

Diplomarbeit

Kombinierte Schwachstellenanalyse für schnelle Exponentiationen

September 2007



Mustafa Gökhan Sögüt

Technische Universität Darmstadt

Betreuer:

Prof. Dr. Johannes A. Buchmann,
Dipl.-Inf. Daniel Schepers

Fachbereich Informatik

Theoretische Informatik - Kryptographie und Computergebra

Ich danke Herrn Prof. Johannes Buchmann für seine stets unterhaltsamen und lehrreichen Vorlesungsstunden. Meine Diplomarbeit an seinem Lehrstuhl verfassen zu dürfen, ist für mich eine große Ehre.

Mein besonderer Dank gilt Daniel Schepers. Ohne Ihn wäre diese Arbeit nicht zustande gekommen. Seine Betreuung und Hilfsbereitschaft war vorbildlich.

Darüber hinaus danke ich meiner Familie, meinen Freunden und Kommilitonen, ohne die ich dieses Studium nicht gemeistert hätte.

Schlussendlich gilt mein größter Danke meiner Freundin, die mich unterstützt und immerzu motiviert.

Mustafa Gökhan Sögüt
01. September 2007

Abstract

Elliptische-Kurven-Kryptosysteme (EKK) [Kob87] sind sehr gut für kleine Geräteeinheiten, wie SmartCards, geeignet. Da diese Geräteeinheiten in ihrer Speicher- und Rechenleistung stark begrenzt sind, nutzen Sie den Vorteil der Speichereffizienz, welches ein EKK bietet: Ein 160bit großer EKK Schlüssel weist dieselbe Sicherheit wie ein 1024bit großer RSA Schlüssel auf und die benötigte Schlüssellänge ist im Vergleich zu RSA kürzer [Buc04]. Diese Art von Verfahren verringert dadurch die Rechenzeit und die Übertragung des Geheimnisses ist um ein Drittel schneller gegenüber RSA. Hingegen muss mit dem Problem der effizienten Skalarmultiplikation beim Einsatz von EKK umgegangen werden. Die Skalarmultiplikation stellt die wichtigste Operation in Bezug auf die physischen Grenzen kleiner Geräteeinheiten dar. Die bekanntesten Rekodierungsalgorithmen zur Effizienzsteigerung der Skalarmultiplikation sind beispielsweise die *Sliding Window Exponentiation* [MvOV97] und NAF bzw. w -NAF [Sol00].

Des Weiteren müssen EKK sicher gegenüber Seitenkanalangriffen sein. Zahlreiche Gegenmaßnahmen zur Abwehr dieser sind vorhanden [Cor99],[OT03b],[OT03a].

In dieser Arbeit wird eine neue Art des Seitenkanalangriffes vorgestellt. Es wird die Kombination aus den bisher bekannten Seitenkanalangriffen (TA, SPA, DPA) und den Rekodierungsalgorithmen (w -NAF) zur Schwachstellenanalyse verwertet. Diese Kombination kann den Angreifer mit wertvollen Informationen über den geheimen Exponenten versorgen. Darüber hinaus werden Grundlagen und Möglichkeiten zur Effizienzsteigerung aufgezeigt.

Inhaltsverzeichnis

1	Einleitung	5
2	Elliptische Kurven	7
2.1	Gruppe	7
2.2	Körper	7
2.2.1	Existenz und Eindeutigkeit	8
2.3	Einführung in Elliptische Kurven	10
2.3.1	Weierstrass Gleichung	10
2.3.2	Einheitliche Weierstrass Gleichung	11
2.3.3	Die elliptische Kurven Gruppe	12
2.3.4	Punktrepräsentation in Projektiven Koordinaten	16
3	Einsatz elliptischer Kurven in Kryptosystemen	18
3.1	Funktionsprinzip	19
3.2	Das Diskrete Logarithmus Problem	19
3.3	Schlüsseldaten der elliptischen Kurve	19
3.3.1	Bereichsparameter	20
3.3.2	Sicherheitsbedingungen	21
3.4	Randomisierung	21
4	Effizienz	23
4.1	Einfach Punktmultiplikation	23
4.1.1	Square and Multiply	23
4.1.2	Fixed-size-window Exponentiation	24
4.1.3	Sliding Window Exponentiation	24
4.1.4	Exponentiation mit Non-adjacent forms	25
4.1.5	Spezialfall: ± 1 -Ketten	27
4.1.6	LimLee Exponentiation	28
4.1.7	<u>Komplexitätsvergleich und Analyse</u>	29
4.2	Mehrfache Punktmultiplikation	30
4.2.1	Simultane Exponentiation	31
4.2.2	Simultane 2^w -ary Exponentiation	31
4.2.3	Simultane Sliding-Window Exponentiation	33
4.2.4	Basic Interleaving Exponentiation	34
4.2.5	w -NAF-based Interleaving Exponentiation	35
4.2.6	<u>Komplexitätsvergleich und Analyse</u>	36

5	Angriffe	37
5.1	Die Exponentenrekodierung als wertvolle Informationsquelle	37
5.1.1	NAF	37
5.1.2	w -NAF	38
5.1.3	Fractional Window	39
5.1.4	Das Informationsleck: Exponentenrekodierung	41
5.1.5	Der Angreifer	41
5.1.6	Das Leck	41
5.1.7	Der Angriff	42
6	Kombinierte Schwachstellenanalyse	50
6.1	Konzept	50
6.1.1	Das Szenario I	53
6.1.2	Das Szenario II	55
6.1.3	Das Szenario III	58
6.2	Ergebnis	60
7	Resümee	62

Kapitel 1

Einleitung

SmartCards gehören zu einem bedeutenden Anwendungsfeld kryptographischer Algorithmen und beinhalten sensitive Daten, wie zum Beispiel den privaten EKK Schlüssel. Einige Implementierungen kryptographischer Algorithmen lassen Seitenkanalinformationen durchsickern. Zu den Seitenkanalinformationsquellen zählen der Stromverbrauch, elektromagnetische Felder und die Laufzeit. Die Idee der Seitenkanalangriffe, welche Seitenkanalinformationen nutzen, die durch ein ‐Leck‐ tats chlich implementierter kryptographischer Algorithmen ‐austreten‐, wurde zum ersten Mal von Kocher entdeckt und zusammen mit Jaffe und Jun [Koc96a, KuBJ99] weiterentwickelt. Seitenkanalangriffe k nnen h ufig m chtiger sein als die mathematische Kryptoanalyse. Dementsprechend wurden schon viele Arbeiten zur Seitenkanalkryptoanalyse vorgestellt [Koc96a], [KuBJ99], [AK97], [CKN00], [KR02], [KSWH98].

Elliptische-Kurven-Kryptosysteme [Kob87] ben tigen die Berechnung der skalaren Punktmultiplikation eines Punktes auf einer elliptischen Kurve. Die Performanz kryptographischer Protokolle mit einem Public-Key-Kryptosystem (PKI) h ngt stark von der Effizienz der Skalarmultiplikation ab. Daher ist es w nschenswert Algorithmen bereit zu stellen, welche eine effiziente Anwendung erm glichen. Ein Ansatz zur Reduzierung der Komplexit t der Berechnungen der Skalarmultiplikation ist die Konversion der bin ren Darstellung eines gegebenen Exponenten in eine Darstellung, welche weniger nicht-Null Eintr ge hat. Diese Transformierung des Exponenten von einer in die andere Darstellung wird ‐Exponentenrekodierung‐ genannt.

Seitdem die Idee der Seitenkanalangriffe zun chst von Kocher, Jaffe und Jun [Koc96a, KuBJ99] vorgestellt worden war, folgten weitere Variationen von Angriffen und Gegenma nahmen [Cor99], [OT03b], [OT03a]. Zum Beispiel stellte Coron, bezogen auf ein Public-Key-Kryptosystem, Gegenma nahmen zur Abwehr von DPA (Differential Power Analysis) auf Elliptischen-Kurven-Kryptosystemen vor [Cor99]: 1) Die *double-and-add-always* Methode, 2) Die Randomisierung: i) Randomisierung des geheimen Exponenten ii) Randomisierung des Basispunktes, iii) Randomisierung der projektiven Koordinaten. Seitenkanalinformationen von kryptographischen Ger teeinheiten k nnen Informationen  ber die stattfindenden Operationen und der involvierten Parameter liefern.

In dieser Arbeit werden wir uns mit Seitenkanalinformationen, verursacht durch kryptographische Ger teeinheiten, n her besch ftigen. Es wird eine neue Art der Gewinnung dieser Informationen vorgestellt. Wir werden kombinierte Angriffsme-

thoden für die w -NAF Rekodierung [MOC97] zeigen.

Der Rest der Arbeit gliedert sich wie folgt: Kapitel 2 gibt eine kurze Einführung in die mathematischen Grundlagen elliptischer Kurven. Kapitel 3 stellt den Einsatz elliptischer Kurven vor und zeigt deren Sicherheitsaspekte auf. Kapitel 4 zeigt verschiedene Möglichkeiten der Effizienzsteigerung mit Hilfe der Exponentenrekodierung. Kapitel 5 betrachtet den Exponentiationsablauf genauer. Hier finden wir Phasen, welche von den Anweisungen des geheimen Exponenten abhängen. Effiziente Exponentiationen müssen im Voraus eine Exponentenrekodierung durchführen, dann erst wird die Hauptphase (Punktmultiplikation und Punktaddition) der Exponentiation vollzogen. Da die Exponentenrekodierung Verzweigungsbedingungen inne hat und diese in Abhängigkeit des geheimen Exponenten ausgeführt werden, kann die Exponentenrekodierung einen Informationskanal mit wertvollen Information für den Angreifer bereitstellen. Hier werden methodische Angriffsweisen auf die Exponentenrekodierung für w -NAF [MOC97] und der vorzeichenlosen/vorzeichenbehafteten fractional window Repräsentation [Moe02] gezeigt. Kapitel 6 stellt die neue Art der Gewinnung von Seitenkanalinformationen vor. Hier wird das Grundkonzept grob erläutert. Anschließend kommt die Idee in verschiedenen Szenarien zum Einsatz. Hier wird auf die Umsetzung des Konzepts für die w -NAF [MOC97] Rekodierung näher eingegangen. Den Abschluss der Arbeit bildet das Kapitel 7 mit dem Resümee.

Kapitel 2

Elliptische Kurven

In diesem Kapitel werden wir eine kurze Einführung und einen groben Überblick über elliptische Kurven bekommen.

2.1 Gruppe

Definition 1.1 (Gruppe)[Bec06]

Eine Gruppe ist auf einer Menge G mit binären Operationen definiert und besitzt folgende Eigenschaften:

$$\begin{aligned} \forall x, y \in G: \quad x \circ y \in G & \quad \text{Abschluss} & (1.1) \\ \forall x, y, z \in G: \quad x \circ (y \circ z) = (x \circ y) \circ z & \quad \text{Assoziativität} & (1.2) \\ \exists e \in G, \forall x \in G: \quad x \circ e = e \circ x = x & \quad \text{Identität} & (1.3) \\ \forall x \in G, \exists z \in G: \quad x \circ z = z \circ x = e & \quad \text{Inverse} & (1.4) \end{aligned}$$

Eine abelsche Gruppe erfüllt zusätzlich:

$$\forall x, y \in G: x \circ y = y \circ x \quad \text{Kommutativität}$$

Eine Gruppe ist endlich, wenn die Menge G endlich ist. Die Anzahl der Elemente in G bezeichnet die Ordnung von G .

Falls eine Gruppe G der Ordnung n ein Element g mit der Ordnung n besitzt, wird G *zyklische Gruppe* genannt und das Element g als Erzeuger von G bezeichnet.

2.2 Körper

Definition 1.2 (Körper)[Bec06]

Ein Körper besteht aus der Menge \mathbb{F} und der zwei Operationen, Addition (+) und Multiplikation (\cdot).

Eigenschaften:

- a.) $(\mathbb{F}, +)$ ist eine abelsche Gruppe mit der additiven Identität 0.
- b.) $(\mathbb{F} \setminus 0, \cdot)$ ist eine abelsche Gruppe mit der multiplikativen Identität 1.
- c.) Distributivität: $\forall a, b, c \in \mathbb{F}: (a + b) \cdot c = (a \cdot c) + (b \cdot c)$

Ein Körper ist endlich, wenn die Menge \mathbb{F} endlich ist.

