle, die in vielen Hardwaregegenmaßnahmen oft gänzlich ignoriert wurde [CAM<sup>+</sup>02] kann bei der hier vorgestellten Gegenmaßnahme verhindert werden, indem I/O Vorgänge während sensibler kryptographischer Prozesse untersagt werden.

#### 4.5.3 Vergleich zu Vorgänger

Die hier vorgestellte Gegenmaßnahme mittels Kondensatoren stellt eine Modifikation bzgl. vorheriger Gegenmaßnahmen mittels einfacher Batterien dar. Jedoch hat der Ansatz mit Kondensatoren den Vorteil, dass

- Kondensatoren kleiner sind und dadurch „einfacher“ auf dem Chip zu platzieren
- Kondensatoren günstiger in der Herstellung sind

- Kondensatoren sich beliebig oft aufladen lassen, während Batterien begrenzte Ladezyklen haben
- Kondensatoren keinen Memory Effekt haben, so daß sie wiederaufgeladen werden können ohne vollständig entleert worden zu sein
- Kondensatoren sich sekundenschnell aufladen können und somit auch spontaner Gebrauch möglich ist.

##### 4.5.4 Abschlußbetrachtung

Die Energieverschleierung mittels Kondensatoren stellt eine Hardwaregegenmaßnahme dar um einen Angriff zusätzlich zu erschweren, sollte jedoch nie als einzige Gegenmaßnahme verwendet werden - wie auch alle anderen Hardwaregegenmaßnahmen. Schließlich kann ein Angreifer, falls dieser die Möglichkeit besitzt die Kondensatoren von der Smart Card zu entfernen und diese gegebenenfalls mit anderen Komponenten zu ersetzen, dieser Gegenmaßnahme die Grundlage nehmen.

## 5 Neue Gegenmaßnahmen ab 2003

Dieses Kapitel beschreibt die erweiterten und modifizierten Gegenmaßnahmen nach 2003, die weitestgehend auf den in Abschnitt 4 vorgestellten Abwehrtechniken basieren.

### 5.1 Ununterscheidbarer Code für ECADD und ECDBL [Wal04]

Da die Ausführung der Operationen für ECADD und ECDBL bei elliptischen Kurven unterschiedliche Spannungsverläufe haben und dadurch eine potenzielle Angriffsfläche bieten, wird das Vereinheitlichen dieser beiden Operationen (engl. Unified Code for ECADD and ECDBL) als eine mögliche Gegenmaßnahme betrachtet. Das Vereinheitlichen der Operationen ähnelt dem Vorgang der Dummy Operationen (vergl. Abschnitt 3.1.1.1).

#### 5.1.1 Grundlage der Gegenmaßnahme

Ausgehend vom „quadriere und multipliziere“-Algorithmus (engl. square and multiply) bei der modularen Exponentiation bzw. „verdopple und addiere“-Algorithmus (engl. add-and-double-always Methode) bei elliptischen Kurven, werden bei der Punktmultiplikation zwei Aspekte im besonderen beachtet. Jedes gesetzte Bit hat eine Verdopplung mit einer anschließenden Addition zur Folge (Schritt 16.4 und 16.3 in Algorithmus 16). Andernfalls folgen zwei Verdopplungen aufeinander. Dieser Informationsfluß stellt einen Seitenkanal dar, den ein Angreifer mittels einer SPA Maßnahme nutzen könnte.

---

**Algorithmus 16** : verdopple und addiere-Algorithmus, MSB

---

**Input** :  $P, d = d_{n-1}d_{n-2} \dots d_0$ , wobei  $d_{n-1} = 1$

**Output** :  $Q = dP$

```
16.1  $Q \leftarrow P$ 
16.2 for  $i = n - 2$  to  $0$  do
16.3   |  $Q \leftarrow 2Q$  ;
16.4   | if  $d_i = 1$  then  $Q \leftarrow Q + P$  ;
16.5 end
16.6 return  $Q$ 
```

---

### 5.1.2 Vereinheitlichung der Operationen

Brier und Joye [BJ] liefern Formeln, die die klassischen Formeln für die Addition (ECADD) und Verdopplung (ECDBL) in elliptischen Kurven vereinheitlichen. D.h. von außen betrachtet, kann ein Angreifer mittels einfacher SPA Maßnahme nicht erkennen, welcher der beiden Operationen stattfindet. Sie unterscheiden die Fälle für projektive und affine Koordinatenangaben.

In der Formel für affine Koordinaten kann der Nenner keinen Nullwert haben, so dass keine zusätzliche Formel für die Punktmultiplikation erforderlich ist.

Für den Fall, dass für die Darstellung der Punkte auf einer elliptischen Kurve projektive Koordinatenpunkte  $P = (x, y, c)$  in Weierstraß Form

$$y^2z = x^3 + axz^2 + bz^3$$

verwendet werden, wobei die Charakteristik der Kurve  $\neq 2$  oder  $3$  ist, berechnet sich die Summe  $P_3 = P_1 + P_2$  nach Brier und Joye, wie folgt

$$\begin{aligned} x_3 &= 2fw; \\ y_3 &= r(g - 2w) - l^2; \\ z_3 &= 2f^3; \end{aligned} \tag{5.1}$$

mit den Koeffizientenbelegungen

$$\begin{aligned} u_1 &= x_1z_2, & u_2 &= x_2z_1, & t &= u_1 + u_2; \\ s_1 &= y_1z_2, & s_2 &= y_2z_1, & m &= s_1 + s_2; \\ z &= z_1z_2, & f &= zm, & l &= mf, g = tl; \\ r &= t^2 - u_1u_2 + az^2, & w &= r^2 - g. \end{aligned} \tag{5.2}$$

Für diese Formeln werden 18 Multiplikationen benötigt, 5 Multiplikationen für die Gleichungen aus (5.1), 13 Multiplikationen für die Gleichungen aus (5.2). Ohne Beschränkung der Allgemeinheit kann angenommen werden, dass im Falle eines Angriffs für die Berechnung von  $P_3$  die Gleichungen aus (5.2) in der angegebenen Reihenfolge berechnet werden [Wal04]. Es werden jeweils die rechten Seiten der Gleichungen berechnet, bevor die Werte an die Gleichungen im aus (5.1) weitergegeben werden.

### 5.1.3 Abschlußbetrachtung

Es hat sich in der Folge schon früh gezeigt, dass das Vereinheitlichen des Codes keine sichere Gegenmaßnahme darstellt, da beispielsweise viele Hardwareimplementierungen noch genügend Seitenkanalinformationen ausschütten [Wal04] (vergl. auch Abschnitt 3.1.2.3). Einzig gegen SPA Maßnahmen kann das Vereinheitlichen des Codes eingesetzt werden und sollte daher mit anderen Gegenmaßnahmen kombiniert werden. Das Vereinheitlichen des Codes wurde auch auf bereits bekannte Gegenmaßnahmen, wie der Montgomery-Multiplikation [EW93] aufgesetzt, jedoch läßt sich auch hier die für die Montgomery-Multiplikation charakteristische, abschließende Subtraktion über hardware bedingte Seitenkanäle detektieren [Wal04].

Man kann festhalten, dass das Vereinheitlichen des Codes eine Sicherheitssteigerung zu Corons Gegenmaßnahme [Cor99] oder auch den blinding Methoden (vergl. Abschnitt 3.1.2.2) darstellt, jedoch ebenfalls keine ausreichende Sicherheit gewährleisten kann [Wal04].

## 5.2 Parametrisierung in elliptischen Kurven [Ols04]

Die Parametrisierung von bestimmten Punkten in elliptischen Kurven stellt eine Maßnahme dar, die verhindern soll, dass bei sensiblen Operationen, wie beispielsweise der Punktmultiplikation  $nP$  Seitenkanalinformationen nach außen dringen können.

### 5.2.1 Grundlagen der Gegenmaßnahme

Bei der Addition von zwei Punkten  $P + Q$  auf einer elliptischen Kurve  $E$  kann man detektieren, ob es sich um die gleichen Punkte ( $P = Q$ ) oder um ungleiche Punkte ( $P \neq Q$ ) handelt. Die Unterscheidung kann mittels Energieverbrauchsmessung (vergl. EMA) oder Zeitmessung (vergl. TA) festgestellt werden.

Ein Lösungsvorschlag besteht in der Parametrisierung der Punkte einer elliptischen Kurve. Nach einer Parametrisierung existieren für jeden Punkt  $P_x \in E$  eine Menge alternativer Darstellungen, die mit

$$\mathbb{P}_x = \{\text{Menge aller Punkte, die isomorph zu } P_x \text{ sind}\}$$

zusammengefasst werden. Ziel dieser Parametrisierung ist es, dass eine Unterscheidung ob ( $P = Q$ ) oder ( $P \neq Q$ ), keine Rolle mehr spielt und somit dieser Seitenkanal geschlossen werden kann.

**Definition 5.2.1** Sei  $\mathbb{E}$  eine elliptische Kurve mit dem Identitätselement  $e$ . Dann existiert für jedes Paar  $(M, \mathbb{E})$ , bestehend aus einem beliebigen Punkt  $M \in \mathbb{E}$ , und der elliptischen Kurve  $\mathbb{E}$  eine isomorphe Kurve  $C_M$ . Diese Kurve wird als "Gewichtete Projektive Biquadratische Darstellung" (engl. *weighted projective quartic curve*) bezeichnet.

Diese Gewichtete Projektive Biquadratische Darstellung erfüllt die gewünschte Anforderung, wie im folgenden noch gezeigt wird. Der Punkt  $M$  und damit einhergehend die Kurve  $C_M$  können mit sehr geringem Aufwand dahingehend verändert werden, so dass eine Parametrisierung für jede Berechnung  $nP$  in Betracht gezogen werden kann.

### 5.2.2 Umsetzung der Gegenmaßnahme

Sei  $k$  ein Feld mit einer Charakteristik  $> 3$ . Sei die elliptische Kurve  $E \subseteq \mathbb{P}^2$ , definiert durch die homogene Gleichung

$$\bullet Y^2Z = X^3 + a_4XZ^2 + a_6Z^3$$

mit neutralem Element  $e = (0,1,0)$ . Sei  $M \neq e$  ein  $k$ -rationaler Punkt auf  $E$  mit Koordinaten  $M = (\alpha, \beta, 1)$ . Weiterhin definiert man die Konstanten  $c_i \in k$ , wie folgt

## 5 Neue Gegenmaßnahmen ab 2003

- $c_2 = -(3\alpha/2)$
- $c_3 = -\beta$
- $c_4 = -(4a_4 + 3\alpha^2)/16$

Sei  $D_M$  eine affine, biquadratische Kurve, beschrieben durch

- $W_2 = R(S) = S^4 + c_2S^2 + c_3S + c_4 = S^4 - (3\alpha/2)S^2 - \beta S - (4a_4 + 3\alpha^2)/16$

welche den affinen Teil der Kurve darstellt, die durch die elliptische Kurve  $E$  und dem Punkt  $M \neq e$  charakterisiert wird.