Die Ordnung eines endlichen Körpers ist die Anzahl der Elemente im Körper.

Die *Charakteristik* eines Körpers \mathbb{F} , angegeben mit $\text{char}(\mathbb{F})$, beschreibt die kleinste, positive und ganze Zahl k mit,

$$\underbrace{1 + \cdots + 1}_{k\text{-mal}} = 0$$

wobei 1, 0 die Identität der Körpermultiplikation und der Körperaddition sind.

Definition 1.3 (Körpererweiterung) [Bec06]

Eine Untermenge K_S eines Körpers K ($K_S \subseteq K$) ist ein Unterkörper von K , falls K_S dieselben Operationen wie K inne hat. Ist das der Fall, wird K als Körpererweiterung von K_S bezeichnet.

2.2.1 Existenz und Eindeutigkeit

Es existiert ein Körper \mathbb{F} der Ordnung q , falls q einzig eine Primzahlpotenz ist, mit $q = p^m$. p ist eine Primzahl und wird *Charakteristik* von \mathbb{F} genannt. m ist eine positive ganze Zahl.

Falls $m = 1$, dann ist \mathbb{F} ein Primkörper.

Falls $m \geq 2$, dann ist \mathbb{F} ein Erweiterungskörper. Für jede Primzahlpotenz q existiert ein eindeutiger Körper der Ordnung q . Das bedeutet, dass zwei beliebige endliche Körper der Ordnung q strukturell gleich sind, aber sich in ihrer Repräsentation unterscheiden. Solche Körper sind Isomorph zueinander und werden mit \mathbb{F}_q gekennzeichnet.

Definition 1.4 (Primkörper \mathbb{F}_p)[Bec06]

p ist eine Primzahl. Die Menge der ganzen Zahlen modulo p , $\{0, 1, 2, \dots, p-1\}$, mit Addition und Multiplikation bezüglich modulo p , bildet einen endlichen Körper der Ordnung p .

Dieser Körper ist ein sogenannter Primkörper gekennzeichnet durch \mathbb{F}_p . Die Primzahl p ist der Modulus von \mathbb{F}_p .

Für jede ganze Zahl a ist $r = a \bmod p$ der eindeutige Rest mit $0 \leq r \leq p-1$, den wir durch die Division von a durch p erhalten. Diese Operation wird auch *Reduktion modulo p* genannt.

Definition 1.5 (Binärer Körper)[Bec06]

Endliche Körper der Ordnung 2^m werden *binäre Körper* bzw. *endliche Charakteristik-2 Körper* genannt.

\mathbb{F}_{2^m} kann konstruiert werden, in dem eine polynomiale Repräsentation gewählt wird:

Die Elemente von \mathbb{F}_{2^m} sind binäre Polynome von maximalem Grad $m - 1$:

$$\mathbb{F}_{2^m} = \{a_0 + a_1x + a_2x^2 + \cdots + a_{m-1}x^{m-1} : a_i \in \{0, 1\}, 0 \leq i \leq m - 1\}. \quad (1.5)$$

Ein irreduzibles Polynom $f(x)$ vom Grad m wird gewählt. Irreduzibel bedeutet, dass $f(x)$ nicht als ein Produkt von binären Polynomen, mit einem Grad kleiner als m , faktorisiert werden kann.

Die Addition von Körperelementen ist eine gewöhnliche Addition von Polynomen mit dem arithmetischem Mittel modulo 2. Die Multiplikation der Körperelemente erfolgt mittels modulo Reduktionspolynom $f(x)$. Für jedes beliebige Polynom $a(x)$, ist $r(x) = a(x) \bmod f(x)$ das eindeutige Restpolynom. Dieses ist vom Grad kleiner als das durch die Division, $a(x)$ durch $f(x)$, gewonnene m . Diese Operation wird *Reduktion modulo $f(x)$* genannt.

Definition 1.6 (Erweiterungskörper \mathbb{F}_{p^m}) [Bec06]

Die Polynomiale Repräsentation für binäre Körper kann für alle Erweiterungskörper generalisiert werden. p sei eine Primzahl und $m \geq 2$. $\mathbb{F}_p[x]$ kennzeichnet die Menge aller Polynome in x mit den Koeffizienten aus \mathbb{F}_p . Sei $f(x)$ das Reduktionspolynom; irreduzibel mit dem Grad m in $\mathbb{F}_p[x]$.

Die Elemente von \mathbb{F}_{p^m} sind Polynome in $\mathbb{F}_p[x]$ vom maximalem Grad $m - 1$:

$$\mathbb{F}_{p^m} = \{a_0 + a_1x + a_2x^2 + \cdots + a_{m-1}x^{m-1} : a_i \in \mathbb{F}_p, 0 \leq i \leq m - 1\}. \quad (1.6)$$

Die Addition der Körperelemente ist eine gewöhnliche Addition von Polynomen mit arithmetischen Koeffizienten in \mathbb{F}_p . Die Multiplikation der Körperelemente erfolgt mittels modulo Reduktionspolynom $f(x)$.

Theorem 1.1 (Unterkörper von \mathbb{F}_{p^m}) [Bec06]

Ein endlicher Körper \mathbb{F}_{p^m} hat genau einen Unterkörper der Ordnung p^l für jeden positiven Divisor l aus m . Die Elemente dieses Unterkörpers sind die Elemente $a \in \mathbb{F}_{p^m}$ mit $a^{p^l} = a$.

Umgekehrt hat jeder Unterkörper \mathbb{F}_{p^m} die Ordnung p^l für einige positive Divisor l aus m .

Theorem 1.2 (\mathbb{F}_p^*)[Bec06]

Die nicht-null Elemente eines endlichen Körpers \mathbb{F}_p , gekennzeichnet durch \mathbb{F}_p^* , bilden eine zyklische Gruppe unter der Multiplikation. Folglich existieren Elemente $b \in \mathbb{F}_p^*$, sogenannte Erzeuger der Form

$$\mathbb{F}_p^* = \{b^i : 0 \leq i \leq p - 2\}. \quad (1.7)$$

Die Ordnung eines Elementes $a \in \mathbb{F}_p^*$ ist die kleinste positive ganze Zahl t mit $a^t = 1$.

Da \mathbb{F}_p^* eine zyklische Gruppe der Ordnung $p - 1$ ist, folgt daraus, dass t ein Divisor von $p - 1$ ist.

2.3 Einführung in Elliptische Kurven

2.3.1 Weierstrass Gleichung

Definition 1.7 (Elliptische Kurve) [Sil86]

Eine elliptische Kurve E über einem Körper K ist definiert durch die Gleichung

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1.8)$$

mit $a_1, a_2, a_3, a_4, a_6 \in K$ und $\Delta \neq 0$. Δ ist die Diskriminante von E und ist wie folgt definiert:

$$\Delta = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \quad (1.9)$$

$$d_2 = a_1^2 + 4a_2 \quad (1.10)$$

$$d_4 = 2a_4 + a_1a_3 \quad (1.11)$$

$$d_6 = a_3^2 + 4a_6 \quad (1.12)$$

$$d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \quad (1.13)$$

Die Menge der Punkte $(x, y) \in K \times K$, die mit dem Punkt im Unendlichen \mathcal{O} die Gleichung (1.8) erfüllen werden beschrieben durch

$$E(K) = \{(x, y) \in K \times K : y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0\} \cup \{\mathcal{O}\}.$$

Hinweis:

- Die Gleichung 1.8 heißt *Weierstrass Gleichung*.
- Die Bedingung $\Delta \neq 0$ stellt sicher, dass die elliptische Kurve gleichmässig ist. Das heißt es gibt keine Punkte an denen die Kurve zwei oder mehrere verschiedene Tangenten hat.
- E , auch $E(a_1, a_2, a_3, a_4, a_6)$, ist definiert über K , weil die Koeffizienten a_1, a_2, a_3, a_4, a_6 der definierten Gleichung von E Elemente aus K sind.
- Da E über K definiert ist, ist E auch über jeden Erweiterungskörper von K definiert.

Die Abbildungen 2.1 und 2.2 zeigen Beispiele elliptischer Kurven.

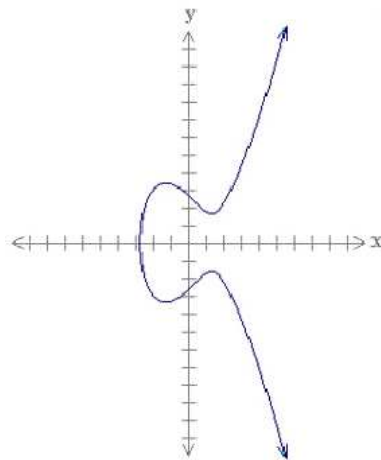


Abbildung 2.1: $E: y^2 = x^3 - 5x + 7$

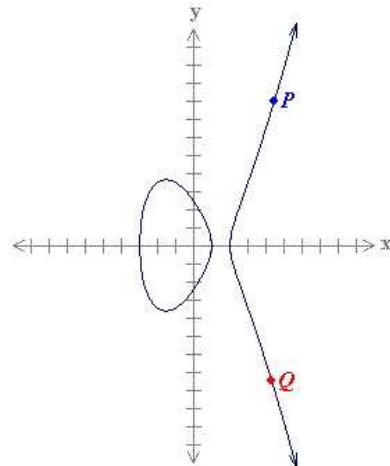


Abbildung 2.2: $E: y^2 = x^3 - 7x + 6$

Definition 1.8 (E_1 und E_2 sind isomorph) [Bec06]

Zwei elliptische Kurven E_1 und E_2 über K mit den gegebenen Weierstrass Gleichungen

$$E_1 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1.14)$$

$$E_2 : y^2 + \bar{a}_1xy + \bar{a}_3y = x^3 + \bar{a}_2x^2 + \bar{a}_4x + \bar{a}_6 \quad (1.15)$$

sind Isomorph über K , falls $u, r, s, t \in K, u \neq 0$ mit

$$(x, y) \mapsto (u^2x + r, u^3y + u^2sx + t) \quad (1.16)$$

die Gleichung E_1 in die Gleichung E_2 transformiert.

2.3.2 Einheitliche Weierstrass Gleichung

Die Weierstrass Gleichung

$$E_1 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1.17)$$

ist definiert über K und kann vereinfacht werden. Wir berücksichtigen hier verschiedene Fälle, bei denen der zugrunde liegende Körper Charakteristika von ungleich 2 und 3 bzw. gleich 2 oder 3 aufweist. [Bec06]

1. Falls die Charakteristik von K ungleich 2 oder 3 ist, dann lässt sich die Gleichung (1.17) transformieren zu

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

wobei $a, b \in K$ mit der Diskriminante $\Delta = -16(4a^3 + 27b^2)$.

2. Falls die Charakteristik von K gleich 2 und $a_1 \neq 0$ ist, dann lässt sich die Gleichung (1.17) überführen zu

$$y^2 + xy = x^3 + ax^2 + b$$

wobei $a, b \in K$ mit der Diskriminante $\Delta = b$. Solch eine Kurve wird *nicht supersingulär* genannt. Falls $a_1 = 0$, dann kann die Gleichung (1.17) überführt werden zu

$$y^2 + cy = x^3 + ax + b$$

wobei $a, b \in K$ mit der Diskriminante $\Delta = c^4$. Solch eine Kurve wird *supersingulär* genannt.

3. Falls die Charakteristik von K gleich 3 und $a_1^2 \neq -a_2$ ist, dann lässt sich die Gleichung (1.17) transformieren zu

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

wobei $a, b \in K$ mit der Diskriminante $\Delta = -a^3b$. Solch eine Kurve wird *nicht supersingulär* genannt. Falls $a_1^2 = -a_2$, dann lässt sich die Gleichung (1.17) transformieren zu

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

wobei $a, b \in K$ mit der Diskriminante $\Delta = -a^3$. Solch eine Kurve wird als *supersingulär* bezeichnet.

2.3.3 Die elliptische Kurven Gruppe

Die Menge der Punkte über E bildet eine abelsche Gruppe unter der Addition mit \mathcal{O} , dem Punkt im Unendlichen. Diese Menge wird $E(K)$ bezeichnet.