Sei nun umgekehrt eine biquadratische Gleichung durch ihren affinen Anteil, wie folgt gegeben durch

- $W_2 = R(S) = S^4 + c_2S^2 + c_3S + c_4$

mit  $c_i \in k$ , so dass  $R(S)$  keine multiplen Wurzeln hat. Dann gilt

- $a_4 = -[(c_2^2/3) + 4c_4]$
- $a_6 = [2(c_2/3)^3 - 8(c_2c_4/3) + c_3^2]$
- $\alpha = -2c_2/3$
- $\beta = -c_3$

Und die Gleichung

- $Y^2Z = X^3 + a_4XZ^2 + a_6Z^3$

definiert eine elliptische Kurve  $E$  mit dem Punkt  $M \neq e$  auf  $E$  mit den Koordinaten  $M = (\alpha, \beta, 1)$ . Dann existiert ein Isomorphismus

$$I : (E - (M, e)) \rightarrow D_M$$

der durch folgende Entsprechungen gegeben ist

- $S = (Y + \beta)/2(X - \alpha)$
- $W = (X/2) + (\alpha/4) - Y + \beta^2/4(X - \alpha)^2$
- $X = 2W + 2S^2 - (\alpha/2)$
- $Y = 4SW + 4S^3 - 3\alpha S - \beta$

Für die genaue Herleitung dieser klassischen Formeln sei auf Fricke und Reid verwiesen [Ols04, Fri22, BJ].

### 5.2.3 Gruppenaxiome auf $C_M$

Da das Hauptziel der Parametrisierung in elliptischen Kurven die Ununterscheidbarkeit von 2 Punkten ist, muss in der isomorphen Struktur eine einheitliche bzw. nicht unterscheidbare Operation für ECADD und ECDBL existieren. D.h. es darf für einen potentiellen Angreifer nicht erkennbar sein, ob gerade zwei identische Punkte einer elliptischen Kurve ( $P = Q$ ) oder zwei ungleiche Punkte ( $P \neq Q$ ) zusammenaddiert werden. Olson [Ols04] illustriert in seiner ausführlichen Berechnung, wie zwei Punkte  $P_1, P_2 \in E$ , nach ihrer isomorphen Transformation  $\phi : E \rightarrow C_M$  in die Punkte  $Q_1, Q_2 \in C_M$  abgebildet werden, die die gewünschte Eigenschaft erfüllen. Er überführt die Gleichung für ECADD (falls  $Q_1 \neq Q_2$  gilt) durch eine Äquivalenzumformung in die Gleichung für ECDBL (falls  $Q_1 = Q_2$  gilt).

Der hierzu angewendete Algorithmus [Ols04] startet mit der Belegung von den Werten für  $e_i$ , aus denen anschließend die Werte für  $c_i, S_i$  und  $W_i$  resultieren. Zur Komplexitätsabschätzung werden die benötigten Operationen aufaddiert und gewichtet. Für diesen Algorithmus werden 31 Multiplikationen ausgeführt. Einen effizienteren Algorithmus stellen Brier und Joye [BJ] vor, der mit 17 Multiplikationen plus einer Multiplikation mit einer Konstanten auskommt.

### 5.2.4 Abschlußbetrachtung

Die Parametrisierung in elliptischen Kurven sichert gegen SPA und DPA. Weiterhin gilt, dass sie auf alle elliptische Kurven adaptierbar ist und dadurch eine breite und langfristige Anwendung möglich ist. Für jede elliptische Kurve existiert eine große Anzahl an alternativen Darstellungen, die man mittels Parametrisierung erhält und dies bei nahezu vernachlässigbaren Kosten.

## 5.3 RPA - ZPA Gegenmaßnahmen

Verbesserte Gegenmaßnahmen gegen DPA Maßnahmen, haben neue Angriffe und dadurch wiederum neue Gegenmaßnahmen hervorgebracht. Im folgenden werden zwei erweiterte DPA Maßnahmen vorgestellt, die ebenfalls Beispiele für Sekundarstufige bzw. Mehrstufige DPA (vergl. Abschnitt 3.1.2.3) sind.

### 5.3.1 Refined Power Analysis

Verfeinerte DPA (engl. Refined Power Analysis, kurz RPA) stellt eine modifizierte DPA Maßnahme im Bereich der elliptischen Kurven dar. Das Hauptaugenmerk liegt auf vorhandenen Nullwerten in einem der beiden Koordinatenpunkten, z.B.

$$P_1 = (0,y) \text{ oder } P_2 = (x,0).$$

Ein Angreifer macht sich hierbei zunutze, dass das Vorhandensein eines Nullwertes anhand des Energieverbrauchs eindeutig messbar ist. Auch nach einer skalaren Randomisierung der Punkte mittels eines Zufallswerts  $r$ , bleiben die Nullwerte fix. In der projektiven Darstellung sehen die Koordinaten, wie folgt aus

$$P_1^* = (0, ry, r) \text{ oder } P_2^* = (rx, 0, r)$$

Goubins Angriff (vergl. Abschnitt 3.1.2.3) stellt eine mögliche RPA Maßnahme dar. Bei diesem Angriff führt ein Angreifer mehrere Punkte  $P_i$  in die Berechnung ein, für die entweder  $x = 0$  oder  $y = 0$  gilt, bei gleichbleibendem Geheimnis  $d$ . Da nun wiederholt  $s$  Punkte  $P_i$ ,  $i \in \{1, \dots, s\}$  berechnet werden, kann man wiederum mittels einer (Sekundärstufigen) DPA Maßnahme auf die geheime Information schließen.

### 5.3.2 Zero Value Point Attack

Falls sich die Analyse nicht nur auf Nullwerte in Koordinaten, sondern auch auf ebensolchen in Registern, die zur Speicherung der Zwischenwerte eingesetzt werden, erstreckt, dann spricht man auch allgemein von einer ZPA Maßnahme (Zero-Value Point Attack) [AT03]. Hierbei werden speziell die Nullwertregister analysiert, die weder mit Corons 3.ter noch von der Joye-Tymen (vergl. Abschnitt 5.1) Gegenmaßnahme randomisiert wurden.

### 5.3.3 Gegenmaßnahmen

Das grundlegende Verfahren besteht aus einer einfachen, zusätzlichen Addition eines Zufallswerts  $R$  zur Hauptberechnung  $dP$ , so daß

$$dP + R$$

berechnet wird, gefolgt von einer abschließenden Subtraktion

$$(dp + R) - R = dP$$

um am Ende das gewünschte Ergebnis zu erhalten. Algorithmus 17 zeigt die Umsetzung dieses Verfahrens, das eine binäre Erweiterung mittels eines Zufallswerts (engl. Binary Expansion with RIP, verkürzt BRIP) darstellt.

---

#### Algorithmus 17 : BRIP-Algorithmus

---

```

Input   :  $d, P$ 
Output :  $dP$ 

17.1  $R \leftarrow$  Zufallswertberechnung
17.2  $T_0 = R, T_1 = -R, T_2 = P - R$ 
17.3 for  $i = n - 1$  to  $0$  do
17.4    $T_0 = 2T_0$ 
17.5   if  $d_i = 0$  then
17.6      $T_0 = T_0 + T_1$ 
17.7   else
17.8      $T_0 = T_0 + T_2$ 
17.9   end
17.10 end
17.11 return  $T_0 + T_1$ 

```

---

Dieses einfache BRIP Verfahren sichert gegen DPA, RPA und sogar gegen ZPA Maßnahmen, da bei jeder Ausführung die Werte von  $T_0, T_1$  und  $T_2$  von einem neu generiertem Zufallswert ermittelt werden. Dadurch wird eine statistische Analyse über mehrere Berechnungen hinweg unwirksam gemacht, da - statistisch gesehen - eine Nullkoordinate bzw. ein Nullwertregister nicht in jeder dieser Berechnungen vorliegen kann.

Der BRIP Algorithmus benötigt  $n$  ECDBLs,  $n$  ECADDs und 3 Register- bzw. Speicherplätze.

**Verallgemeinerte Gegenmaßnahme** Der BRIP Algorithmus kann zu einem effizienterem Algorithmus verallgemeinert werden mit Hilfe einer vorberechneten Tabelle, soweit dieses Verfahren mittels MSB Berechnung erfolgt. Mittels LSB Berechnung ist es nicht möglich ein allgemeines Verfahren anzugeben, das hinsichtlich der Laufzeit eine Optimierung gegenüber dem einfachen BRIP Algorithmus aufweist [MMM04]. Die Hauptidee ist den Exponenten in zwei Teile zu splitten und die Hauptberechnung durch zwei kleinere Nebenberechnung zu ersetzen. Es stehen für vorberechnete Werte zwei Realisierungen zur Auswahl, zum einen die Fenster Methoden, zum anderen der erweiterte Binär-Algorithmus (engl. extended binary algorithm). Der ursprüngliche Gebrauch des erweiterten Binäralgorithmus bestand darin, einen Exponenten aus zwei Teilkomponenten  $aP + bQ$ , wie folgt zu berechnen.

Sei  $d = \sum_{i=0}^{n-1} d_i s^i$  und  $n$  gerade

- a) Teile  $d$  in zwei Komponenten  $b$  und  $a$  ( $d = b \parallel^1 a$ ), mit  $b = (d_{n-1} \dots d_{\frac{n}{2}})$  und  $a = (d_{\frac{n}{2}-1} \dots d_0)$
- b) Berechne  $Q = 2^{\frac{n}{2}} P$
- c) Erstelle Tabelle mit vorberechneten Werten  $\{P, Q, P + Q\}$
- d) Berechne  $aP + bQ$  mittels der erweiterten Binärmethode unter Gebrauch der vorberechneten Tabellenwerte

**Erweiterter Binär-Algorithmus** Algorithmus 18 veranschaulicht den erweiterten Binär-Algorithmus (EBRIP), und zwar für den vereinfachten Fall, dass der geheime Schlüssel  $d$  in 2 Teile gesplittet und die Binärlänge  $n$  gerade ist.

Unter der Annahme, dass der Wert  $R$  ein reiner Zufallswert sei, sichert der EBRIP Algorithmus analog dem BRIP Algorithmus gegen DPA, RPA und ZPA.

EBRIP benötigt  $n'$  ECDBLs und  $(\frac{n'}{t} + 2^t)$  ECADDs. Zusätzlich wird eine Hilfsvariable gebraucht, so dass in der Summe  $2^t + 1$  Speichereinheiten benötigt werden.

---

<sup>1</sup> splittet  $d$  so, dass in  $b$  die höherwertigen Bits und in  $a$  die niederwertigen Bits sind

**Algorithmus 18** : EBRIP-Algorithmus

---

**Input** :  $d, P$   
**Output** :  $dP$

**18.1**  $R \leftarrow$  Zufallswertberechnung  
**18.2** Setze  $d = b \parallel a$ , so dass  $b = (b_{(\frac{n}{2}-1} \dots b_0)_2 = (d_{n-1} \dots d_{\frac{n}{2}})_2$  und  
**18.3**  $a = (a_{\frac{n}{2}-1} \dots a_0)_2 = (d_{\frac{n}{2}-1} \dots d_0)_2$   
**18.4**  $T_4 = P, T_3 = 2^{\frac{n}{2}}P, T_0 = R, T_1 = -R, T_2 = T_1 + T_4,$   
**18.5**  $T_3 = T_1 + T_3,$  und  $T_4 = T_3 + T_4$   
**18.6** **for**  $\frac{n}{2} - 1$  **to**  $0$  **do**  
**18.7**      $T_0 = 2T_0$   
**18.8**     **if**  $(a_i, b_i) = (0, 0)$  **then**  $T_0 = T_0 + T_1$   
**18.9**     **else if**  $(a_i, b_i) = (1, 0)$  **then**  $T_0 = T_0 + T_2$   
**18.10**    **else if**  $(a_i, b_i) = (0, 1)$  **then**  $T_0 = T_0 + T_3$   
**18.11**    **else**  $T_0 = T_0 + T_4$   
**18.12** **end**  
**18.13** **return**  $T_0 + T_1$

---

**Fenster Methode** Alternativ kann man den EBRIP Algorithmus auf die Fenster-Methode (engl. window-Methode) adaptieren. Der modifizierte Algorithmus heißt dann WRIP-Algorithmus (window-BRIP). Die notwendigen Anpassungen werden im folgenden aufgelistet.

**Anpassungen von EBRIP zu WRIP:**

- a)  $R \leftarrow$  Zufallswertberechnung
- b) Festlegen der Fenstergröße  $w$
- c) erweitere  $n$  durch linksseitiges Auffüllen von Nullen, so dass  $d' = 0 \dots 0 d_{n-1} \dots d_0$  mit  $(n' < n + w)$  gilt
- d) Berechne  $R' = -(2^w - 1)R$
- e) Lege eine vorberechnete Tabelle fest mit  $T_w = \{R', P+R', 2P+R', 3P+R', \dots, (2^w - 2)P + R', (2^w - 1)P+R'\}$ , wobei  $|T_w| = 2^w$  gilt
- f) Berechne  $\underbrace{(0 \dots 0 d_{n-1} \dots d_0)}_{n'}_2 P + (1 \overline{11} \dots \overline{11})_{n'}_2 R$  mittels der Arithmetik der Fenster Methoden in  $T_w$

WRIP sichert analog zu EBRIP gegen DPA, RPA und ZPA Maßnahmen unter der Prämisse, dass  $R$  ein reiner Zufallswert ist. WRIP benötigt  $(n'+w)$  ECDBLs  $(\frac{n'}{w} + 2^w)$  ECADDs mit  $n' < n+w$ . Die Summe der benötigten Speichereinheiten beträgt hier  $2^w + 1$ . Beide Verfahren brauchen also eine gleich große Anzahl an Speichereinheiten, jedoch ist

EBRIP effizienter, da weniger ECDBLs benötigt werden.

### 5.3.4 Effizienz Betrachtung

Tabelle 5.1 faßt nochmals, die in diesem Kapitel vorgestellten BRIP und EBRIP Verfahren bzgl. ihres Speicheraufwands und ihrer Effizienz zusammen und vergleicht diese mit der Exponentensplitting Methode (in der Tabelle verkürzt ES), auf denen diese beiden Verfahren basieren. Die Tabelle zeigt den Speicher- und Rechenaufwand (Spalte 2 bzw. Spalte 3+4). Der Rechenaufwand wird in Spalte 3 bzgl. eines 160-Bit Felds angegeben und in Spalte 4 wird eine Umrechnung vorgenommen, die den Aufwand bzgl. der modularisierten Multiplikation (in der Tabelle kurz mit M angegeben) und Quadrierung (in der Tabelle kurz mit S angegeben) wiedergibt. Die Variable t gibt die Anzahl der Teilungen bei der EBRIP Methode wieder (vergl. Algorithmus 18 für  $t = 2$ ). In Spalte 4+5 werden die M und S Werte addiert, hierzu wird die Entsprechung  $S \approx 0.8 M$  festgelegt. Spalte 5 von Tabelle 5.1 gibt den Rechenaufwand in einem 160-Bit Feld und die jeweilige Umrechnung des Aufwands pro Bit wieder.

BRIP hat für die Berechnung von  $dP$  einen Rechenaufwand von 160 ECDBLs + 160 ECADDs und benötigt dafür 3 Speichereinheiten. Der Rechenaufwand pro Bit ist hierbei 24 M pro Bit, der dem Aufwand der Exponentensplitting Methode entspricht. EBRIP mit 2 Teilungen braucht bei einem Rechenaufwand von 160 ECDBLs + 84 ECADDs 5 Speichereinheiten, pro Bit fallen hier 16.0 M an. EBRIP mit 4 Teilungen benötigt bei einem Rechenaufwand von 160 ECDBLs + 56 ECADDs den grössten Speicheraufwand mit 17, ist jedoch am effektivsten mit 12.9 M pro Bit.

	Speicheraufwand	Rechenaufwand		Rechenaufwand pro bit
	#(Punkte,Skalare)	#(D + A)	#(M + S)	
ES	(4,2)	160D+160A	2856 M+1600 S(3840M)	16M+10S(24.0M)
BRIP	(3,0)	160D+160A	2856M+1600A(3840M)	16M+10S(24.0M)
EBRIP (t=2)	(3,0)	160D+84A	1648M+1140A(2560M)	10.3M+7.1S(16.0M)
EBRIP (t=3)	(9,0)	160D+62A	1392M+1008A(2198M)	8.7M+6.3S(13.7.0M)
EBRIP (t=4)	(17,0)	160D+56A	1312M+948A(2069M)	8.2M+5.9S(12.90M)

Tabelle 5.1: Vergleich der Gegenmaßnahmen

### 5.3.5 Abschluß Betrachtung

Der BRIP und der EBRIP Algorithmus sichern, sowohl gegen SPA als auch DPA Maßnahmen, wie auch gegen die darauf aufbauenden RPA und ZPA Maßnahmen. Das BRIP Verfahren im Vergleich zu EBRIP benötigt keine vorberechnete Tabelle, was bei Ressourcenknappheit, beispielsweise im Falle von Smart Cards, sehr wichtig sein kann. EBRIP benötigt zwar eine vorberechnete Tabelle, die auf der jeweiligen technischen Einheit abgelegt werden muss, kommt jedoch mit einer sehr geringen Anzahl an Operationen aus. Im Falle von  $t=4$  reichen 12.9 M pro Bit aus, was mit einer enormen Effizienzsteigerung verbunden ist.

# 6 Neue Angriffe

Die Entwicklung neuer Gegenmaßnahmen bewirkte stets die Entwicklung neuer Angriffstechniken. Dieses Kapitel beschreibt zwei Angriffstechniken, die von den zuvor genannten Gegenmaßnahmen nicht abgewehrt werden können.

## 6.1 Seitenkanäle während der Exponentenrekodierung [SS04]

Die meisten Seitenkanalangriffe und Gegenmaßnahmen in Public Key Kryptosystemen setzen bei RSA, während der modularen Exponentiation, bei elliptischen Kurven während der Punktmultiplikation ein. Das Rekodieren des Exponentes geschieht in der Regel vor diesem Schritt. Einige Rekodierungsalgorithmen haben verzweigte Befehle, die direkt vom Exponentenwert gesteuert werden. Diese Schwachstelle stellt einen Seitenkanal dar, über den ein Angreifer an geheime Informationen gelangen kann.

### 6.1.1 Grundlagen

Die Algorithmen 3, 4, 5 und 6 aus Abschnitt 2.2 stellen die verschiedenen NAF Varianten dar. Diese Algorithmen weisen verzweigte Befehle auf, die man mittels einfacher Beobachtung (vergl. SPA) nicht erkennen kann, jedoch mittels einer DPA Maßnahme. Um einen Angriff über diesen Seitenkanal durchführen zu können, muss ein Angreifer

- Zugang zur ausführenden Recheneinheit haben
- detaillierte Kenntnisse über die vorliegende Implementierung, inklusive des Algorithmus, der Parameter und des verwendeten Rekodierungs-Algorithmus haben
- die verzweigten Befehle unterscheiden können, mittels statistischer Analyse, während des Rekodierungs-Algorithmus

um letzten Endes die geheime Information erlangen zu können.

### 6.1.2 Seitenkanal

Der Width-w NAF Algorithmus von Seite 7 hat zwei verzweigte Befehle. Diese sind in Schritt 4.3 und 4.5 innerhalb der Schleife. In Schritt 4.3 wird abgefragt, ob ein Wert gerade ist, in Schritt 4.5 wird abgefragt, ob ein Wert unterhalb einer bestimmte Marke liegt. Dementsprechend wird im Code dann weitergegangen oder zum jeweiligen Label gesprungen. Für die weiteren Betrachtungen werden 3 verschiedene Sequenzen definiert.

- SSN steht für eine Sequenz, bei der zwei Subtraktionen stattfinden (in Schritt 4.6 und 4.8)
- SN steht für eine Sequenz, bei der eine Subtraktion stattfindet (in Schritt 4.6)
- N steht für eine Sequenz, bei der keine Subtraktion stattfindet

Ein Angreifer kann diese Sequenzen mittels einer DPA Maßnahme über den Stromverbrauch detektieren. Im Gegenteil zur Width-w NAF und den beiden Fractional Window Methoden, enthält die einfache NAF Realisierung keine verzweigte Befehle. Daher ist diese Realisierung aus reinen Sicherheitsaspekten vorzuziehen.

### 6.1.3 Angriffe

Für den Fall, dass die Realisierungen verzweigte Befehle enthalten, kann ein Angreifer mittels der ausgewerteten Sequenzfolgen, Rückschlüsse auf die geheimen Informationen gewinnen.

**Grundidee** Für einen fixen Exponenten ist die Sequenzfolge eindeutig festgelegt. Bei einem Angriff versucht man sich dieser Tatsache zu nutze zu machen. Jedoch muss hier die Umkehrung nicht eindeutig sein. Eine Sequenzfolge SN kann durch die Bitfolge  $(001)_2$  oder  $(011)_2$  entstanden sein, wie Tabelle 6.1 zeigt.

Daten im Fenster	entsprechende bits $d_i$	Sequenzen
0 = (000)	0	N
1 = (001)	1	SN
2 = (010)	0	N
3 = (011)	3	SN
4 = (100)	0	N
5 = (101)	-1	SSN
6 = (110)	0	N
7 = (111)	-1	SSN

Tabelle 6.1: Sequenzentsprechungen der Fensterwerte bzgl. Width-3 NAF

Ein Angreifer muss weiterhin beachten, dass bei einem Übertrag die Entsprechung, zwischen Bitfolgen und Sequenzen nicht immer stimmen muss. Für den Fall, dass Schritt 4.6 aus Algorithmus 4 ausgeführt wird, wäre der Wert für  $b_i$  negativ. Diese beiden Fälle müsste ein Angreifer einkalkulieren.

**Angriff auf die Width-w NAF Methode** Algorithmus 19 beschreibt einen Angriff auf den Width-w NAF Algorithmus, der auf folgende Beobachtungen basiert.

- Bei einer SSN Sequenz ist ein Übertrag in Schritt 19.8

## 6 Neue Angriffe

- Bei einer SSN Sequenz und danach folgenden N Sequenzen (SSN, N, N, ...,N, SSN) ist der Übertrag bei der folgenden SSN Sequenz

Ein Angreifer geht dann, wie folgt vor.

- Falls eine SSN Sequenz mit anschließendem Übertrag erscheint, sollte der Angreifer  $d_i = 0$  ‚erraten‘. Falls kein Übertrag folgt, sollte der Angreifer  $d_i = 1$  festlegen
- Nach einer SSN Sequenz sollte der Angreifer  $d_{i+w+1} = 1$  setzen
- Falls eine N Sequenz mit anschließendem Übertrag erscheint, sollte der Angreifer  $d_i = 1$  festlegen. Bei keinem anschließendem Übertrag  $d_i = 0$ .
- Für SN Sequenzen gilt die gleiche Verfahrensweise, wie für SSN Sequenzen bzgl. des ‚Erratens‘ von  $d_i$
- Falls eine SN Sequenz erscheint wird ein Übertrag nicht berücksichtigt

Das Symbol ‚state‘ in Algorithmus 19 soll ein Hinweis auf einen möglichen Übertrag sein, bei gesetztem Bit. An den Stellen, an denen unklar ist, ob  $d_i$  gesetzt ist oder nicht, erfolgt der Eintrag ‚nicht eindeutig‘ markiert (in den Algorithmen kurz mit ‚n.e.‘ abgekürzt).