2.3.3.1 Punktaddition

Sei E eine elliptische Kurve über dem Körper K . Die Punktaddition lässt sich am besten anhand einer geometrischen Verbildlichung erklären: Seien $P = (x_1, y_1)$ und $Q = (x_2, y_2)$ zwei verschiedene Punkte auf E . Die Summe R , aus P und Q , ist dann wie folgt definiert: Zuerst wird eine Linie durch P und Q gezeichnet. Diese Linie schneidet die elliptische Kurve im dritten Punkt R' . Außerdem ist R die Spiegelung von R' an der x -Achse. Zu sehen ist dies in der Abbildung 2.3. Die Formale Beschreibung ist wie folgt.

Definiton 1.9 (Punktaddition) [Bec06]

Seien $P = (x_1, y_2)$ und $Q = (x_2, y_2)$ zwei Punkte auf E verschieden von \mathcal{O} .

Falls $x_1 = x_2$ und $y_1 + y_2 + a_1x_2 + a_3 = 0$, dann $P + Q = \mathcal{O}$.

Andernfalls $P + Q = (x_3, y_3)$ wobei

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2, \quad (1.18)$$

$$y_3 = -y_1 - (x_3 - x_1)\lambda - a_1x_3 - a_2, \quad (1.19)$$

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & \text{if } P = Q \end{cases} \quad \begin{matrix} (1.20) \\ (1.21) \end{matrix}$$

Zum Schluss definieren wir noch $P + \mathcal{O} = \mathcal{O} + P = P$ für alle $P \in E(K)$. Das Inverse von $P = (x, y) \in E(K)$ ist $(x, -y - a_1x - a_3)$.

Die Addition der Punkte auf einer elliptischen Kurve E über dem endlichen Körper K mit der Charakteristik ungleich 2 oder 3 ist wie folgt definiert.

Definition 1.10 (Punktaddition $E : y^2 = x^3 + ax + b$, $\text{char} \neq 2, 3$) [Bec06]

Sei E eine elliptische Kurve über dem endlichen Körper K nach der *Einheitlichen Weierstrass Gleichung*

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

mit der Charakteristik ungleich 2 oder 3. Sei $P = (x_1, y_1) \in E(K)$ und $Q = (x_2, y_2) \in E(K)$ wobei $P \neq \pm Q$. Ausserdem $P + Q = (x_3, y_3)$ mit

$$x^3 = \left(\frac{y_2 - y_1}{x_2 - y_1}\right)^2 - x_1 - x_2 \quad (1.22)$$

$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right) (x_1 - x_3) - y_1. \quad (1.23)$$

Wir definieren $P + \mathcal{O} = \mathcal{O} + P = P$ für alle $P \in E(K)$. Das inverse von $P \in E(K)$ ist $-P = (x, -y)$.

2.3.3.2 Punktverdoppelung

Um einen Punkt P mit sich selber zu addieren, wird eine Tangente an die Kurve durch den Punkt P gezeichnet. Falls yP ungleich \mathcal{O} ist, dann schneidet die Tangente die elliptische Kurve exakt an einem anderen Punkt $-R$. $-R$ ist die Spiegelung von R entlang der x -Achse. Diese Operation wird Punktverdoppelung von P genannt. Siehe dazu Abbildung 2.4.

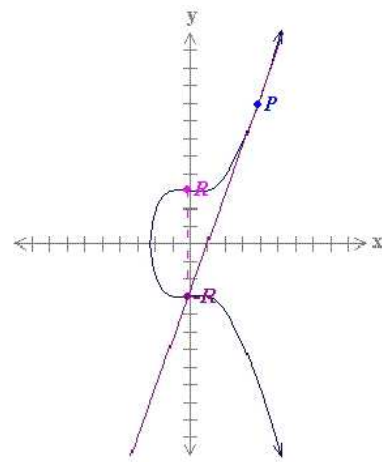
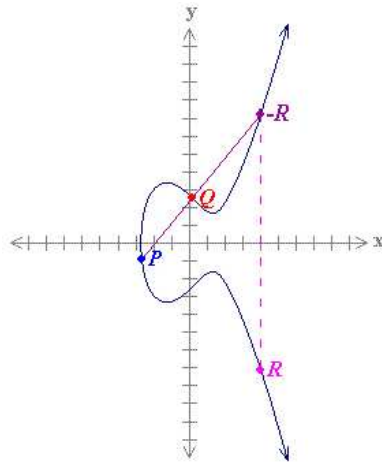


Abbildung 2.3: Punktaddition: $P \neq Q$ Abbildung 2.4: Punktverdoppelung von P

Die Verdoppelung von Punkten auf einer elliptische Kurve E über einem endlichen Körper K mit der Charakteristik ungleich 2 oder 3 ist wie folgt definiert.

Definition 1.11 (Punktverdoppelung $E : y^2 = x^3 + ax + b$, $\text{char} \neq 2,3$)
 [Bec06]

Sei E eine elliptische Kurve über einem endlichen Körper K nach der *Einheitlichen Weierstrass Gleichung*

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

mit der Charakteristik ungleich 2 oder 3. Sei $P = (x_1, y_1) \in E(K)$ mit $P \neq -P$. Dann ist $2P = (x_3, y_3)$ mit

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1} \right)^2 - x_1 - x_2 \quad (1.24)$$

$$y_3 = \left(\frac{3x_1^2 + a}{2y_1} \right) (x_1 - x_3) - y_1. \quad (1.25)$$

2.3.3.3 Gruppenordnung

Definition 1.12 (Gruppenordnung) [Bec06]

Sei E eine elliptische Kurve über K . Die Anzahl der Punkte in $E(K)$, gekennzeichnet durch $\#E(K)$, wird Ordnung von E über K genannt.

Da die Weierstrass Gleichung (1.8) höchstens zwei Lösungen für jedes $x \in \mathbb{F}_q$ hat

(da E symmetrisch zur x -Achse ist), wissen wir, dass $\#E(\mathbb{F}_q) \in [1, 2q + 1]$. Das Hasse Theorem liefert uns schärfere Grenzen:

Theorem 1.3 (Hasse Theorem)

Sei K eine elliptische Kurve über \mathbb{F}_q . Dann gilt

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}. \quad (1.26)$$

Das Intervall $[q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$ wird als Hasse Intervall bezeichnet. Der Beweis dieses Theorems ist in [Sil86] nachzulesen.

Im Folgenden sehen wir die Vorgehensweise zur Bestimmung der möglichen Werte für $\#E(\mathbb{F}_q)$.

Theorem 1.4 (Zulässige Ordnungen für elliptische Kurven) [Sil86]

Sei $q = p^m$, wobei p die Charakteristik von \mathbb{F}_q ist. Es existiert eine elliptische Kurve E über \mathbb{F}_q mit $\#E(\mathbb{F}_q) = q + 1 - t$ genau dann, wenn einer der folgenden Bedingungen gilt:

1. $t \not\equiv 0 \pmod{p}$ und $t^2 \leq 4q$.
2. m ist ungerade und entweder
 - (a) $t = 0$ oder
 - (b) $t^2 = 2q$ und $p = 2$ oder
 - (c) $t^2 = 3q$ und $p = 3$.
3. m ist gerade und entweder
 - (a) $t^2 = 4q$ oder
 - (b) $t^2 = q$ und $p \not\equiv 1 \pmod{3}$ oder
 - (c) $t = 0$ und $p \not\equiv 1 \pmod{4}$.

Theorem 1.5 [Sil86]

Für eine beliebige Primzahl p und eine ganze Zahl t mit $|t| \leq 2\sqrt{q}$, existiert eine elliptische Kurve E über \mathbb{F}_p mit $\#E(\mathbb{F}_p) = p + 1 - t$.

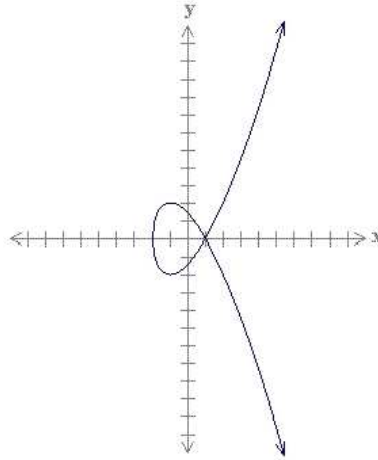
Folglich existiert eine Kurve E über \mathbb{F}_p mit $\#E(\mathbb{F}_p) = p + 1$.

Definition 1.13 (Singularität)

Sei E eine elliptische Kurve und $P = (x, y)$ ein Punkt in E . Dann ist P singularär auf E , falls $\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y} = 0$ ist. Eine singularäre Kurve ist eine Kurve mit mindestens einem singularären Punkt.

Theorem 1.6 (Singularität) [Bec06]

Eine Kurve nach der Weierstrass Gleichung ist singularär genau dann, wenn ihre Diskriminante gleich 0 ist. Die Abbildung 2.5 zeigt eine singularäre Kurve.

Abbildung 2.5: Singuläre Kurve $E: y^2 = x^3 - 3x + 2$ **Definition 1.14 (Zyklische Gruppe)**[Bec06]

Sei E eine elliptische Kurve über einem endlichen Körper \mathbb{F}_q . Sei P ein Punkt in $E(\mathbb{F}_q)$ und wir nehmen an, dass P die Primordnung n hat. Die zyklische Untergruppe von $E(\mathbb{F}_q)$ erzeugt durch P ist $\langle P \rangle = \{\mathcal{O}, P, 2P, \dots, (n-1)P\}$.

2.3.4 Punktrepräsentation in Projektiven Koordinaten

Die Formeln für die Punktaddition und die Punktverdoppelung benötigen innerhalb des Körpers das Inverse und mehrere Multiplikationen. Falls das Inverse im Körper K erheblich teurer ist als die Multiplikation, dann ist es vorteilhaft die Punkte in projektiven Koordinaten anstatt in affinen Koordinaten zu repräsentieren.

Die projektive Form der Weierstrass Gleichung (1.8) einer elliptischen Kurve E über K erhalten wir durch die Ersetzung von x durch X/Z^2 , von y durch Y/Z^3 und durch die Freistzung der Nenner mit $c, d \in \mathbb{N}^{>0}$.

2.3.4.1 Jacobische Koordinaten

Für jacobische Koordinaten [CMO98] setzen wir für $x = X/Z^2$ und für $y = Y/Z^3$ ein. Nun können wir die *Vereinfachte Weierstrass Gleichung* aus (1.3.2) zu

$$E_J : Y^2 = X^3 + aXZ^4 + bZ^6.$$

transformieren.

Außerdem werden zwei Punkte (X, Y, Z) und (r^2X, r^3Y, rZ) als der gleiche Punkt

für einige $r \in_R \mathbb{F}_p^*$ verstanden. Der Punkt im Unendlichen wird mit $(1, 1, 0)$ beschrieben. Für die Addition zweier Punkte P_0 und P_1 auf einer elliptischen Kurve und für die Verdoppelung eines Punktes P_1 kommt die folgende Definition aus [IEE] für uns in Betracht.

Definition 1.15 (Projektive Addition und Verdoppelung)[Bec06]

Sei E eine elliptische Kurve über dem endlichen Körper \mathbb{F}_q nach der *Vereinfachten Weierstrass Gleichung*

$$y^2 = x^3 + ax + b \pmod{p}.$$

mit der Charakteristik ungleich 2 oder 3. E kann transformiert werden zu

$$Y^2 = X^3 + aXZ^4 + bZ^6.$$

Seien $P_0 = (X_0, Y_0, Z_0)$ und $P_1 = (X_1, Y_1, Z_1)$ Punkte in $E(\mathbb{F}_q)$.

Desweiteren ist $P_0 + P_1 = (X_2, Y_2, Z_2)$ wie folgt definiert:

Falls $P_0 = \mathcal{O}$, dann $P_2 = P_1$. Falls $P_1 = \mathcal{O}$, dann $P_2 = P_0$.

Falls $P_0 = -P_1$, dann $P_0 + P_1 = \mathcal{O}$.

Falls $P_1 \neq \pm P_2$, dann $P_0 + P_1 = (X_2, Y_2, Z_2)$ mit

$$\begin{aligned} U_0 &:= X_0 Z_1^2, & S_0 &:= Y_0 Z_1^3, & U_1 &:= X_1 Z_0^2, & S_1 &:= Y_1 Z_0^3, \\ W &:= U_0 U_1, & R &:= S_0 S_1, & T &:= U_0 + U_1, & M &:= S_0 + S_1, \\ Z_2 &:= Z_0 Z_1 W, & X_2 &:= R^2 T W^2, & V &:= T W^2 2 X_2, & 2 Y_2 &:= V R M W^3. \end{aligned}$$

Die projektive Form für die Verdoppelung ist: $2P_1 = (X_2, Y_2, Z_2)$ mit

$$M := 3X_1^2 + aZ_1^4, \quad Z_2 := 2Y_1 Z_1, \quad S := 4X_1 Y_1^2,$$

$$X_2 := M^2 - 2S, \quad T := 8Y_1^4, \quad Y_2 := M(SX_2)T.$$

Angemerkt sei noch, dass der einzige Koeffizient a der elliptischen Kurve für die Verdoppelung genutzt wird.

Bei Speicherung einiger Zwischenergebnisse benötigt eine Addition 16 Körpermultiplikationen und eine Verdoppelung 10 Körpermultiplikationen, wie in [IEE] gezeigt wird.

Kapitel 3

Einsatz elliptischer Kurven in Kryptosystemen

Der Einsatz elliptischer Kurven in der Kryptographie wurde von Neal Koblitz von der Universität Washington im Jahre 1987 [Kob87] und von Victor S. Miller (IBM) im Jahre 1985 [Mil85] als eine alternative für die public key Kryptographie über ganzzahlig basierte zyklische Gruppen vorgeschlagen. Das Elliptische-Kurven-Kryptosystem (EKK bzw. ECC) ist ein asymmetrisches Kryptosystem. Die gängigen asymmetrischen Verschlüsselungsverfahren, beispielsweise RSA-Algorithmen, sind sehr rechenaufwendig. Dieses Problem soll durch EKK gelöst werden. Basis dafür sind elliptische Kurven.

Elliptische Kurven haben die notwendigen mathematischen Eigenschaften für kryptographische Algorithmen und sind besonders geeignet zur effizienten Implementierung, sowohl in Hardware als auch in Software.

Durch die EKK-Verschlüsselung ist es möglich trotz einer geringeren Schlüssellänge von beispielsweise 160 Bit, genauso sicher wie andere asymmetrische Verfahren mit einer Schlüssellänge von 1024 Bit, zu verschlüsseln. Ein EKK eignet sich daher immer dann, wenn die Speicher- oder Rechenkapazität begrenzt ist, zum Beispiel bei Smartcards.

Für einen beliebigen endlichen Körper \mathbb{F}_q gibt es von der Ordnung q verschiedene elliptische Kurven (bis auf Isomorphismen), welche für Kryptosysteme genutzt werden können. Folglich gibt es für einen fixen endlichen Körper mit q Elementen, mit einem großen q , eine große Auswahl für die elliptische Kurvengruppe.

Das Elliptische-Kurven-Kryptosystem (EKK bzw. ECC) basiert auf dem Problem des diskreten Logarithmus, von dem angenommen wird, dass er sicherer gegenüber den ganzzahlig basierten zyklischen Gruppen ist. Deshalb können die Schlüssel in der elliptischen Kurven Kryptographie kürzer sein als bei den ganzzahlig basierten Systemen.

3.1 Funktionsprinzip

Beide Seiten **A** und **B** des zu sichernden Kommunikationskanals einigen sich öffentlich auf eine gültige elliptische Kurve und einen Punkt P auf dieser Kurve. **A** erzeugt eine Zufallszahl a_s . Diese Zahl ist der geheime private Schlüssel von **A**. Analog erzeugt sich **B** seinen privaten Schlüssel b_s . **A** berechnet nun seinen öffentlichen Schlüssel $a_p = a_s \cdot P$. Analog bestimmt **B** seinen öffentlichen Schlüssel $b_p = b_s \cdot P$. Nach der Theorie der elliptischen Kurven liegen beide öffentliche Schlüssel auf der Kurve. Es gilt nun $a_s \cdot b_p = a_s \cdot b_s \cdot P = b_s \cdot a_p$. Damit ist ein Schlüssel gegeben, der nur für **A** und **B** einfach zu berechnen ist und ein öffentlich ausgetauschtes Geheimnis darstellt. Das Geheimnis ist zerstört, wenn aus den öffentlichen Punkten P und a_p bzw. P und b_p die Zufallszahlen a_s oder b_s mit vertretbarem Aufwand berechnet werden können[Ble05].

3.2 Das Diskrete Logarithmus Problem

Das *Elliptische Kurven Diskrete Logarithmus Problem* (ECDLP) ist die Basis aller Elliptischer-Kurven-Kryptosysteme.

Definition 2.1 (Das Elliptische Kurven Diskrete Logarithmus Problem)[Hen03]

Bei gegebener elliptischer Kurve E über einem endlichen Körper \mathbb{F}_q , einem Punkt $P \in E(\mathbb{F}_q)$ der Ordnung n und einem Punkt $Q \in \langle P \rangle$, finde ein $l \in [0, n - 1]$, sodass $Q = lP$. Die ganze Zahl l wird der *diskrete Logarithmus von Q zur Basis P* , mit $l = d\log_P Q$, genannt.

Zur Sicherheit sollte das ECDLP sowohl unlösbar sein, als auch die Parameter (q , die Gleichung von E , P , $n = \text{ord } P$) sollten sorgfältig gewählt worden sein, um allen bekannten Angriffen auf ECDLP stand zu halten.

Definition 2.2 (Starke Elliptische Kurve)[Bec06]

Eine kryptographisch starke elliptische Kurve ist eine elliptische Kurve, sodass das diskrete Logarithmus Problem nur schwer lösbar ist.

Die natürlichste Art das ECDLP zu lösen beruht auf *exhaustive search*, bei der ein Computer die Folge $P, 2P, 3P, \dots$ solange durch exerziert, bis das Ergebnis gleich Q ist. Im schlimmsten Fall benötigt die Berechnung n Schritte und im Durchschnitt $n/2$ Schritte.

Daher ist n ausreichend gross gewählt (zum Beispiel $n \geq 2^{160}$) [X9.99].

3.3 Schlüsseldaten der elliptischen Kurve

Die Schlüsseldaten beinhalten eine Menge an Bereichsparametern und ein elliptisches Kurvenschlüsselpaar.

Definition 2.5 (Elliptisches Kurvenschlüsselpaar)[Bec06]

Ein Schlüsselpaar besteht aus einem öffentlichen Schlüssel Q und einem dazugehörigen privaten Schlüssel d .

Der Private Schlüssel d : Ein privater Schlüssel d ist eine statistisch gesehen einzigartige und unvorhersehbare ganze Zahl im Intervall $[1, n - 1]$ mit n als Primordnung des Basispunktes P .

Der öffentliche Schlüssel: Der zum privaten Schlüssel d dazugehörige öffentliche Schlüssel Q ist der Punkt $Q = dP$ auf der elliptischen Kurve, wobei P der Basispunkt ist. Besonders ist, dass Q nie gleich 2 sein wird, da $1 \leq d \leq n - 1$.

Die Parameter und Schlüssel sollten so gewählt sein, dass das ECDLP resistent gegen alle bekannten Attacken ist. Es gibt einige Standards die gemeine public key Verfahren spezifizieren und die Schlüsselerzeugungssysteme für asymmetrisch kryptographische Verfahren definieren, wie zum Beispiel IEEE P1363 [IEE] und ANSI X9.63 [X9.99].

3.3.1 Bereichsparameter

Die Bereichsparameter für ein Elliptisches-Kurven-Kryptosystem beschreiben eine elliptische Kurve E über einem endlichen Körper \mathbb{F}_q , einen Basispunkt $P \in E(\mathbb{F}_q)$ und dessen Ordnung n :

Definition 2.6 (Bereichsparameter $D = (q, S, a, b, P, n, h)$)[Bec06]

- q ist die Anzahl der Elemente im Körper
- S ist ein optionaler Starwert (*SEED*)
- a, b sind Elemente von \mathbb{F}_q , die eine elliptische Kurve E über \mathbb{F}_q definieren mit $E : y^2 = x^3 + ax + b$
- P ist ein spezieller Punkt der elliptischen Kurve E , welcher der Basispunkt bzw. der Erzeugungspunkt ist
- n gibt die Ordnung des Basispunktes an
- h ist ein Ko-Faktor (Anzahl der Punkte auf E geteilt durch n)

Die Bereichsparameter könnten beispielsweise durch öffentliche Zertifikate bereitgestellt werden.

Die Bereichsparameter sollten bestimmten Bedingungen unterliegen, welche im Abschnitt 3.3.2 aufgelistet werden.

3.3.2 Sicherheitsbedingungen

- Um die Pohlig-Hellmann Attacke und die Pollard- ρ Attacke bezüglich ECDLP zu verhindern, ist es notwendig, dass $\#E(\mathbb{F}_q)$ teilbar durch eine ausreichend große Primzahl n ist (zum Beispiel $n > 2^{160}$)[Sma99].
- Größtmöglicher Widerstand zu den oben genannten Attacken wird erreicht durch die Wahl von E , sodass $\#E(\mathbb{F}_q)$ prim oder nahezu prim ist, beispielsweise $\#E(\mathbb{F}_q) = hn$, wobei n prim und h klein ist (zum Beispiel $h = 1, 2, 3$)[Sma99].
- Um Attacken auf *vom Primkörper abweichenden Kurven* zu verhindern, sollte verifiziert werden, dass $\#E(\mathbb{F}_q) \neq q$ gilt [SA98].
- Die elliptische Kurve sollte der MOV Bedingung genügen, um eine Weil und Tate Paarungsattacke zu vermeiden[MOV91],[IEE].
- Zur Sicherung des Widerstandes gegen die Weil Senkungsattacke sollte nur ein binärer Körper \mathbb{F}_2^m genutzt werden mit primem m [FR94].
- Zum Schutz gegenüber zukünftigen Attacken für spezielle Klassen von nicht-supersingulären Kurven, ist es notwendig eine zufällige elliptische Kurve zu wählen [X9.99].

3.4 Randomisierung

Die Randomisierung kann als Gegenmaßnahme gegen Seitenkanalangriffe angewendet werden. Im speziellen für Elliptische-Kurven-Kryptosysteme, ist die Randomisierung zur Änderung der Punktrepräsentation ein einfaches und effektives Hilfsmittel. Andere Gegenmaßnahmen, die die Berechnung unabhängig von der Punktrepräsentation randomisieren, zum Beispiel die Randomisierung des Skalars, sind auch möglich. Im Folgenden werden wir einige Konzepte der Randomisierung vorstellen [IIT03]:

- *Randomisierte Projektive Koordinaten (RPC):*
Sei $P = (x, y, z)$ ein Basispunkt in projektiven Koordinaten. Dann gilt $(x, y, z) = (rx, ry, rz)$ für alle Elemente r im Körper, aber es entspricht verschiedenen Daten als Bitfolge [Cor99].
- *Randomisierte Kurve (RC):*
Diese Randomisierung wird von Joye und Tymen in [JT01] beschrieben, welches eine Isomorphe Kurve so wählt, dass der Punkt (x, y, z) dem Punkt (r^2x, r^3y, rz) einer anderen, isomorphen Kurve entspricht. Die elliptischen Kurven sind mathematisch dieselben, exemplarisch sind sie Isomorph, aber unterscheiden sich in der jeweiligen Bitfolge.
- *Randomisierter Exponent:*
Der Term dP kann mit $(d + rn)P$ ausgedrückt werden, wobei n die Ordnung des Basispunktes P und r eine Zufallsszahl ist [Cor99].

- *Randomisierter Basispunkt:*
Der Term dP kann mit $d(P + R) - dR$ ausgedrückt werden, wobei R ein zufälliger Punkt ist [Cor99].
- *Randomisierter Startpunkt:*
Messerges et al. stellen in [MDS99] ein Splitting des Skalars durch die Wahl eines Startbits d_i vor. Die Multiplikation von d mit P wird berechnet mit dem gewählten Bit über MSB für die oberen Bits und über LSB für die niedrigeren Bits, zum Beispiel mit dem “left-to-right”-Algorithmus.
- *Exponentensplitting:*
Dieses Verfahren unterteilt den Skalar in r und in $d - r$, wobei r eine Zufallszahl ist und die Tatsache, dass dP durch $(d - r)P + rP$ ausgedrückt werden kann, gilt [CJ01].
- *Randomisierte Adressierung:*
Dieses Verfahren wurde in [IIT03] vorgestellt und randomisiert die Registeradressen durch eine Zufallszahl r wie folgt: Alle Parameter d_i werden zu $d_i \oplus r_i$, wobei \oplus für die XOR Operation steht.