---

**Algorithmus 19** : Angriff auf die Exponentenrekodierung mit Width-w NAF

---

**Input** : eine SN Sequenz  $(v_0, v_1, \dots)$  mit  $v_i \in \{SSN, SN, N\}$ , eine Zahl  $w \geq 2$ , eine t-bit lange Zahl d

**Output** : Exponent  $d = (d_{t-1} \dots d_0)_2$

```

19.1 i ← 0
19.2 state ← 0
19.3 while i < t do
19.4   if  $v_i = SSN$  then
19.5     if state = 1 then  $d_i \leftarrow 0$ 
19.6     else if state = 0 then  $d_i \leftarrow 1$ 
19.7     for j = i + 1 to i + w - 2 do  $d_j \leftarrow$  "unknown"
19.8      $d_{i+w-1} \leftarrow 1$ 
19.9     i ← i + w
19.10    state ← 1
19.11  else if  $v_i = SN$  then
19.12    if i + w - 1 < t then
19.13      if state = 1 then  $d_i \leftarrow 0$ 
19.14      else if state = 0 then  $d_i \leftarrow 1$ 
19.15      for j = i + 1 to t - 2 do  $d_j \leftarrow$  "unknown"
19.16       $d_{t-1} \leftarrow 1$ 
19.17      i ← t
19.18    else
19.19      if state = 1 then  $d_i \leftarrow 0$ 
19.20      else if state = 0 then  $d_i \leftarrow 1$ 
19.21      for j = i + 1 to i + w - 2 do  $d_j \leftarrow$  "unknown"
19.22       $d_{i+w-1} \leftarrow 1$ 
19.23      i ← i + w
19.24    end
19.25    state ← 0
19.26  else if  $v_i = N$  then
19.27    if state = 1 then  $d_i \leftarrow 0$ 
19.28    else if state = 0 then  $d_i \leftarrow 0$ 
19.29    i = i + 1
19.30  end
19.31 end
19.32 return  $d = (d_{t-1} \dots d_0)_2$ 

```

---

**Angriff auf die vorzeichenlose Fractional Window Methode** Hier erfolgt der Angriff analog zu einem Angriff auf die Width-w NAF Methode. Die datenabhängigen Instruktionen sind in Schritt 5.6 und 5.8 aus Algorithmus 5. Der Unterschied zur w-NAF Methode besteht darin, dass selbst wenn die entsprechende Subtraktion in Schritt 19.8 ausgeführt

wird, kein Übertrag auftritt.

---

**Algorithmus 20** : Angriff auf die vorzeichenlose Fractional Window Methode

---

**Input** : eine SN Sequenz  $(v_0, v_1, \dots)$  mit  $v_i \in \{SSN, SN, N\}$ , eine Zahl  $w \geq 2$ , eine ungerade Zahl  $m$ , mit  $1 \leq m \leq 2^w - 3$ , eine t-bit lange Zahl  $d$

**Output** : Exponent  $d = (d_{t-1} \dots d_0)_2$

```

20.1  $i \leftarrow 0$ 
20.2 while  $i < t$  do
20.3   if  $v_i = SSN$  then
20.4      $d_i \leftarrow 1$ 
20.5     for  $j = i + 1$  to  $i + w - 1$  do  $d_j \leftarrow$  "unknown"
20.6      $i \leftarrow i + w$ 
20.7   else if  $v_i = SN$  then
20.8      $d_i \leftarrow 1$ 
20.9     if  $i + w - 1 > t$  then
20.10      for  $j = i + 1$  to  $t - 2$  do  $d_j \leftarrow$  "unknown"
20.11       $d_{t-1} \leftarrow 1$ 
20.12       $i \leftarrow t$ 
20.13   else
20.14     for  $j = i + 1$  to  $i + w$  do  $d_j \leftarrow$  "unknown"
20.15      $i \leftarrow i + w + 1$ 
20.16   end
20.17   else if  $v_i = N$  then
20.18      $d_i \leftarrow 0$ 
20.19      $i = i + 1$ 
20.20   end
20.21 end
20.22 return  $d = (d_{t-1} \dots d_0)_2$ 

```

---

**Angriff auf die vorzeichenbehaftete Fractional Window Methode** Auch hier verläuft der Angriff analog dem Angriff auf width- $w$  NAF. Ein Überlauf entsteht, falls nach einer SSN Sequenz  $(w+1)$  N Sequenzen aufeinander folgen. Der Übertrag wirkt sich dann auf die folgende SN oder SSN Sequenz aus. In Algorithmus 21 zählt die Variable  $c$  die Anzahl der aufeinanderfolgenden N Sequenzen.

---

**Algorithmus 21** : Angriff auf die vorzeichenbehaftete Fractional Window Methode

---

**Input** : eine SN Sequenz  $(v_0, v_1, \dots)$  mit  $v_i \in \{\text{SSN}, \text{SN}, \text{N}\}$ , eine Zahl  $w \geq 2$ , eine ungerade Zahl  $m$ , mit  $1 \leq m \leq 2^w - 3$ , eine t-bit lange Zahl  $d$

**Output** : Exponent  $d = (d_{t-1} \dots d_0)_2$

```

21.1  $i \leftarrow 0$ 
21.2  $\text{state} \leftarrow 0$ 
21.3  $c \leftarrow 0$ 
21.4 while  $i < t$  do
21.5   if  $v_i = \text{SSN}$  then
21.6     if  $c \geq w + 1$  und  $\text{state} = 1$  then  $d_i \leftarrow 0$ 
21.7     else if  $c = w$  und  $\text{state} = 1$  then  $d_i \leftarrow \text{"unknown"}$ 
21.8     else  $d_i \leftarrow 1$ 
21.9     for  $j = i + 1$  to  $i + w - 1$  do  $d_j \leftarrow \text{"unknown"}$ 
21.10     $i \leftarrow i + w + 1$ 
21.11     $c \leftarrow w$ 
21.12  else if  $v_i = \text{SN}$  then
21.13    if  $c \geq w + 1$  und  $\text{state} = 1$  then  $d_i \leftarrow 0$ 
21.14    else if  $c = w$  und  $\text{state} = 1$  then  $d_i \leftarrow \text{"unknown"}$ 
21.15    else  $d_i \leftarrow 1$ 
21.16    if  $i + w + 1 > t$  then
21.17      for  $j = i + 1$  to  $t - 2$  do  $d_j \leftarrow \text{"unknown"}$ 
21.18       $d_{t-1} \leftarrow 1$ 
21.19       $i \leftarrow t$ 
21.20    else
21.21      for  $j = i + 1$  to  $i + w$  do  $d_j \leftarrow \text{"unknown"}$ 
21.22       $d_{i+w+1} \leftarrow 0$ 
21.23       $i \leftarrow i + w + 2$ 
21.24    end
21.25     $c \leftarrow w$ 
21.26     $\text{state} \leftarrow 0$ 
21.27  else if  $v_i = \text{N}$  then
21.28    if  $\text{state} = 1$  then  $d_i \leftarrow 1$ 
21.29    else if  $\text{state} = 0$  then  $d_i \leftarrow 0$ 
21.30     $c \leftarrow c + 1$ 
21.31     $i \leftarrow i + 1$ 
21.32  end
21.33 end
21.34 return  $d = (d_{t-1} \dots d_0)_2$ 

```

---

### 6.1.4 Experimentelle Ergebnisse

Um die vorgestellten Angriffe vergleichen zu können, wurde getestet, wieviel Prozent der Bits, der geheimen Informationen detektiert werden können. Hierzu wurden 160-Bit, 512-Bit und 1024-Bit Exponenten verwendet. Bei der Width- $w$  NAF Methode wurden die Fenstergrößen  $w = 2, 3, 4, 5$  getestet. Bei der Fractional Window Methode wurde mit den Werten für  $m = 1, 5, 13, 29, \dots, 2^w - 3$  getestet. Tabelle 6.2 zeigt, dass für große Fenstergrößen die Trefferwahrscheinlichkeit rapide sinkt. Dies hängt damit zusammen, dass bei größerer Fenstergröße die mittleren Bits öfters nicht eindeutig angegeben werden können. Dies spiegelt sich in den Algorithmen durch den Eintrag ‚n.e.‘ (nicht eindeutig) wieder.

w	m	width-w NAF	vorzeichenlose F.W.	vorzeichenbehaftete F.W.
2	-	100	-	-
	1	-	50.5	50.3
3	-	75.1	-	-
	1	-	38.9	36.3
	5	-	40.0	46.0
4	-	60.2	-	-
	1	-	31.3	28.3
	13	-	33.7	41.4
5	-	50.3	-	-
	1	-	26.2	23.5
	1	-	29.1	37.1

Tabelle 6.2: Messergebnisse: Anzahl der reproduzierten Bits in % (= Anzahl der reproduzierten Bits/ Bitlänge des Exponentens  $d$ )

### 6.1.5 Abschlußbetrachtung

Um die Berechnung des Produkts  $dP$  zu beschleunigen, werden Rekodierungsverfahren, wie die  $w$ -NAF und die Fractional Window Methoden benutzt. Diese öffnen jedoch einen Seitenkanal. Diesen kann man verhindern indem auf die einfache NAF Rekodierungsmethode zurückgegriffen wird. Dieser Sachverhalt impliziert, dass solange die Exponentenrekodierung bis zu einem gewissen Grad nicht sicherheitsbewußt implementiert ist, bleiben elliptische Kurven-Kryptosysteme (und auch RSA-Kryptosysteme) stets anfällig gegenüber DPA Maßnahmen und ihren sich stets verbesserten Modifikationen. Dies würde dann auch die in Abschnitt 5.3 vorgestellten BRIP und WRIP-Methoden betreffen, die vor der Hauptberechnung einen Rekodierungsvorgang durchlaufen.

## 6.2 Simple Branch Prediction Analysis

Neuere Forschungsergebnisse zeigen, dass der interne Aufbau von Prozessoren und von Cachearchitekturen Angriffsflächen für Seitenkanalangriffe bieten können [AKKS07, JP]. Einer dieser Angriffe setzt an einer bestimmten Stelle, während der Befehlsabarbeitung (engl. Instruction Execution, IE) [Hof93] und zwar an der Sprungvorhersage an (engl. Simple Branch Prediction Analysis, SBPA), die in den jeweiligen Zyklen der Befehlsabarbeitung entscheidet zu welcher Sprungmarke (engl. label) im Folgetakt gesprungen wird.

### 6.2.1 Grundlagen des Angriffes

Aktuelle CPU Einheiten verwenden zur parallelen Abarbeitung mehrerer Befehle, eine Pipeline-Architektur (engl. Pipelining) [Hof93]. Eine Pipeline-Architektur beschreibt die Fähigkeit von CPU Einheiten, die Abarbeitung von Maschinenbefehlen so zu parallelisieren, dass alle Funktionseinheiten der CPU voll ausgelastet werden und dadurch eine größere Effizienz erreicht wird. Hierbei werden komplexe Befehle in einfachere Befehle zerlegt, die dann parallel in ihren jeweiligen Pipelineinstufen takt synchron abgearbeitet werden können.