Kapitel 4

Effizienz

In diesem Kapitel werden wir auf die effiziente Skalarmultiplikation eingehen. Es sei angemerkt, dass die in der Hauptberechnung verwendete Punktaddition mit “ $R + P$ ” der bekannten ECADD-Funktion entspricht und die in der Hauptberechnung verwendete Punktmultiplikation “ $2 \cdot R$ ” die gewohnte ECDBL-Funktion erfüllt.

4.1 Einfach Punktmultiplikation

Sei P ein Punkt auf der Kurve E über dem Körper \mathbb{F}_p und k der n -Bit lange Exponent mit der Binärdarstellung $k = \sum_{i=0, \dots, n-1} k_i 2^i$ mit $k_i \in \{0, 1\}$.

4.1.1 Square and Multiply

Diese Methode [MvOV97] startet mit dem Punkt im Unendlichen \mathcal{O} und scannt k bitweise von links nach rechts. Für jeden neuen Index i verdoppelt sie den bisherigen Ergebnispunkt und für jedes $k_i \neq 0$ addiert sie P dazu. Der Algorithmus 4.1.1 sieht wie folgt aus.

ALG 4.1.1: Square and Multiply:

```
Input:  $k, P$ 
Output:  $k \cdot P$ 
 $R \leftarrow \mathcal{O}$ 
for  $i = n - 1$  down to 0
     $R \leftarrow 2 \cdot R$ 
    if  $k_i = 1$  then
         $R \leftarrow R + P$ 
return  $R$ 
```

Dieser Algorithmus benötigt immer $n-1$ Punktverdoppelungen und $w(k)$ Punktadditionen. $w(k)$ bezeichnet die Anzahl der $k_i = 1$ und wird auch die Gewichtung bzw. das Hamming-Gewicht von k genannt.

4.1.2 Fixed-size-window Exponentiation

Die Fixed-size sliding window Exponentiation Methode [MvOV97] betrachtet jeweils w Bits des Exponenten k auf einmal. w wird Fenstergröße genannt. Diese w Bits werden auf einmal abgearbeitet, danach wird das Fenster um seine Breite weiter geschoben. Der Algorithmus 4.1.2 zeigt diese Methode.

ALG 4.1.2:Fixed-size sliding window Exponentiation:

Vorbereitung:

Input: w, P

Output: $P_i = i \cdot P$ mit $0 \leq i < 2^w$

$P_0 \leftarrow P$

for $i = 1$ to $2^{w-1} - 1$

$j = 2 \cdot i$

$P_{j-1} \leftarrow 2 \cdot P_{i-1}$

$P_j \leftarrow P_{i-1} + P_i$

Hauptberechnung:

Input: $k, P, P_i = i \cdot P$ mit $0 \leq i < 2^w$

Output: $k \cdot P$

$R \leftarrow \mathcal{O}, i \leftarrow n - 1$

$t = (i+1) \% w$

if ($t \neq 0$)

 if $(k_{n-1}k_{n-2} \dots k_{t+1})_2 \neq 0$ then

$R \leftarrow R + P_{(k_{n-1}k_{n-2} \dots k_{t+1})_2}$

$i \leftarrow i - t$

while $i \geq 0$

 for $j = 0$ to $j \leq w - 1$

$R \leftarrow 2 \cdot R$

 if $(k_i k_{i-1} \dots k_{i-w+2} k_{i-w+1})_2 \neq 0$ then

$R \leftarrow R + P_{(k_i k_{i-1} \dots k_{i-w+2} k_{i-w+1})_2}$

$i \leftarrow i - w$

return R

Die Komplexität ist von der Fenstergröße w abhängig. Ein optimales w für die kleinstmögliche Komplexität, ergibt sich aus dem Verhältnis der Laufzeit der Addition zu der Laufzeit der Verdoppelung.

4.1.3 Sliding Window Exponentiation

Diese Methode [MvOV97] scannt k nicht bitweise, sondern immer w Bits auf einmal. w bezeichnet man als Fenster. Diese Methode berechnet zuerst 2^{w-1} Punkte vor, die in der Hauptberechnung wieder verwendet werden. Dies ist in Algorithmus 4.1.3 zu sehen.

ALG 4.1.3: Sliding Window Exponentiation:

Vorbereitung:

Input: w, P Output: $P_i = i \cdot P$ mit $0 \leq i < 2^w$ $P_1 \leftarrow P, t \leftarrow 2 \cdot P$ for $i = 1$ to $2^{w-1} - 1$ $P_i \leftarrow P_{i-1} + t$

Hauptberechnung:

Input: k, P, P_i Output: $k \cdot P$ $R \leftarrow \mathcal{O}, i \leftarrow n - 1$ while $i \geq 0$ if $k_i = 0$ then $R \leftarrow 2 \cdot R, i \leftarrow i - 1$

else

 $m \leftarrow i - w$, if $(w < -1)$ then $m \leftarrow -1$ $j \leftarrow m + 1$ while $(k_j = 0)$ $j++$ $s \leftarrow (k_i k_{i-1} \dots k_j)_2$ while $(i \geq j)$ $R \leftarrow 2 \cdot R, i \leftarrow i - 1$ $R \leftarrow R + P_{(s-1)/2}$ while $(i > m)$ $R \leftarrow 2 \cdot R, i \leftarrow i - 1$ return R **4.1.4 Exponentiation mit Non-adjacent forms**

In der additiven Punktgruppe ist die Invertierung eines Punktes $P = (x, y)$ sehr einfach. $-P = (x, -y)$. Dadurch kann die Punktmultiplikation durch eine Vorzeichen-behaftete-Darstellung des Exponenten k noch effizienter gemacht werden. Folgende Technik wurde in [MOC97] vorgestellt:

Sei k der Exponent mit der Bitlänge n und w eine Fenstergröße. Eine *width-(w+1) non-adjacent-form* (w -NAF) von k ist ein Feld $N[b], \dots, N[0]$ ganzer Zahlen, so dass:

- Jeder Eintrag $N[j]$ ist entweder 0 oder ungerade und $-2^w < N[j] < 2^w$
- $k = \sum_{0 \leq j \leq b} N[j] \cdot 2^j$
- höchstens einer der $w + 1$ aufeinanderfolgenden Einträge des Feldes N ist Nicht-Null.

w -NAFs existieren immer und können durch folgenden Algorithmus 4.1.4a) berechnet werden [Sol00]:

```

ALG 4.1.4a):

c ← k
j ← 0
while c > 0 do
  if c[0] > 0 then
    u ← c[w ... 0]
    if u[w] = 1 then
      u ← u - 2w+1
    c ← c - u
  else
    u ← 0
  N[j] ← u, j ← j + 1
  c ← c/2
while j ≤ b
  N[j] ← 0; j ← j + 1
return N[b] ... N[0]

```

Für eine n -Bit lange Zahl k ist N höchstens $n + 1$ Felder lang. Die durchschnittliche Gewichtung $w(N)$ beträgt $n/(w + 2)$ für $n \rightarrow \infty$ [Sol00]. Siehe dazu den Algorithmus 4.1.4b) .

ALG 4.1.4b): Exponentiation mit Non-adjacent-forms:
 Vorberechnung:
 Input: w, P
 Output: $P_i = (2i + 1) \cdot P$ mit $0 \leq i < 2^w$
 $t = 2 \cdot P$
 $P_0 = P$
 for $i = 1$ to $2^w - 1$
 $P_i \leftarrow P_{i-1} + t$

Hauptberechnung:
 Input: k, P, P_i
 Output: $k \cdot P$
 $R \leftarrow \mathcal{O}$
 for $i = N.\text{length}; i \geq 0; i \leftarrow i - 1$
 $R \leftarrow 2 \cdot R$
 if $N[i] \neq 0$ then
 $R \leftarrow R + N[i] \cdot P$
 return R

4.1.5 Spezialfall: ± 1 -Ketten

Die Punktmultiplikation mit ± 1 -Ketten ist der Spezialfall der Exponentiation mit Non-adjacent-forms [Hen03], nämlich für $w = 1$. Die 1-NAF eines jeden Exponenten k besteht nur aus -1, 0 und 1.

Sei n die maximale mögliche Bitlänge des Exponenten k , dann werden $n - 1$ Punkte vorberechnet, alle 2er-Potenzen von P . Die Punktmultiplikation kann dann alleine mit Punktadditionen berechnet werden, Punktverdoppelungen sind nicht mehr nötig. Siehe dazu den Algorithmus 4.1.5. .

ALG 4.1.5: Exponentiation mit ± 1 -Ketten:
 Vorberechnung:
 Input: P
 Output: $P_i = 2^i \cdot P$ mit $0 \leq i < n$
 $P_0 = P$
 for $i = 1$ to $n - 1$
 $P_i \leftarrow 2 \cdot P_{i-1}$

Hauptberechnung:
 Input: $k, P, P_i, N(k)$
 Output: $k \cdot P$
 $R \leftarrow \mathcal{O}$
 for $i = N.length; i \geq 0; i \leftarrow i - 1$
 if $N[i] == -1$ then
 $R \leftarrow R - P_i$
 else if $N[i] == 1$ then
 $R \leftarrow R + P_i$
 return R

4.1.6 LimLee Exponentiation

Diese Methode [LL94] hat einen großen Aufwand in der Vorberechnung, ist jedoch äußerst effizient in der Hauptberechnung.

Sei k der n -Bit lange Exponent. Teile k in h gleich große Teile der Länge v ($k_i = (k_{ia+h-1}k_{ia+h-2} \dots k_{ia+1}k_{ia+0})$, mit $i = 0, \dots, h - 1$). Diese h Teile der Länge a wiederum teile in v gleich große Teile der Länge b .

Beispiel: $n = 24$, $h = 4$, $v = 3$.

$$k = \left| \begin{array}{cc|cc|cc} k_{0,2,1} & k_{0,2,0} & k_{0,1,1} & k_{0,1,0} & k_{0,0,1} & k_{0,0,0} \\ k_{1,2,1} & k_{1,2,0} & k_{1,1,1} & k_{1,1,0} & k_{1,0,1} & k_{1,0,0} \\ k_{2,2,1} & k_{2,2,0} & k_{2,1,1} & k_{2,1,0} & k_{2,0,1} & k_{2,0,0} \\ k_{3,2,1} & k_{3,2,0} & k_{3,1,1} & k_{3,1,0} & k_{3,0,1} & k_{3,0,0} \end{array} \right|$$

Dabei läuft $k_{i,j,l}$ i von 0 bis $h - 1$, j von 0 bis $v - 1$ und l von 0 bis $b - 1$. Sei $\mu \in 0, \dots, 2^h - 1$ in Binärdarstellung und $K_\mu = \sum_{i=0, \dots, h-1} \mu_i 2^{ai}$. Berechnet und gespeichert werden alle $P_K = K_\mu \cdot P$ und weiterhin alle $P_{K_\mu, j} = 2^{jb} \cdot P_{K_\mu}$ für $j = 0, \dots, v - 1$. Das sind $v(2^h - 1)$ Einträge.

$k \cdot P$ lässt sich mit folgendem Algorithmus 4.1.6 berechnen:

```

ALG 4.1.6:LimLee Exponentiation:

Vorbereitung:
Input:  $P$ 
Output:  $P_{i,j} = 2^{i \cdot b} \cdot jP$  mit  $0 \leq i < v$  und  $0 \leq j < h$ 
 $D_{0,0} = P$ 
for  $i = 0$  to  $h - 1$ 
  for  $j = 0$  to  $v - 1$ 
    if  $i = 0$  then  $j = 1$ 
    if  $j = 0$  then
       $D_{i,j} = D_{i-1,v-1}$ 
    else
       $D_{i,j} = D_{i,j-1}$ 
  for  $k = 0$  to  $b - 1$ 
     $D_{i,j} = 2 \cdot D_{i,j}$ 
  for  $k = 0$  to  $v - 1$ 
    for  $i = 1$  to  $2^h - 1$ 
       $P_{k,i} = \mathcal{O}$ 
      for  $j = 0, s = 0$  to  $s < i, j = j + 1$ 
        if  $i \& j \neq 0$  then
           $s = s + (i \& j)$ 
       $P_{k,i} = P_{k,i} + D_{k,j-1}$ 
return  $P_{-, -}$ 

Hauptberechnung:
 $R \leftarrow \mathcal{O}$ 
for  $i = b - 1$  to  $0$ 
   $R \leftarrow 2 \cdot R$ 
  for  $j = v - 1$  to  $0$ 
    return  $R \leftarrow R + P_{K_\mu, j}$ 
return  $R$ 

\\ \mu ist hier die Binärdarstellung von  $k_{h-1,j,0} \dots k_{0,j,0}$ 

```

4.1.7 Komplexitätsvergleich und Analyse

Wir haben nun fünf verschiedene Methoden zur einfachen Punktmultiplikation der Form $k \cdot P$ gesehen. Vier von ihnen profitieren von einer Vorberechnungsphase: Eine Menge von Vielfachen des Punktes P wird einmal berechnet und in der Hauptberechnungsphase mehrfach wieder verwendet. Diese Vorberechnungsphase kann, wenn P im Voraus bekannt ist, einmal berechnet und gespeichert werden. Wenn diese Operation mit dem gleichen Punkt P wiederholt wird, können diese Punkte in der Methode wieder verwendet werden. Ein Anwendungsgebiet ist die Signaturerzeugung: P ist in diesem Fall der Basispunkt G der EC Parameter, die man für viele Signaturen wieder verwendet.

Nachfolgende Tabelle 4.1 zeigt die Komplexitäten [Hen03].

Tabelle 4.1: Komplexitätsvergleich I

Methode	Vorberchnung		Hauptberechnung	
	# ADD	# DBL	# ADD	# DBL
left-to-right	-	-	$\approx \frac{n}{2}$	$n - 1$
Fixed Sliding-Window	2^{w-1}	2^{w-1}	$\frac{n(2^w-1)}{w \cdot 2^w}$	$n - (n \% w)$ oder $n - 1$
Sliding-Window	2^{w-1}	1	$\frac{n}{w+1}$	$n - w$ bis $n - 1$
w -NAF	2^{w-1}	1	$\frac{n}{w+2}$	$n - w$ bis $n - 1$
LimLee	$v(2^{h-1} - 1)$	$n - \frac{n}{hv}$	$\frac{n}{h}$	$\frac{n}{hv} - 1$

4.2 Mehrfache Punktmultipliktion

In diesem Abschnitt beschreiben wir die Methoden zur Punktmultiplikation, die sich besonders für Signaturverfahren eignen.

Die teuerste Operation der Verifikation ist $h_1 \cdot G + h_2 \cdot W$, also eine zwei Punktmultiplikationen, deren Ergebnis noch miteinander addiert wird. Die Idee ist, diese drei Einzeloperationen zu einer einzigen zu verknüpfen und dabei Punktverdoppelungen einzusparen. Dafür gibt es zwei Ansätze: Die einzelnen benötigten Punktadditionen und -verdoppelungen werden simultan ausgeführt. Damit lässt sich etwa die Hälfte der Verdoppelungen einsparen. Dieser Ansatz wird in den Algorithmen 4.2.1, 4.2.2 und 4.2.3 verwendet. Der zweite führt die Punktverdoppelungen auch simultan aus, die Addition aber nur abwechselnd. Dadurch ist im Schnitt die Anzahl der benötigten Punktadditionen kleiner. Dieser Ansatz wird in 4.2.4 und 4.2.5 verwendet.

Jeder dieser Algorithmen ist nicht auf zwei Exponenten mit zwei Basen beschränkt. Sie können beliebig erweitert werden. Wir betrachten hier nur den Fall $h_1 \cdot G + h_2 \cdot W$. Im Folgenden seien $i = 1, 2$, k_i die ganzzahligen Exponenten und P_i die Basen. R sei der Ergebnispunkt.

Die Algorithmen beinhalten eine Vorberechnung.

4.2.1 Simultane Exponentiation

Für die Operation $R = k_1 \cdot P_1 + k_2 \cdot P_2$ besteht die Vorberechnung lediglich aus einer Punktaddition $A = P_1 + P_2$. Das Grundprinzip leitet sich aus Algorithmus 4.1.1 ab: Von links nach rechts wird der Zwischenergebnispunkt R verdoppelt und für jedes Paar $(k_1[j], k_2[j]) \neq (0, 0)$, $j = n - 1, \dots, 0$, P_1, P_2 oder A dazuaddiert [MvOV97]. Algorithmus 4.2.1 zeigt dies.

ALG 4.2.1: Simultane Exponentiation:

Vorberechnung:

$$A = P_1 + P_2$$

Hauptberechnung:

$$R \leftarrow \mathcal{O}$$

for $i = n - 1$ down to 0

$$R \leftarrow 2 \cdot R$$

if $k_1[i] = 1$ then

if $k_2[i] = 1$ then

$$R \leftarrow R + A$$

else

$$R \leftarrow R + P_1$$

else

if $k_2[i] = 1$ then

$$R \leftarrow R + P_2$$

return R

4.2.2 Simultane 2^w -ary Exponentiation

Die Simultane 2^w -adische Exponentiation [Moe01][Str64] betrachtet w Bits jedes Exponenten gleichzeitig. Sie ist damit die Generalisierung des Algorithmus 4.1.2 und wird anhand des Algorithmus 4.2.2 beschrieben.

ALG 4.2.2: Simultane 2^w -ary Exponentiation:

Vorbereitung:

Input: w, P

Output: $P_{i,j} = i \cdot P_1 + j \cdot P_2$ mit $0 \leq i, j < 2^w$

$P_{0,0} \leftarrow \mathcal{O}$

$P_{1,1} \leftarrow P_1 + P_2$

for $i = 1$ to $2^w - 1$

$j = 2i$

$P_{0,j} \leftarrow 2 \cdot P_{0,i}, P_{0,j+1} \leftarrow P_{0,i} + P_{0,i+1}$

$P_{j,0} \leftarrow 2 \cdot P_{i,0}, P_{j+1,0} \leftarrow P_{i,0} + P_{i+1,0}$

$P_{j,j} \leftarrow 2 \cdot P_{i,i}, P_{j+1,j+1} \leftarrow P_{i,i} + P_{i+1,i+1}$

for $i = 1$ to $2^w - 1$

 for $j = 1$ to $2^w - 1$

 if $(i = j)$ then $j \leftarrow j + 1$

$P_{i,j} \leftarrow P_{i,0} + P_{0,j}$

Hauptberechnung:

Input: $k_1, P_1, k_2, P_2, w, P_{i,j} = i \cdot P_1 + j \cdot P_2$ mit $0 \leq i, j < 2^w$

Output: $k_1 \cdot P_1 + k_2 \cdot P_2$

$R \leftarrow \mathcal{O}$

for $j = \lfloor (n-1)/w \rfloor w$ down to 0

 for $i = 1$ to w

$R \leftarrow 2 \cdot R$

 if $(k_1[j+w-1 \dots j]), (k_2[j+w-1 \dots j]) \neq (0,0)$ then

$R \leftarrow R + P_{(k_1[j+w-1 \dots j]), (k_2[j+w-1 \dots j])}$

return R

4.2.3 Simultane Sliding-Window Exponentiation

Diese Methode [YLL94] [Moe01] ist eine Generalisierung der einfachen Sliding Window Methode und wird deutlich anhand des Algorithmus 4.2.3.

ALG 4.2.3: Simultane sliding-window Exponentiation:

Vorberechnung:

Input: w, P

Output: $P_{i,j} = i \cdot P_1 + j \cdot P_2$ mit $0 \leq i, j < 2^w$,
wobei i oder j ungerade ist.

$P_{12} \leftarrow 2 \cdot P_1, P_{22} \leftarrow 2 \cdot P_2$

$P_{1,0} \leftarrow P_1, P_{0,1} \leftarrow P_2$

for $i = 3$ to $2^w - 1$ step 2

$P_{0,i} \leftarrow P_{0,i-2} + P_{22}, P_{i,0} \leftarrow P_{i-2,0} + P_{12}$

for $i = 1$ to $2^w - 1$ step 2

for $j = 1$ to $2^w - 1$

$P_{i,j} \leftarrow P_{i,j-1} + P_2$

for $i = 2$ to $2^w - 1$ step 2

for $j = 1$ to $2^w - 1$ step 2

$P_{i,j} \leftarrow P_{i-1,j} + P_1$

Hauptberechnung:

Input: $w, P_1, P_2, k_1, k_2, P_{i,j} = i \cdot P_1 + j \cdot P_2$ mit $0 \leq i, j < 2^w$,
wo i oder j ungerade ist.

Output: $k_1 \cdot P_1 + k_2 \cdot P_2$

$R \leftarrow \mathcal{O}, j \leftarrow n - 1$

while $j \geq 0$

if $k_1[j] = 0$ and $k_2[j] = 0$

$R \leftarrow 2 \cdot R, j \leftarrow j - 1$

else

$j_{new} \leftarrow \max(j - w, -1), h \leftarrow j_{new} + 1$

while $k_1[h] = 0$ and $k_2[h] = 0$

$h \leftarrow h + 1$ (d.h. $j \geq h > j_{new}$)

$K_1 \leftarrow k_1[j \dots h], K_2 \leftarrow k_2[j \dots h]$

while $j \geq h$

$R \leftarrow 2 \cdot R, j \leftarrow j - 1$

$R \leftarrow R + A_{K,L}$

while $j > j_{new}$

$R \leftarrow 2 \cdot R, j \leftarrow j - 1$

return R

4.2.4 Basic Interleaving Exponentiation

Der Vorteil der Basic Interleaving Exponentiation [Moe01] und der nachfolgenden w -NAF Interleaving Exponentiation [Moe01] liegt darin, dass weniger Punkte vorberechnet werden und die Methoden beschleunigt werden können, falls einer der Punkte P_1 oder P_2 im Voraus bekannt ist. Dies kann der Fall bei der Signaturverifikation sein. Dabei ist P_1 der Basispunkt G . Vielfache dieses Punktes können gemäß den folgenden Methoden vorberechnet und bei Bedarf wieder verwendet werden.

Folgende Methode ist eine Generalisierung der Sliding Window Methode mit einem Exponenten. Siehe hierzu den Algorithmus 4.2.4.

ALG 4.2.4: Basic Interleaving Exponentiation:

Vorbereitung:

Input: w_1, w_2, P_1, P_2

Output: $P_{1,j} = j \cdot P_1$ und $P_{2,j} = j \cdot P_2$ mit $0 \leq j < 2_{1,2}^w$,
 j ungerade

$P_{12} \leftarrow 2 \cdot P_1, P_{22} \leftarrow 2 \cdot P_2$

for $i = 3$ to 2^{w_1} step 2

$P_{1,i} \leftarrow P_{1,i-1} + P_{12}$

for $i = 3$ to 2^{w_2} step 2

$P_{2,i} \leftarrow P_{2,i-1} + P_{22}$

Hauptberechnung:

Input: $w_1, w_1, P_1, P_2, P_{1,j} = j \cdot P_1$ und

$P_{2,j} = j \cdot P_2$ mit $0 \leq j < 2_{1,2}^w, j$ ungerade

Output: $k_1 \cdot P_1 + k_2 \cdot P_2$

$R \leftarrow \mathcal{O}$

for $i = 1$ to 2

$window_handle_i \leftarrow \text{null}$

for $j = n - 1$ down to 0

$R \leftarrow 2 \cdot R$

for $i = 1$ to 2

if $window_handle_i = \text{null}$ and $k_i[j] = 1$

$J \leftarrow j - w_i + 1$

while $k_i[J] = 0$

$J \leftarrow J + 1 \rightarrow j \geq J > j - w$ und $J \geq 0$

$window_handle_i \leftarrow J$

$K_i \leftarrow K_i[j \dots J]$

if $window_handle_i = j$

$R \leftarrow R + (E_i \cdot P_i)$ (mittels Tabelleneintrag)

$window_handle_i \leftarrow \text{null}$

return R

4.2.5 w -NAF-based Interleaving Exponentiation

Die w -NAF-based interleaving Exponentiation [Moe01] ist eine Generalisierung der einfachen Exponentiation mit NAF, ALG 4.1.4.

Sie profitiert auch davon, dass die Negation von Punkten nahe zu umsonst ist. So werden im Vergleich zur letzten Methode weitere Punktadditionen eingespart, wie wir auch in Algorithmus 4.2.5 sehen können .