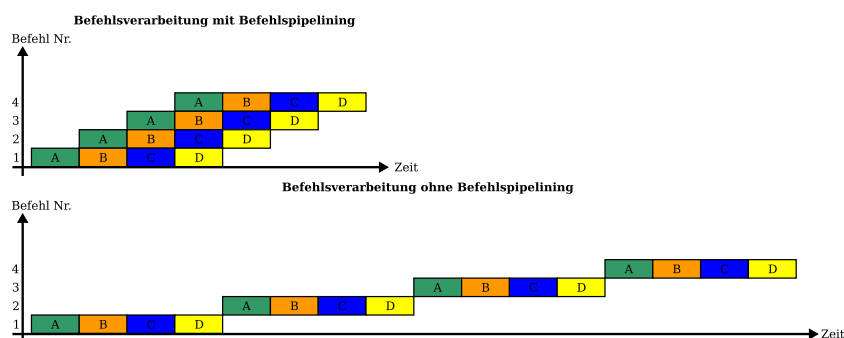


Abbildung 6.1: Pipeline-Architektur

Ist es für die Bearbeitung eines Befehls in einer Stufe der Pipeline notwendig, dass ein Befehl, der sich weiter vorne in der Pipeline befindet, zuerst abgearbeitet wird, so spricht man von Abhängigkeiten. Eine bestimmte Abhängigkeit stellt der Kontrollflußkonflikt dar [Hof93]. Hier muß die Pipeline abwarten, ob ein bedingter Sprung ausgeführt wird oder nicht. Diese bedingten Sprünge sind hauptverantwortlich für große Latenzzeiten und dadurch bedingte Effizienzeinbußen, da bei einem Sprung die eingelesenen Befehle, ohne dass sie ausgeführt wurden, aus der Pipeline „ausgespült“ (engl. Flush) [Hof93] werden und neue Befehle in die Pipeline geladen werden müssen. Um die Anzahl dieser sogenannten Spülvorgänge, die hauptverantwortlich für Effizienzeinbußen sind, zu verringern wurde das Konzept der „Sprungvorhersage“ (engl. Branch Prediction) [AKKS07] entwickelt. Je besser diese Sprungvorhersage arbeitet, desto effizienter arbeitet der Prozessor.

Die Sprungvorhersage versucht aufgrund statistischer Abschätzungen den Folgebefehl zu erraten. Dadurch soll verhindert werden, dass Befehle in die Pipeline geladen werden, und bei einem Sprung aus der Pipeline gespült werden, da dies eine enorme Performanzverschlechterung bewirkt.

### 6.2.2 Aufbau der Sprungvorhersage Einheit

Abbildung 6.2 zeigt den Aufbau einer Sprungvorhersage Einheit (engl. Branch Prediction Unit BPU). Die BPU besteht hauptsächlich aus zwei logischen Einheiten, dem Zieladressenpuffer (engl. Branch Target Buffer BTB) und der Vorhersageeinheit (engl. predictor), die wiederum in kleinere Einheiten aufgesplittet ist. Der BTB fungiert ähnlich einem Cachespeicher und speichert potenzielle Sprungadressen, die bei vorherigen Sprungbefehlen anvisiert wurden. Auch dies ist eine Maßnahme zu Verbesserung der Eeffizienz, da dadurch die Adressen nicht erst aus dem Speicher geholt werden müssen und einige Takte eingespart werden. Jedoch kann der BTB nicht beliebig viele Adressen speichern, so dass bei Einlagerung neuer Adressen, ältere Adressen aus dem BTB entfernt werden. Und dieser Sachverhalt ist zentral für den hier beschriebenen Angriff. Die Vorhersageeinheit gibt die potenziellen Sprungadressen an, die als mögliche Folgebefehle in der BTB eingespeichert werden.

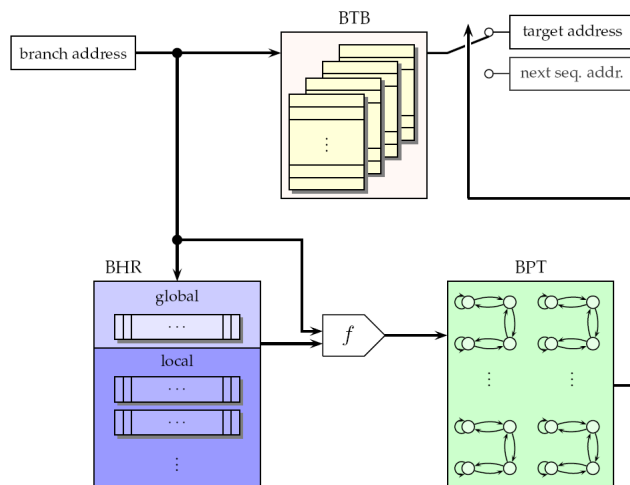


Abbildung 6.2: BPU Architektur

### 6.2.3 Angriff

Der Angriff sieht so aus, dass während einer Berechnung mit einem geheimen Schlüssel, ein Lauschprozeß (engl. spy process) parallel mitläuft. Für den folgenden Angriff wurde eine RSA Entschlüsselung bzw. Signierung zugrunde gelegt. Jedoch ist dieser Angriff auch auf andere Angriffe, somit auch auf elliptische Kurven übertragbar.

Dieser Lauschprozeß sollte

- alle Verzweigungen (engl. branches) der zugrundeliegenden Berechnung sequentiell durchlaufen und
- misst die Ausführungszeit an allen Verzweigungen

Hierbei müssen beide Prozesse auf den gleichen Zieladressenpuffer (BTB) zeigen, der auch die bedingten Sprünge speichert, die in Abhängigkeit des geheimen Schlüssels des Kryptoprozesses, erfolgen. Der Angreifer startet den Lauschprozeß vor dem jeweiligen Kryptoprozeß. Wenn der Kryptoprozeß daraufhin ebenfalls startet, hat dies zur Folge, dass die CPU unter den anvisierten Sprüngen nicht die „spekulierte“ Folgesprungadresse im Zielspeicherpuffer (BTB) findet und das Ergebnis der Vorhersageeinheit für den Kryptoprozeß „nicht springen“ heißt. Man kann hier zwei Fälle unterscheiden:

- Falls der Sprung getätigt wird, dann wird eine Fehlvorhersage (engl. misprediction) eintreten und eine **falsche** Sprungadresse C (bzw. Zieladresse) wird im BTB gespeichert. Dadurch wird eine Sprungadresse L des Lauschangriffes (die zuvor im BTB eingespeichert wurde) aus dem BTB entfernt um diese neue Sprungadresse C darin einzuspeichern. Sobald die Sprünge des Lauschangriffes nachvollzogen sind, stellt man eine Fehlvorhersage an der jeweiligen Sprungstelle fest, deren Zieladresse aus dem BTB entfernt worden ist. Da weiterhin die Laufzeit aller Sprünge vom Lauschprozeß gemessen werden, kann dadurch festgestellt werden, bei welchen Sprüngen der BTB verändert wurde.
- Falls der Sprung nicht getätigt wurde, dann trat keine Fehlvorhersage ein und der BTB wurde nicht modifiziert. Durch das Nachvollziehen der Sprünge in Kombination mit der gemessenen Zeit, kann man schlußfolgern, dass der Kryptoprozeß keine Sprünge genommen hat.

Mit dieser prinzipiell einfachen Vorgehensweise kann ein Angreifer die einzelnen Schritte der Berechnung nachvollziehen, indem er die Entscheidungen bei den anvisierten Sprüngen beobachtet und die Ausführungszeit dabei misst. Auf diese Weise ermöglicht der Lauschprozeß die komplette Folge von richtigen und falschen Vorhersagen der anvisierten Sprünge zu rekonstruieren und ermittelt dadurch den geheimen Schlüssel.

### 6.2.4 Messung

Um die Umsetzbarkeit des vorgestellten Angriffs zu illustrieren, ließ man eine RSA Entschlüsselung parallel zu einem Lauschprozeß laufen. Das Ergebnis zeigt Abbildung 6.3. Auf der Abzisse ist der Wert für die Zahl der Messungen aufgetragen, auf der Ordinate die Anzahl der Quadrierungen (blaue Vierecke) und Multiplikationen (rote Dreiecke). Je

höher der Wert der Messungen ist, desto vernachlässigbarer werden die Störgeräusche und desto klarer kann man die jeweilige Operation an der gemessenen Stelle detektieren.

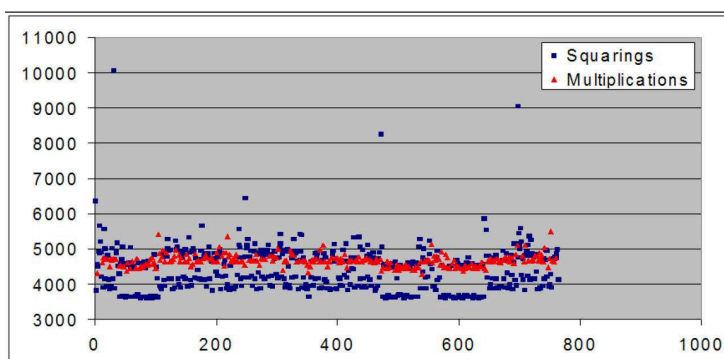


Abbildung 6.3: Messergebnisse bei SPBA

### 6.2.5 Heuristik

Jedoch reichen die bisher gewonnenen Informationen noch nicht aus um einen geheimen Schlüssel aus einem einzigen Lauschangriff zu restrukturieren, da immer noch zu viele Störgeräusche in den Messwerten vorhanden sind. Diese Störgeräusche werden durch die vielen verschiedenen Prozesse, die auf einer PC-Plattform parallel laufen bedingt. Je nachdem welche und wie viele dieser Störprozesse laufen, wird die Messung dementsprechend verfälscht. Um jedoch aus den Messergebnissen eine aussagekräftige Aussage ziehen zu können, behilft man sich einem einfachen statistischem Prinzip, der Heuristik [AKKS07, JP]. Diese besagt, dass es unter einer bestimmten Anzahl von Messungen eine Messung geben muß, die minimal an Störprozessen ist. Diese eine Messung ist mit sehr großer Wahrscheinlichkeit die gesuchte Messung, die eine SBPA Maßnahme ermöglicht.

### 6.2.6 Fazit

Ein SPBA Maßnahme machte alle bisher vorgestellten Gegenmaßnahmen unbrauchbar, da diese Angriffstechnik einem gänzlich anderen Ansatz als die bisherigen Angriffstechniken folgt. Waren in der Vergangenheit Blinding- oder diverse Randomisierungsverfahren und deren Kombinationen die stärksten Verfahren um beispielsweise eine RSA Berechnung gegen Seitenkanalangriffe zu schützen, zeigt der erfolgreiche Angriff auf die gängigste RSA Implementierung, OpenSSL [JP], dass dies in Zukunft überdacht werden muss. Weiterhin ist die bisher weitverbreitete Annahme, dass das Ausbalancieren von Operationen nach bedingten Sprüngen, eine höhere Sicherheitsstufe darstellt, durch diesen Angriff ebenfalls dahin. Die Tatsache, dass für diesen Angriff eine einzelne Berechnung ausreicht, impliziert den Bedarf an verbesserten Verschlüsselungstechniken, da sie die Sicherheit von Verfahren, die auf einem geheimen Schlüssel, d.h. alle asymmetrischen Verfahren, beruhen, akut gefährden.

# 7 Vergleich aller Gegenmaßnahmen

Das Ziel aller Gegenmaßnahmen ist es, den Aufwand für einen potenziellen Angreifer so dermaßen zu erschweren, dass dieser an die Grenzen seiner finanziellen und technischen Möglichkeiten gelangt. Um dies genauer spezifizieren zu können, wäre es von großem Vorteil, die jeweils am besten geeignete Gegenmaßnahme für jeden Fall genau angeben zu können. Dies erweist sich als schwierig, da bisher keine einheitlichen Klassifizierungen von Gegenmaßnahmen bzgl. ihrer Sicherheitsgüte, ihrer Komplexität und der benötigten technischen Voraussetzungen existieren. In diesem Kapitel greifen wir das von Itoh, Izu und Takenaka entwickelte Evaluationsschema [IIT03] auf, das es ermöglicht Gegenmaßnahmen in Hinblick auf ihr Sicherheitslevel, ihre Rechengeschwindigkeit und ihren Speicherbedarf zu vergleichen und vervollständigen die Bewertungstabelle mit den in dieser Arbeit beschriebenen Gegenmaßnahmen.

## 7.1 Bewertungsschema

Abschnitt 7.2 listet alle gängigen Standardalgorithmen und Gegenmaßnahmen auf, die in das folgende Bewertungsschema eingeflossen sind. Die Methoden 1-4 stellen die Standardalgorithmen für die skalare Multiplikation  $dP$  dar. Die Methoden 5-11 stellen Gegenmaßnahmen dar, die durch Modifikationen, in der Regel unterschiedliche Randomisierungen, aus den Standardalgorithmen entwickelt wurden. Diese Methoden wurden in Kapitel 3 allgemein und in den Kapiteln 4, 5 und 6 mit konkreten Implementierungen detaillierter beschrieben. Um für alle Gegenmaßnahmen vergleichbare Werte zu erhalten, wurden die Messungen in einem 160-Bit ECC Feld vorgenommen. Die folgenden 3 Messgrößen sind, wie folgt, definiert:

**Sicherheitslevel** Die Sicherheit einer Gegenmaßnahmen gegen SPA, DDPA<sup>1</sup> und ADPA wird mit dem Dezimalwert  $AR^2$ , der zwischen 0 und 1 liegt, angegeben. Der  $AR$ -Wert stellt das arithmetische Mittel der auffälligen, extremalen Werte der dazugehörigen Spannungsverlaufkurve [IIT03] dar.  $AR = 0$  impliziert, dass aus der Spannungsverlaufkurve keine Seitenkanalinformationen detektiert werden können.  $AR = 1$  impliziert, dass stets Seitenkanäle aus der Spannungsverlaufkurve entnommen werden können und diese Gegenmaßnahme absolut unsicher ist. Ein kleiner  $AR$ -Wert ist dementsprechend wünschenswert für ein hohes Sicherheitslevel. Der  $AR$ -Wert für eine SPA Maßnahme wird mit  $AR_S$  bezeichnet, dementsprechend für DDPA mit  $AR_D$  und für ADPA mit  $AR_A$ . Während die Werte für  $AR_S$ , entweder genau 0 oder 1 sein können, sind die Werte für

---

<sup>1</sup>DDPA (Kurzform für Daten DPA) wird in der Literatur oft nur mit DPA angegeben, vergl. Kap3

<sup>2</sup>Kurzform für Attenuation Ratio [IIT03]

## 7 Vergleich aller Gegenmaßnahmen

$AR_D$  und  $AR_A$  aus dem Intervall  $[0;1]$ . Dies erklärt sich wie folgt: Die Sicherheit der Randomisierungsalgorithmen 5-11 hängt von der Sicherheit des Zufallswerts ab. Falls ein Angreifer den Zufallswert leicht erraten kann, ist der Algorithmus unsicher. Da wir von einem 160-Bit ECC Feld ausgehen, ist die Wahrscheinlichkeit dafür  $1/2^{160}$ .

**Rechengeschwindigkeit** Die Rechengeschwindigkeit einer skalaren Exponentiation  $dP$  wird durch die Anzahl der ausgeführten ECADD (kurz  $N_A$ ) und ECDBL Operationen (kurz  $N_D$ ) angegeben. Für die Gegenmaßnahmen 1-4 werden konkrete Zahlenwerte angegeben. Für die weiteren Gegenmaßnahmen wird ein Faktor angegeben. Z.B. bedeutet "×a", dass diese Gegenmaßnahmen a-mal so viele Operationen benötigt, im Vergleich zu Methode 1, MSB. Die Gegenmaßnahmen 5-11 sind durch Modifikationen aus den Gegenmaßnahmen 1-4 entstanden. Die Laufzeiten für Randomisierungen und Transformationen, werden hier vernachlässigt, da sie im Vergleich zu den  $N_A$  und  $N_D$  Werten relativ gering sind.

**Speicherbedarf**  $R_P$  gibt die Anzahl der benötigten Register für Punkte auf der jeweiligen elliptischen Kurve wieder,  $R_S$  die Anzahl der Register für Skalare. Analog zur Rechengeschwindigkeit werden diese beiden Werte bei den Gegenmaßnahmen 1-4 mit konkreten Zahlenwerten angegeben, bei den Gegenmaßnahmen 5-11 die zusätzlich benötigten Register. Dementsprechend bedeutet "+a", dass diese Gegenmaßnahme zusätzlich a Register benötigt.

## 7.2 Standardalgorithmen und Gegenmaßnahmen

- (1) **Binäre Methode MSB**, Standardalgorithmus zur Berechnung des skalaren Produkts  $nP$  (Algorithmus 1)
- (2) **Binäre Methode LSB**, Standardalgorithmus zur Berechnung des skalaren Produkts  $nP$  (Algorithmus 2)
- (3) **Add-and-double-always Methode**, Gegenmaßnahme, die gegen SPA absichert (Algorithmus 12)
- (4) **Montgomery Ladder**, Gegenmaßnahme, die gegen SPA absichert [OKS00] (Algorithmus 13)
- (5) **Randomisierte Projektive Koordinate (RPC)**, der Punkt  $(x, y, z)$  wird mittels eines Zufallswert  $r$  in den Punkt  $(rx, ry, rz)$  transformiert [Cor99] (vergl. Abschnitt 2.1.3 und 3.1.2.2)
- (6) **Randomisierte Kurve (RC)**, der Punkt  $(x, y, z)$  wird in den Punkt  $(r^2x, r^3y, rz)$  transformiert, der auf einer isomorphen Kurve liegt [JT01] (vergl. Abschnitt 2.1.3 und 3.1.2.2)
- (7) **Randomisierung des Skalars**, die Berechnung  $dP$  wird durch den Term  $d(P + R) - dR$  substituiert [Cor99] (vergl. Abschnitt 5.3.3)
- (8) **Randomisierter Exponent**, die Berechnung  $dP$  wird durch  $(d + rn)P$  substituiert, wobei  $n = \text{ord}(P)^h$  gilt [Cor99]
- (9) **Randomisierter Startpunkt**, die Berechnung  $dP$ , beginnt ab einem gewähltem Punkt  $d - i$ , für die höherwertigen Bits erfolgt die Berechnung mittels MSB, für die niederwertigen Bits erfolgt die Berechnung mittels LSB [MDS99]
- (10) **Exponentensplitting**, die Berechnung  $dP$  wird durch  $(d - r)P + rP$  substituiert, wobei  $r$  einen Zufallswert darstellt [CJ01]
- (11) **Randomisierte Adressmethode (RA)**, wird in Abschnitt 4.3 beschrieben

---

<sup>h</sup> „ $\text{ord}(P)^h$ “ meint die Ordnung von  $P$

### 7.3 Auswertung

Die zuvor aufgelisteten Gegenmaßnahmen wurden anhand des Bewertungsschemas analysiert. Die Auswertungen zeigt Tabelle 7.1.

Bei der Betrachtung der Tabelle 7.1 fällt auf, dass keine der Methoden 3-11 eine Sicherheit gegen die drei gängigen Angriffstechniken SPA, DDPA und ADPA darstellen kann. Jede Methode schneidet zumindest gegen eine Angriffstechnik gut ab, gegen die anderen beiden wiederum sehr schlecht ab, was man an den häufigen  $AR = 1$  Einträgen erkennen kann.

**SPA-sicher** Methode 1 und 2 kann man bei dieser Betrachtung vernachlässigen, da sie lediglich die einfachen Standardalgorithmen für die skalare Multiplikation darstellen und keine Sicherungsmechanismen integriert haben. Methode 3 (Add-and-always-Methode) und 4 (Montgomery Ladder) sichern gegen SPA ab, da sie in jedem Schleifendurchlauf für jedes Bit eine ECDBL und eine ECADD Operation ausführen. Jedoch ist bei beiden, sowohl die Daten- als auch die Adresszuweisung an das jeweilige Bit gekoppelt (bei Methode 3 in Schritt 12.5, bei Methode 4 in den Schritten 13.3, 13.5 und 13.6, der Algorithmen auf Seite 29). Daher sind beide Verfahren unsicher gegen DDPA und ADPA.

**DDPA-sicher** Methode 5 (RPC), 6 (RC) und 7 (Randomisierung des Skalars) haben ein relativ hohes Sicherheitslevel (sehr kleiner AR-Wert) bzgl. DDPA, da sie für jede Berechnung einen neuen Zufallswert  $r$  generieren, der in die Berechnung einfließt. Dadurch wird eine statistische Analyse unwirksam. Gegen SPA und ADPA sichern diese Maßnahmen nicht ab, da sie, wie Methode 1, auf der diese Methoden basieren, keine Sicherungsmechanismen bzgl. dieser Angriffstechniken integriert haben. Eine Kombination mit Methode 3 oder 4 würde zusätzlich gegen SPA absichern.

**ADPA-sicher** Methode 8, 9 und 10 sichern ebenfalls gegen DDPA ab, da für jede Berechnung eine neue Zufallszahl  $r$  generiert wird. Dadurch, dass diese Methoden den Exponenten  $d$  randomisieren, sichern sie zusätzlich gegen ADPA ab. Denn die Adresszuweisung erfolgt über ein einzelnes Bit aus der Bitfolge  $d_{n-1}d_{n-2}...d_0$  und da genau diese Bitfolge  $d$  für jede Berechnung randomisiert wird, ist dieser Seitenkanal geschlossen. Dass Methode 11 gegen ADPA absichert, kann man Abschnitt 4.3 entnehmen.