ALG 4.2.5: w -NAF-based Interleaving Exponentiation:

Vorberechnung:

Input: w_1, w_2, P_1, P_2

Output: $P_{1,j} = j \cdot P_1$ und $P_{2,j} = j \cdot P_2$ mit $0 \leq j < 2_{1,2}^w$,
 j ungerade

$P_{12} \leftarrow 2 \cdot P_1, P_{22} \leftarrow 2 \cdot P_2$

for $i = 3$ to 2^{w_1} step 2

$P_{1,i} \leftarrow P_{1,i-1} + P_{12}$

for $i = 3$ to 2^{w_2} step 2

$P_{2,i} \leftarrow P_{2,i-1} + P_{22}$

Hauptberechnung:

Input: $w_1, w_1, P_1, P_2, P_{1,j} = j \cdot P_1$ und

$P_{2,j} = j \cdot P_2$ mit $0 \leq j < 2_{1,2}^w, j$ ungerade

Output: $k_1 \cdot P_1 + k_2 \cdot P_2$

$R \leftarrow \mathcal{O}$

for $i = 1$ to 2

$N_i[n], \dots, N_i[n_i + 1] \leftarrow 0, \dots, 0$

$N_i[n_i], \dots, N_i[0] \leftarrow w$ -NAF von k_i

for $j = n$ down to 0

$R \leftarrow 2 \cdot R$

for $i = 1$ to 2

if $N_i[j] = 0$

$R \leftarrow R + N_i[j] \cdot P_i$

return R

4.2.6 Komplexitätsvergleich und Analyse

Nachfolgende Tabelle 4.2 [Hen03] zeigt die Komplexitäten der Algorithmen zur mehrfachen Punktmultiplikation der Form $k_1 \cdot P_1 + k_2 \cdot P_2$. Die Anzahl der Punktadditionen ($\#$ ADD) ist als Erwartungswert zu lesen und ist nicht absolut.

Tabelle 4.2: Komplexitätsvergleich II

Methode	Vorberchnung		Hauptberechnung	
	$\#$ ADD	$\#$ DBL	$\#$ ADD	$\#$ DBL
Sim. Exp	1	-	$\frac{3}{4} \cdot n$	$n - 1$
Sim. 2^w -ary	$2^{2w} - 2^{2w-2} - 2$	$2^{w-2} - 1$	$\frac{1 - \frac{1}{2^{2w}}}{w} \cdot n$	$n - (n \% w)$ oder $n - 1$
Sim. slid. win.	$2^{2w} - 2^{2w-2} - 2$	2	$\frac{n}{w + \frac{1}{3}}$	$n - (n \% w)$ oder $n - 1$
Basic Inter.	$2^{w_1-1} + 2^{w_2-1} - 2$	2	$\frac{n \cdot (w_1 + w_2 + 2)}{(w_1 + 1) \cdot (w_2 + 1)}$	$n - \max_i w_i$ oder $n - 1$
w -NAF Inter.	$2^{w_1-1} + 2^{w_2-1} - 2$	2	$\frac{n \cdot (w_1 + w_2 + 4)}{(w_1 + 2) \cdot (w_2 + 2)}$	$n - \max_i w_i$ oder $n - 1$

Kapitel 5

Angriffe

5.1 Die Exponentenrekodierung als wertvolle Informationsquelle

Bei genauerer Betrachtung der effizienten Exponentiation, sehen wir, dass die Vorberechnungsphase bzw. Umrechnung des Exponenten eine verwertbare Informationsquelle für den Angreifer darstellen kann.

Im Folgenden werden wir Angriffe auf die Vorberechnungsphase der w -NAF [Sol00] und der vorzeichenlosen/vorzeichenbehafteten *fractional-Window* Repräsentation [Moe02] sehen.

5.1.1 NAF

Die vorzeichenbehaftete Binärdarstellung, auch *non-adjacent-form* genannt (NAF) [Rei60], ist eine vorzeichenbehaftete Darstellung eines Integers. Die durchschnittliche *non-zero-density* bzw. Dichte der Einser liegt hier bei $1/3$ für den Exponenten $d \rightarrow \infty$. Die Algorithmen 5.1.1a), 5.1.1b) zeigen die Exponentenrekodierung mittels dem einfachen NAF Algorithmus.

ALG 5.1.1a): Exponentenrekodierung für NAF:

Eingabe: ein nicht-negativer t -bit großer Integer $d = (d_{t-1} \dots d_0)_2$

Ausgabe: $b = (b_t \dots b_0)$, mit $b_i \in \{-1, 0, 1\}$ und $b_i b_{i+1} = 0$ für alle i

1. $c_0 \leftarrow 0, d_{t+1} \leftarrow 0, d_t \leftarrow 0$
2. for $i = 0$ to t do
3. $c_{i+1} \leftarrow \lfloor (c_i + d_i + d_{i+1})/2 \rfloor$
4. $b_i \leftarrow c_i + d_i - 2 \cdot c_{i+1}$
5. end for
6. return $b = (b_t \dots b_0)$

ALG 5.1.1b): Variante der Exponentenrekodierung für NAF:

Eingabe: ein nicht-negativer t -bit großer Integer $d = (d_{t-1} \dots d_0)_2$

Ausgabe: $b = (b_t \dots b_0)$, mit $b_i \in \{-1, 0, 1\}$ und $b_i b_{i+1} = 0$ für alle i

1. $b = (s_t \dots s_0 d_0)_2 \leftarrow 3d$
2. $b \leftarrow b - 2d$, ($b_i = s_i - d_{i+1}$ mit $0 - 1 = \underline{1}$), mit $\underline{1} = -1$
3. return $b = (b_t \dots b_0)$

5.1.2 w -NAF

w -NAF [Sol00] ist eine Generalisierung von NAF. Das gewöhnliche NAF arbeitet mit $w = 2$. Algorithmus 5.1.2 zeigt die Exponentenrekodierung für w -NAF. Die durchschnittliche *non-zero density* bzw. Dichte liegt hier bei $1/(w + 1)$ für den Exponenten $d \leftarrow \infty$.

ALG 5.1.2: Exponentenrekodierung für w -NAF:

Eingabe: ein nicht-negativer t -bit großer Integer $w \geq 2$

Ausgabe: $b = (b_t \dots b_0)$, mit $b_i \in \{0, \pm 1, \pm 3, \dots, \pm(2^{w-1} - 1)\}$
und bei beliebigen w mit aufeinanderfolgenden $b_j S$,
ist mindestens eines Nicht-Null.

1. $c \leftarrow d, i \leftarrow 0$
2. while $c > 0$ do
3. if c **ungerade** then
4. $b_i \leftarrow c \bmod 2^w$
5. if $b_i \geq 2^{w-1}$ then
6. $b_i \leftarrow b_i - 2^w$
7. end if
8. $c \leftarrow c - b_i$
9. else
10. $b_i \leftarrow 0$
11. end if
12. $c \leftarrow \lfloor c/2 \rfloor, i \leftarrow i + 1$
13. end while
14. Return $b = (b_t \dots b_0)$

5.1.3 Fractional Window

In kleinen Geräteeinheiten ist die Wahl des w für die Exponentiation mit w -NAF durch die Speicherbegrenzung vorbestimmt. Möller [Moe02] stellt eine *fractional window* Methode vor, welche als Generalisierung der *sliding window* und des w -NAF Ansatzes betrachtet werden kann. Die *fractional window* Methode ist flexibler in der Tabellengröße.

Die *fractional window* Methode hat zwei Varianten. Die Eine ist eine vorzeichenbehaftete Version, welche negative Bits erlaubt. Siehe dazu ALG 5.1.3a) . Die Andere ist eine vorzeichenlose Variante für den Fall, dass nur nicht-negative Bits zulässig sind. Siehe dazu ALG 5.1.3b).

Sei $w \geq 2$ ein Integer und m ein ungerader Integer mit $1 \leq m \leq 2^w - 3$. Die durchschnittliche Dichte der vorzeichenbehafteten *fractional window* Methode liegt hier bei

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

für $d \leftarrow \infty$.

Die durchschnittliche Dichte der vorzeichenlosen *fractional window* Darstellung liegt bei

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

für $d \leftarrow \infty$ [Moe02].

ALG 5.1.3a): Exponentrekodierung der vorzeichenlosen Darstellung:

Eingabe: ein nicht-negativer Integer d , ein Integer $w \geq 2$,
ein ungerader Integer m , $1 \leq m \leq 2^w - 3$
Ausgabe: $b = (b_{i-1} \dots b_0)$, mit $b_i \in \{0, 1, 3, \dots, 2^w + m\}$

1. $c \leftarrow d, i \leftarrow 0$
2. while $c > 0$ do
3. if c **ungerade** then
4. $b_i \leftarrow c \bmod 2^{w+1}$
5. if $2^w + m < b_i - 2^w$
6. $b_i \leftarrow b_i - 2^w$
7. end if
8. $c \leftarrow c - b_i$
9. else
10. $b_i \leftarrow 0$
11. end if
12. $c \leftarrow \lfloor c/2 \rfloor, i \leftarrow i + 1$
13. end while
14. Return $b = (b_{i-1} \dots b_0)$

ALG 5.1.3b): Exponentrekodierung der vorzeichenbehaftete Darstellung:

Eingabe: ein nicht-negativer Integer d , ein Integer $w \geq 2$,
ein ungerader Integer m , $1 \leq m \leq 2^w - 3$
Ausgabe: $b = (b_{i-1} \dots b_0)$, mit $b_i \in \{0, \pm 1, \pm 3, \dots, \pm(2^w + m)\}$

1. $c \leftarrow d, i \leftarrow 0$
2. while $c > 0$ do
3. if c **ungerade** then
4. $b_i \leftarrow c \bmod 2^{w+2}$
5. if $2^w + m < b_i < 3 \cdot 2^w - m$ then
6. $b_i \leftarrow b_i - 2^{w+1}$
7. else if $3 \cdot 2^w - m \leq b_i < 2^{w+2}$ then
8. $b_i \leftarrow b_i - 2^{w+2}$
9. end if
10. $c \leftarrow c - b_i$
11. else
12. $b_i \leftarrow 0$
13. end if
14. $c \leftarrow \lfloor c/2 \rfloor, i \leftarrow i + 1$
15. end while
16. Return $b = (b_{i-1} \dots b_0)$

5.1.4 Das Informationsleck: Exponentenrekodierung

Die Algorithmen 5.1.2, 5.1.3a) und 5.1.3b) haben Verzweigungsbedingungen. Obwohl die Identifizierung von Verzweigungsbedingungen durch eine direkte Beobachtung des Stromverbrauchs einer Geräteeinheit sich als schwierig erweisen kann, sind die statistischen Operationen, welche bei DPAs verwendet werden, in der Lage zuverlässig aussergewöhnlich kleine Unterschiede im Stromverbrauch zu erkennen [KuBJ]. Im Weiteren werden wir den Informationsverlust der kryptographischen Geräteeinheiten während der Exponentrekodierung behandeln.

5.1.5 Der Angreifer

- Der Angreifer hat Zugang zu den Geräteeinheiten, welche die Exponentrekodierung durchführen.
- Der Angreifer hat Kenntnisse über die Implementierung. Er kennt den Algorithmus einschließlich der Parameter w und m .
- Der Angreifer kann die Verzweigungsbedingungen in Algorithmus 5.1.2, 5.1.3a) und 5.1.3b) durch Überwachung des Stromverbrauches während der Berechnungsphase der Exponentrekodierung klar erkennen.

Ziel: Wiederherstellung des geheimen Exponenten.

5.1.6 Das Leck

Die w -NAF Rekodierung (siehe Alg 5.1.2) hat zwei Verzweigungsbedingungen, Schritt 3 und 4, in der Hauptschleife. Falls c ungerade ist, werden die Schritte 4 bis 8 ausgeführt. Falls c gerade ist wird der Schritt 10 ausgeführt. Falls c ungerade ist, wird die innere Verzweigungsbedingung evaluiert. Falls hier $b_i \geq 2^{w-1}$ gilt, wird 2^w von b_i abgezogen. Folglich verzweigt w -NAF in drei Fällen und vollzieht eine Subtraktion in Abhängigkeit des Exponenten.

Wir definieren eine “ \mathcal{SN} – Sequenz” wie folgt [SS04]:

- Falls bei der i -ten Schleife der Hauptschleife die Subtraktion zwei mal ausgeführt wird, bezeichnen wir den gemessenen Stromverbrauch mit \mathcal{SSN} .
- Falls die Subtraktion nur einmal vollzogen wird, kennzeichnen wir den beobachteten Stromverbrauch mit \mathcal{SN} .
- Falls keine Subtraktion stattfindet, wird dies mit \mathcal{N} gekennzeichnet.