## 7 Vergleich aller Gegenmaßnahmen

Nr.	Methoden	$AR_S$	$AR_D$	$AR_A$	$N_D$	$N_A$	$R_P$	$R_S$
1	Binäre Methode (MSB)	1	1	1	160	80	1	0
2	Binäre Methode (LSB)	1	1	1	160	80	2	0
3	Add-and-double-always Methode	0	1	1	$\times 1$	$\times 2$	+1	+0
4	Montgomery Ladder	0	1	1	160	160	3	0
5	Randomisierte Proj. Koordinate (RPC)	1	$2^{-160}$	1	$\times 1$	$\times 1$	+1	+1
6	Randomisierte Kurve (RC)	1	$2^{-160}$	1	$\times 1$	$\times 1$	+0	+3
7	Randomisierung des Skalars	1	$2^{-160}$	1	$\times 2$	$\times 2$	+2	+0
8	Randomisierter Exponent ( $ r =20$ ) <sup>3</sup>	1	$2^{-20}$	0	$\times 1.13$	$\times 1.13$	+0	+0
9	Randomisierter Startpunkt	1	$\frac{1}{160}$	0	$\times 1$	$\times 1$	+0	+0
10	Exponentensplitting	1	$2^{-160}$	0	$\times 2$	$\times 2+1$	+1	+2
11	Randomisierte Adressmethode (RA)	1	1	0	$\times 1$	$\times 1$	+0	+2
I	Beste Kombination 1+3+5+11	0	$2^{-160}$	0	160	160	3	3
II	Beste Kombination 1+3+6+11	0	$2^{-160}$	0	160	160	2	5
III	weitere Kombination 4+5+11	0	$2^{-160}$	0	160	160	4	3
IV	weitere Kombination 4+6+11	0	$2^{-160}$	0	160	160	3	5

Tabelle 7.1: Vergleich der Gegenmaßnahmen

### Kombinierte Gegenmaßnahmen

Da keine der Methoden 1-11 eine ausreichende Sicherheit gegen alle, der drei gängigen Angriffstechniken SPA, DDPA und ADPA gewährleisten können, werden diese Gegenmaßnahmen kombiniert. Auch hier stellt sich die Frage, welche Kombinationen von Gegenmaßnahmen, für den jeweiligen Fall, am geeignetesten sind. Auch hier gibt das Modell von Itoh, Izu und Takenaka eine Hilfestellung. Die AR-Werte der Kombinationen ergeben sich aus dem Produkt der AR-Werten der kombinierten Methoden. Analog verhält es sich für die Rechengeschwindigkeit. Der Speicherbedarf ergibt sich aus der Summe der Register der kombinierten Methoden. Z.B. stellt Methode I eine Kombination aus den Methoden 1, 3 (zur Absicherung gegen SPA), 5 (zur Absicherung gegen DDPA) und 11 (zur Absicherung gegen ADPA) dar. Die resultierenden Werte für I bestimmen sich wie folgt:

$$AR_S = 1 \times 0 \times 1 \times 1 = 0,$$

$$AR_D = 1 \times 2^{-160} \times 1 \times 1 = 2^{-160},$$

$$AR_A = 1 \times 1 \times 1 \times 0 = 0,$$

$$N_D = 160 \times 1 \times 1 \times 1 = 160,$$

$$N_A = 80 \times 2 \times 1 \times 1 = 160,$$

$$R_P = 1 + 1 + 1 + 0 = 3,$$

$$R_S = 0 + 0 + 1 + 2 = 3,$$

Die Methoden I, II, III und IV aus Abbildung 7.1 stellen die besten Kombinationen aus den Methoden 1-11 dar. Diese kombinierten Gegenmaßnahmen sichern, sowohl gegen SPA, DDPA als auch ADPA ab. Methode I und II stechen zusätzlich heraus, da sie eine

geringe Anzahl an Speicherplätzen benötigen. Dieser Aspekt kann sehr wesentlich, bei begrenzten Speicherressourcen, sein.

Durch dieses vorgestellte Modell ist es möglich die unterschiedlichen Gegenmaßnahmen direkt miteinander zu vergleichen hinsichtlich ihres Sicherheitslevel, ihrer Rechengeschwindigkeit und ihres Speicherbedarfs. Durch die Möglichkeit die Messwerte kombinatorisch verknüpfen zu können, ist es nun möglich eine Sicherheitsaussage über die kombinierten Gegenmaßnahmen zu treffen.

## 7.4 Erweiterung der Tabelle um die neuen Angriffstechniken

Um Tabelle 7.1 zu vervollständigen, fügen wir zu den Angriffstechniken die Höherstufigen DPA Maßnahmen RPA (in der Tabelle mit  $AR_R$  angegeben) und ZPA ( $AR_Z$ ), samt ihrer Gegenmaßnahme BRIP (vergl. 5.3) hinzu. Weiterhin beschreiben wir, welchen Effekt die neue Angriffstechnik SBPA (vergl. Abschnitt 6.2) auf die in dieser Arbeit beschriebenen Sicherungstechniken gegen Seitenkanalangriffe hat.

Der Zufallswert beim Randomisiertem Exponenten (Methode 8) wird auf 160-Bit erweitert um eine vergleichbare Darstellung zu erhalten und mittlerweile die Verknüpfung von ungleich großen Schlüsseln als sehr unsicher gilt [IIT03, OS00].

**RPA und ZPA** stellen modifizierte DPA Maßnahmen dar und werden dementsprechend auch als Sekundärstufige bzw. Mehrstufige DPA bezeichnet. Beide Angriffstechniken zielen auf spezielle Fixpunkte ab, die auch nach einer Randomisierung ihre Fixeigenschaft nicht verlieren. Wie bereits in Abschnitt 5.3 gezeigt, reicht es nicht eine randomisierte Darstellung der Punkte zu wählen, daher sind die Methoden 1-6 alle anfällig gegen RPA, als auch ZPA. Dies impliziert  $AR_R = AR_Z = 1$ . Durch eine Randomisierung des Exponentes können RPA und ZPA abgewehrt werden. Methode 7, 8 und 10 als auch BRIP sichern auf diesem Wege gegen RPA und ZPA. Da bei diesen Abwehrtechniken der Zufallswert, der in die Randomisierung einfließt, ein 160-Bit Wert ist, liegt die Wahrscheinlichkeit für einen Angreifer diesen Wert zu erraten bei  $1/2^{160}$ . Dies impliziert  $AR_R = AR_Z = 2^{-160}$  für diese Techniken. Das „Randomisierte Startpunkt“-Verfahren (Methode 9) splittet die Berechnung in zwei Teile, abhängig vom randomisierten Startpunkt  $r$ . Für die höherwertigen Bits  $d_{n-1}, \dots, d_r$  erfolgt die Berechnung mittels MSB und für die niederwertigen Bits  $d_{r-1}, \dots, d_0$  erfolgt die Berechnungen mittels LSB. Da die Wahrscheinlichkeit diesen Startpunkt zu erraten bei  $1/160$  liegt, ist die Sicherheit sehr gering. Dies impliziert  $AR_R = AR_Z = 1/160$ . Dass Methode 11, die Randomisierte Adressmethode lediglich gegen ADPA absichert, wurde bereits in Abschnitt 4.3 erklärt.

**BRIP** und seine Erweiterungen EBRIP und WRIP stellen Gegenmaßnahmen dar, die erfolgreich gegen RPA und ZPA abwehren. Zur Vereinfachung nehmen wir lediglich BRIP in die Tabelle auf. Wie bereits in Abschnitt 5.3 beschrieben, randomisiert BRIP die Zwischenwerte während der Hauptberechnung. Dadurch sichert BRIP gegen DDPA, RPA und ZPA. ( $\rightarrow AR_D = AR_R = AR_Z = 0$ ). Da in jedem Berechnungsschritt, sowohl eine ECDBL, als auch eine ECADD Operation ausgeführt wird, sichert BRIP ebenfalls gegen

## 7 Vergleich aller Gegenmaßnahmen

SPA ( $\rightarrow AR_S = 0$ ). Da in den Schritten 17.4 und 17.6 von Algorithmus 17 eine bitabhängige Zuweisung der Register erfolgt, sichert BRIP nicht gegen ADPA ab ( $\rightarrow AR_A = 1$ ). Es werden in jedem Schleifendurchlauf eine ECADD und eine ECDBL Operation ausgeführt ( $\rightarrow N_D = N_A = 160$ ). Ferner werden drei Register zur Speicherung der Punkte benötigt ( $\rightarrow R_P = 3$ ).

**Eine Exponentenrekodierung** vor der Hauptberechnung (vergl. Abschnitt 6.1) stellt ebenfalls eine Sicherheitsgefährdung dar, sofern zur Rekodierung die Widt-w NAF Methode bzw. die Fractional Window Methoden eingesetzt werden. Da sich diese Sicherheitslücke einfach durch Vermeidung dieser Rekodierungsverfahren schließen läßt, wird diese Angriffstechnik nicht in die Gesamttabelle hinzugefügt.

**SBPA** stellt, wie in Abschnitt 6.2, beschrieben eine neue Angriffstechnik dar, die einen gänzlich anderen Ansatz verfolgt, als die bisher bekannten Angriffstechniken. Durch das parallele Mitlaufen eines Lauschprozesses während der Hauptberechnung, kann ein Angreifer den Fortgang an allen markanten Sprungbefehlen eines Maschinenbefehls nachvollziehen. Durch diese Angriffstechnik werden Annahmen widerlegt, die die Grundlage für die Methoden 3-11 darstellen. SPA Gegenmaßnahmen basieren auf dem Ausbalancieren von verzweigten Befehlen (Methode 3+4). Blinding- bzw. Randomisierungsverfahren stellen die Grundlage für die Methoden 4b-11 dar. SBPA würde alle genannten Abwehrtechniken aushebeln.

**Gesamttabelle** Durch die Hinzunahme der neuen Angriffstechniken, RPA, ZPA und SBPA, sowie der Gegenmaßnahme BRIP lassen sich neue Kombinationen entwickeln, die in der Gesamttabelle als „Günstige Kombinationen“ bezeichnet werden. Da SBPA auf einer anderen Vorgehensweise basiert, im Vergleich zu allen anderen Angriffstechniken, kann sie durch keine der genannten Gegenmaßnahmen, auch nicht durch mögliche Kombinationen unschädlich gemacht werden. Daher reduzieren wir die Betrachtungen für die kombinierten Gegenmaßnahmen A, B und C auf die übrigen Angriffstechniken und tragen in die Spalte  $ASPBA$  ein „-“ ein.

**Neue Kombinationen** Um eine sichere Gegenmaßnahme zu wählen, nehmen wir als Standardalgorithmus Methode 4b. In Abschnitt 5.3 wurde bereits beschrieben, dass BRIP gegen die hinzugefügten Angriffstechniken RPA und ZPA, sowie gegen DDPA, absichert. Weiterhin sichert BRIP gegen SPA ab, da in jedem Schleifendurchlauf von Algorithmus 17 eine feste, schlüsselunabhängige Abfolge von ECDBL und ECADD-Operationen stattfindet. BRIP sichert jedoch nicht gegen ADPA, daher ist eine Kombination mit den Methoden 9, 11 oder 8 günstig. Eine Kombination mit Methode 10 würde die Anzahl der ECDBL und ECADD Operationen verdoppeln, welches eine enorme Effizienzeinbuße nach sich ziehen würde. Methode A stellt eine Kombination von Methode 4b und Methode 9 dar. Die Bestimmung der kombinierten Werte für die Rechengeschwindigkeit, dargestellt durch die Anzahl der Operationen, und der Werte für den Speicherbedarf, dargestellt durch die Anzahl der benötigten Register, stellt sich einfach dar. Das Modell

## 7 Vergleich aller Gegenmaßnahmen

von Itoh, Izu und Takenaka gibt vor, die  $N$ -Werte zu multiplizieren, und die  $R$ -Werte zu addieren. Während man den kombinierten  $AR_S$ -Wert ebenfalls durch Multiplikation erhält, muss man bei den übrigen  $AR$ -Werten überlegen, wie sich die Gegenmaßnahmen gegenseitig beeinflussen. Im Fall von Methode A, die BRIP mit dem Randomisiertem Startpunkt kombiniert, beobachtet man dass beide Randomisierungen sich nicht beeinflussen. Während BRIP :=  $(dP+R)$ -R den Gesamtterm  $dP$  randomisiert, beschränkt sich die Randomisierte Startpunkt Methode auf den Exponenten  $d$ . Beide Randomisierungen benutzen einen eigenen Zufallswert, der auf den anderen Wert keinen direkten Einfluß übt. In diesem Fall fließen zwei 160-Bit Zufallswerte ein, die jeweils für sich einen  $AR$ -Wert  $AR = 2^{-160}$  zur Folge hätten. Bei einer Kombination von Methode 4b und 9 ist davon auszugehen, dass sich dieser Wert nicht verschlechtert, also höchstens den Wert  $AR = 2^{-160}$  annimmt. Dies wird in der Gesamttabelle mit dem Ausdruck „ $\ll 2^{-160}$ “ wiedergegeben. Die kombinierten Werte für Methode A ergeben dann

- $AR_D = AR_R = AR_Z = \ll 2^{-160}$
- $AR_S = 0, AR_A = 0$
- $N_D = N_A = 160, R_P = 3, R_S = 0$
- $A_{SBPA} = 1$  (wie für alle Methoden)

Methode A, B und C sichern gegen SPA, DDPA, RPA, ZPA und ADPA ab. Sie unterscheiden sich lediglich hinsichtlich ihrer Rechengeschwindigkeit und ihres Speicherverbrauches. Tabelle 7.2 zeigt die Gesamtaufstellung der Gegenmaßnahmen und Angriffstechniken.

Nr.	Methoden	$AR_S$	$AR_D$	$AR_R$	$AR_Z$	$AR_A$	$AR_{SPA}$	$N_D$	$N_A$	$R_P$	$R_S$
1	Binäre Methode (MSB)	1	1	1	1	1	1	160	80	1	0
2	Binäre Methode (LSB)	1	1	1	1	1	1	160	80	2	0
3	Add-and-double-always Methode	0	1	1	1	1	1	$\times 1$	$\times 2$	+1	+0
4	Montgomery Ladder	0	1	1	1	1	1	60	160	3	0
4b	BRIP	0	$2^{-160}$	$2^{-160}$	$2^{-160}$	1	1	160	160	3	0
5	Randomisierte Projektive Koordinaten (RPC)	1	$2^{-160}$	1	1	1	1	$\times 1$	$\times 1$	+1	+1
6	Randomisierte Kurve (RC)	1	$2^{-160}$	1	1	1	1	$\times 1$	$\times 1$	+0	+3
7	Randomisierung des Skalars	1	$2^{-160}$	$2^{-160}$	$2^{-160}$	1	1	$\times 2$	$\times 2$	+2	+0
8	Randomisierter Exponent	1	$2^{-160}$	$2^{-160}$	$2^{-160}$	0	1	$\times 1.13$	$\times 1.13$	+0	+0
9	Randomisierter Startpunkt	1	$\frac{1}{160}$	$\frac{1}{160}$	$\frac{1}{160}$	0	1	$\times 1$	$\times 1$	+0	+0
10	Exponentensplitting	1	$2^{-160}$	$2^{-160}$	$2^{-160}$	0	1	$\times 2$	$\times 2+1$	+1	+2
11	Randomisierte Adressmethode (RA)	1	1	1	1	0	1	$\times 1$	$\times 1$	+0	+2
A	Günstige Kombination 4b+9	0	$\ll 2^{-160}$	$\ll 2^{-160}$	$\ll 2^{-160}$	0	-	160	160	3	0
B	Günstige Kombination 4b+11	0	$\ll 2^{-160}$	$\ll 2^{-160}$	$\ll 2^{-160}$	0	-	160	160	3	2
C	Günstige Kombination 4b+8	0	$\ll 2^{-160}$	$\ll 2^{-160}$	$\ll 2^{-160}$	0	-	181	181	3	0

Tabelle 7.2: Gesamttabelle der Gegenmaßnahmen

# Literaturverzeichnis

- [AHLM03] AHN, MahnKi ; HA, JaeCheol ; LEE, Hoon J. ; MOON, Sang-Jae: A Random M-ary Method Based Countermeasure against Side Channel Attacks. In: *ICCSA (2)*, 2003, S. 338–347
- [AKKS07] ACHIÇMEZ, Onur ; KAYA KOÇ Çetin ; SEIFERT, Jean-Pierre: On the power of simple branch prediction analysis. In: *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*. New York, NY, USA : ACM, 2007. – ISBN 1–59593–574–6, S. 312–320
- [AT03] AKISHITA, Toru ; TAKAGI, Tsuyoshi: Zero-Value Point Attacks on Elliptic Curve Cryptosystem. In: *ISC*, 2003, S. 218–233
- [Ava03] AVANZI, Roberto M.: Countermeasures against Differential Power Analysis for Hyperelliptic Curve Cryptosystems. In: *CHES*, 2003, S. 366–381
- [Ava05] AVANZI, R.: *Side channel attacks on implementations of curve-based cryptographic primitives*. [citeseer.ist.psu.edu/avanzi05side.html](http://citeseer.ist.psu.edu/avanzi05side.html). Version: 2005
- [BDL97] BONEH, Dan ; DEMILLO, Richard A. ; LIPTON, Richard J.: On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In: *EUROCRYPT*, 1997, S. 37–51
- [BJ] BRIER, Eric ; JOYE, Marc: *Weierstraß Elliptic Curves and Side-Channel Attacks*. [citeseer.ist.psu.edu/article/brier02weierstra.html](http://citeseer.ist.psu.edu/article/brier02weierstra.html)
- [Blu05] BLUEMEL, R.: *Algorithmische Randomisierung bei asymmetrischen Kryptographieverfahren*. 2005
- [BNL] B. NIKOLIC, V. Stojanovic W. Jia J. C. V. G. Oklobdzija O. V. G. Oklobdzija ; LEUNG, M.: Improved Sense-Amplifier-Based Flip-Flop: Design and Measurements. In: *IEEE J. Solid-State Circuits* Bd. 35
- [Buc02] BUCHMANN, Johannes: *Einführung in die Kryptographie*. Springer, 2002. – ISBN 0–387–95034–6
- [CAM<sup>+</sup>02] CUNNINGHAM, Paul ; ANDERSON, Ross ; MULLINS, Robert ; TAYLOR, George ; MOORE, Simon: Improving Smart Card Security Using Self-Timed Circuits. In: *ASYNC '02: Proceedings of the 8th International Symposium on Asynchronous Circuits and Systems*. Washington, DC, USA : IEEE Computer Society, 2002. – ISBN 0–7695–1540–1, S. 211

## Literaturverzeichnis

- [CJ01] CLAVIER, Christophe ; JOYE, Marc: Universal Exponentiation Algorithm. In: *CHES*, 2001, S. 300–308
- [CMO98] COHEN ; MIYAJI ; ONO: Efficient Elliptic Curve Exponentiation Using Mixed Coordinates. In: *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*, LNCS, Springer-Verlag, 1998
- [Cor99] CORON, Jean-Sébastien: Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In: *CHES*, 1999, S. 292–302
- [Eck04] ECKERT, Claudia: *IT-Sicherheit*. München [u.a.] : Oldenbourg, 2004. – ISBN 3–486–20000–3
- [EW93] ELDRIDGE, Stephen E. ; WALTER, Colin D.: Hardware Implementation of Montgomery’s Modular Multiplication Algorithm. In: *IEEE Trans. Computers* 42 (1993), Nr. 6, S. 693–699
- [Fri22] FRICKE, R.: *Die elliptischen Funktionen und ihre Anwendungen*. B.G. Teubner, 1922
- [Gou03] GOUBIN, Louis: A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems. In: *Public Key Cryptography*, 2003, S. 199–210
- [HM02] HA, JaeCheol ; MOON, Sang-Jae: Randomized Signed-Scalar Multiplication of ECC to Resist Power Attacks. In: *CHES*, 2002, S. 551–563
- [Hof93] HOFFMAN, R.: *Rechnerentwurf*. Oldenbourg, 1993
- [H.S86] H.SILVERMANN, Joseph: *The Arithmetic of Elliptic Curves*. Springer, 1986. – ISBN 0387962034
- [IIT03] ITOH, Kouichi ; IZU, Tetsuya ; TAKENAKA, Masahiko: A Practical Countermeasure against Address-Bit Differential Power Analysis. In: *CHES*, 2003, S. 382–396
- [JP] JEAN-PIERRE, Onur A.: *Predicting Secret Keys via Branch Prediction*. [citeseer.ist.psu.edu/758468.html](http://citeseer.ist.psu.edu/758468.html)
- [JPS05] JOYE, Marc ; PAILLIER, Pascal ; SCHOENMAKERS, Berry: On Second-Order Differential Power Analysis. In: *CHES*, 2005, S. 293–308
- [JT01] JOYE, Marc ; TYMEN, Christophe: Protections against Differential Analysis for Elliptic Curve Cryptography. In: *CHES*, 2001, S. 377–390
- [Kat93] KATZ, Randy H.: *Contemporary logic design*. Redwood City, CA, USA : Benjamin-Cummings Publishing Co., Inc., 1993. – ISBN 0–8053–2703–7
- [Koc96] KOCHER, Paul C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: *CRYPTO*, 1996, S. 104–113

## Literaturverzeichnis

- [MDS99] MESSERGES, Thomas S. ; DABBISH, Ezzy A. ; SLOAN, Robert H.: Power Analysis Attacks of Modular Exponentiation in Smartcards. In: *CHES*, 1999, S. 144–157
- [Mes00] MESSERGES, Thomas S.: Using Second-Order Power Analysis to Attack DPA Resistant Software. In: *CHES*, 2000, S. 238–251
- [MMM04] MAMIYA, Hideyo ; MIYAJI, Atsuko ; MORIMOTO, Hiroaki: Efficient Countermeasures against RPA, DPA, and SPA. In: *CHES*, 2004, S. 343–356
- [MOV96] MENEZES, Alfred ; OORSCHOT, Paul C. ; VANSTONE, Scott A.: *Handbook of Applied Cryptography*. CRC Press, 1996. – ISBN 0–8493–8523–7
- [OKS00] OKEYA, Katsuyuki ; KURUMATANI, Hiroyuki ; SAKURAI, Kouichi: Elliptic Curves with the Montgomery-Form and Their Cryptographic Applications. In: *Public Key Cryptography*, 2000, S. 238–257
- [Ols04] OLSON, Loren D.: Side-Channel Attacks in ECC: A General Technique for Varying the Parametrization of the Elliptic Curve. In: *CHES*, 2004, S. 220–229
- [OS00] OKEYA, Katsuyuki ; SAKURAI, Kouichi: Power Analysis Breaks Elliptic Curve Cryptosystems even Secure against the Timing Attack. In: *INDOCRYPT*, 2000, S. 178–190
- [Rab] RABAEY, J.: *Digital Integrated Circuits: A design perspective*
- [Rel96] RELEASE, Bellcore P.: *New threat model breaks crypto codes*. 1996
- [Sha99] SHAMIR, Adi: *Method and apparatus for protecting public key schemes from timing and fault attacks*. United States Patent 5991415, 1999
- [Sha00] SHAMIR, Adi: Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies. In: *CHES*, 2000, S. 71–77
- [Sha97] SHAMIR, Adi: How to check modular exponentiation. In: *Presented at the rump session of EUROCRYPT, '97*
- [SS04] SAKAI, Yasuyuki ; SAKURAI, Kouichi: A New Attack with Side Channel Leakage During Exponent Recoding Computations. In: *CHES*, 2004, S. 298–311
- [TAV02] TIRI, K. ; AKMAL, M. ; VERBAUWHEDE, I.: *A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards*. [citeseer.ist.psu.edu/tiri02dynamic.html](http://citeseer.ist.psu.edu/tiri02dynamic.html). Version: 2002
- [Wal02] WALTER, Colin D.: MIST: An Efficient, Randomized Exponentiation Algorithm for Resisting Power Analysis. In: *CT-RSA*, 2002, S. 53–66

*Literaturverzeichnis*

- [Wal04] WALTER, Colin D.: Simple Power Analysis of Unified Code for ECC Double and Add. In: *CHES*, 2004, S. 191–204