Die \mathcal{SN} -Sequenz ist eine Folge von \mathcal{SSN} , \mathcal{SN} und \mathcal{N} , die bei der Berechnungsphase während der Rekodierung beobachtet wurden:

$$\mathcal{SN}\text{-Sequenz} := \{\mathcal{SSN}, \mathcal{SN}, \mathcal{N}\}^*$$

Der Angreifer kann also durch Monitoring des Stromverbrauchs zum Beispiel eines kryptographischen Gerätes die \mathcal{SN} -Sequenzen ermitteln. Und genau diese \mathcal{SN} -Sequenz stellt die Seitenkanalinformation dar.

Im Falle der vorzeichenbehafteten/vorzeichenlosen fractional window Rekodierung (Algorithmus 5.1.3a und 5.1.3b) hängen die Verzweigungsbedingungen auch vom Wert des Exponenten ab. Folglich kann der Angreifer die \mathcal{SN} -Sequenzen während der Berechnungsphase der Rekodierung ermitteln.

5.1.7 Der Angriff

Im Folgenden beschreiben wir die Angriffe auf die Exponenten Rekodierung für w -NAF und für die vorzeichenbehaftete/vorzeichenlose fractional window Repräsentation [SS04]. Der Angreifer versucht durch Ermittlung der \mathcal{SN} -Sequenzen den geheimen Exponenten zurück zu gewinnen. Wir werden hierbei die Algorithmen 5.1.2, 5.1.3a) und 5.1.3b) betrachten, die eine Exponentenrekodierung implementiert haben.

5.1.7.1 Grundlegende Vorgehensweise

Die w -NAF Rekodierung hat drei Verzweigungen.

Für einen gegebenen Exponenten ist die \mathcal{SN} -Sequenz eindeutig festgelegt. Zum Beispiel für eine 3-NAF Rekodierung, implementiert durch Algorithmus 5.1.2, sind die \mathcal{SN} -Werte ($\in \{\mathcal{SSN}, \mathcal{SN}, \mathcal{N}\}$) für jeden i -ten Schleifendurchlauf in der Tabelle 5.1 wie folgt [SS04]:

Tabelle 5.1: Relation zwischen Fensterinhalt & \mathcal{SN} -Werten für 3-NAF

Fensterinhalt	Zugeordnete Ziffer	\mathcal{SN} -Wert
$0 = (000)_2$	0	\mathcal{N}
$1 = (001)_2$	1	\mathcal{SN}
$2 = (010)_2$	0	\mathcal{N}
$3 = (011)_2$	3	\mathcal{SN}
$4 = (100)_2$	0	\mathcal{N}
$5 = (101)_2$	-3	\mathcal{SSN}
$6 = (110)_2$	0	\mathcal{N}
$7 = (111)_2$	-1	\mathcal{SSN}

Grundsätzlich kann ein Angriffsszenario basierend auf Tabelle 5.1 konstruiert werden. Jedoch besteht die Schwierigkeit im Erraten des Wertes des geheimen Exponenten bei gegebener \mathcal{SN} -Sequenz. Obwohl die \mathcal{SN} -Sequenz für einen gegebenen Exponenten eindeutig bestimmt ist, ist die Konvertierung nicht korrekt. Falls \mathcal{SN} beobachtet wird, sollte der Fensterinhalt $(001)_2$ oder $(011)_2$ sein. Daher kann der Angreifer das LSB und das MSB des Fensters bestimmen, aber das mittlere Bit kann

nicht eindeutig geraten werden. Dieselbe Situation tritt auch bei \mathcal{SSN} auf. Die zweite Schwierigkeit stellt das “*Carry*” dar. Falls der Fensterinhalt ungerade ist und $> 2^{w-1}$, muss 2^w von der zugeordnete Ziffer b_i subtrahiert werden, sodass das resultierende b_i (Schritt 6) einen negativen Wert haben wird. Die folgende Subtraktion (Schritt 8) sollte daraufhin eine Addition sein und ein Carry wird immer vorkommen. Infolgedessen wird der temporäre Exponent c in der nächsten $(i+1)$ -ten Schleife ein abweichendes Bitmuster gegenüber dem ursprünglichen Exponenten d haben. Beispiel: Nehmen wir an, dass $45 = (101101)_2$ der gegebene Exponent d sei und die 3-NAF Rekodierung ausgeführt werde. Im ersten Schleifendurchlauf sollte die dazugehörige Ziffer $b_i - 3$ (Schritt 6) sein, sodass in Schritt 8 der temporäre Exponent $c = 48 = (110000)_2$ sein sollte. Da der Carry in einem späteren Schleifendurchlauf auftreten wird, kann der erhaltene \mathcal{SN} -Wert keine direkte Information zu dem Wert des Exponenten sein.

Die Attacken müssen also die genannten Schwierigkeiten berücksichtigen.

5.1.7.2 Angriff auf die w -NAF Rekodierung

Wir zeigen nun einen Angriff auf die w -NAF Rekodierung [SS04]. Der Algorithmus der Attacke beruht auf folgenden Beobachtungen:

- Im Falle von \mathcal{SSN} tritt ein Carry in Schritt 8 auf.
- Im Falle von \mathcal{SSN} und alle durchlaufenen Werte der vorher beobachteten \mathcal{SSN} waren \mathcal{N} s, wird der Carry weitergereicht zum aktuellen \mathcal{SSN} . Zum Beispiel Subsequenz $(\mathcal{SSN}, \mathcal{N}, \mathcal{N}, \dots, \mathcal{N}, \mathcal{SSN})$.
- Im Falle von \mathcal{SSN} und falls der Carry weitergereicht wurde, sollte der Angreifer $d_i = 0$ annehmen. Andernfalls, wenn der Carry nicht weitergereicht wurde, sollte der Angreifer $d_i = 1$ annehmen.
- Im Falle von \mathcal{SSN} , sollte der Angreifer $d_{i+w-1} = 1$ annehmen.
- Im Falle von \mathcal{N} und bei weitergereichtem Carry, sollte der Angreifer $d_i = 1$ annehmen. Andernfalls, bei nicht weitergereichtem Carry, sollte der Angreifer $d_i = 0$ annehmen.
- Im Falle von \mathcal{SN} sollte der Angreifer zur Bestimmung des d_i nach der gleichen Strategie wie im Falle von \mathcal{SSN} vorgehen.
- Im Falle von \mathcal{SN} und angenommen die Länge der restlichen Bits sei kleiner als die Fenstergröße w , sollte der Angreifer $d_{t-1} = 1$ annehmen. Strikt nach Definition, dass das MSB von d immer “1” ist.
- Im Falle von \mathcal{SN} wird die Weiterreichung des Carry’s unterbrochen.

Der Angriff wird in Algorithmus 5.1.7.2 beschrieben [SS04]. Die Variable “*state*” wird für das Carry benutzt. Falls Daten im mittleren Teil des Fensters nicht eindeutig bestimmt werden können, wird die Variable “*unknown*” benutzt.

ALG 5.1.7.2: Angriff auf die w -NAF Rekodierung:

Eingabe: \mathcal{SN} -Sequenz (v_0, v_1, \dots) mit $v_i \in \{\mathcal{SSN}, \mathcal{SN}, \mathcal{N}\}$ und Integer $w \geq 2$,
 $t = \text{Bitlänge von } d$

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

1. $i \leftarrow 0$
2. $state \leftarrow 0$
3. while $i < t$ do
4. if $v_i = \mathcal{SSN}$ then
5. if $state = 1$ then $d_i \leftarrow 0$
6. else if $state = 0$ then $d_i \leftarrow 1$
7. for j from $i + 1$ to $i + w - 2$ do $d_j \leftarrow \text{“unknown”}$
8. $d_{i+w-1} \leftarrow 1$
9. $i \leftarrow i + w$
10. $state \leftarrow 1$
11. else if $v_i = \mathcal{SN}$ then
12. if $i + w - 1 > t$
13. if $state = 1$ then $d_i \leftarrow 0$
14. else if $state = 0$ then $d_i \leftarrow 1$
15. for j from $i + 1$ to $t - 2$ do $d_j \leftarrow \text{“unknown”}$
16. $d_{t-1} \leftarrow 1$
17. $i \leftarrow t$
18. else
19. if $state = 1$ then $d_i \leftarrow 0$
20. else if $state = 0$ then $d_i \leftarrow 1$
21. for j from $i + 1$ to $i + w - 2$ do $d_j \leftarrow \text{“unknown”}$
22. $d_{i+w-1} \leftarrow 0$
23. $i \leftarrow i + w$
24. end if
25. $state \leftarrow 0$
26. else if $v_i = \mathcal{N}$ then
27. if $state = 1$ then $d_i \leftarrow 1$
28. else if $state = 0$ then $d_i \leftarrow 0$
29. $i \leftarrow i + 1$
30. end if
31. end while
32. Return $d = (d_{t-1} \dots d_0)_2$

Wir können nun abschätzen wie viele Bits anhand der beobachteten \mathcal{SN} -Sequenzen im Falle der w -NAF Rekodierung wiederhergestellt werden können.

Theorem 1.7

Nehmen wir an, dass die w -NAF Rekodierung in Algorithmus 5.1.2 implementiert ist und die \mathcal{SN} -Sequenz während der Rekodierung beobachtet werden kann. Das Verhältnis der erfolgreich wiederhergestellten Bits kann durch $3/(w + 1)$ für $t \rightarrow \infty$ bewertet werden [SS04].

Beweis.

Die Tatsache, dass die Dichte der w -NAF Rekodierung $1/(w + 1)$ für $t \rightarrow \infty$ liefert den Beweis [SS04]. □

5.1.7.3 Angriff auf die Vorzeichenlose Fractional Window Rekodierung

Wir zeigen nun einen Angriff auf die *vorzeichenlose fractional window* Rekodierung [SS04]. Ähnliche Ansätze wie im Falle der w -NAF Rekodierung können angewandt werden. Datenabhängige Subtraktionen müssen in Schritt 6 und 8 des Algorithmus 5.1.3a). Daher kann der Angreifer die \mathcal{SN} -Sequenz während der Rekodierung beobachten. Der Unterschied zum Fall w -NAF ist, dass selbst wenn die Subtraktion in Schritt 8 durchgeführt wird, kein Carry auftritt.

Der Angriff wird in Algorithmus 5.1.7.3 gezeigt [SS04].

ALG 5.1.7.3: Angriff auf die Vorzeichenlose Fractional Window Rekodierung:

Eingabe: \mathcal{SN} -Sequenz (v_0, v_1, \dots) mit $v_i \in \{\mathcal{SSN}, \mathcal{SN}, \mathcal{N}\}$, ein Integer $w \geq 2$,
ein ungerader Integer m , $1 \leq m \leq 2^w - 3$, $t = \text{Bitlänge von } d$

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

1. $i \leftarrow 0$
2. while $i < t$ do
3. if $v_i = \mathcal{SSN}$ then
4. $d_i \leftarrow 1$
5. for j from $i + 1$ to $i + w - 1$ do $d_j \leftarrow \text{“unknown”}$
6. $i \leftarrow i + w$
7. else if $v_i = \mathcal{SN}$ then
8. $d_i \leftarrow 1$
9. if $i + w + 1 > t$
10. for j from $i + 1$ to $t - 2$ do $d_j \leftarrow \text{“unknown”}$
11. $d_{t-1} \leftarrow 1$
12. $i \leftarrow t$
13. else
14. for j from $i + 1$ to $i + w$ do $d_j \leftarrow \text{“unknown”}$
15. $i \leftarrow i + w + 1$
16. end if
17. else if $v_i = \mathcal{N}$ then
18. $d_i \leftarrow 0$
19. $i \leftarrow i + 1$
20. end if
21. end while
22. Return $d = (d_{t-1} \dots d_0)_2$

5.1.7.4 Angriff auf die Vorzeichenbehaftete Fractional Window Rekodierung

Ein Angriff auf die *vorzeichenbehaftete fractional window* Rekodierung wird in Algorithmus 5.1.7.4 gezeigt [SS04]. Die Vorgehensweise ist ähnlich zu der der w -NAF Rekodierung, aber die Handhabung des Carry's gestaltet sich schwierig. Nur wenn $w + 1$ fortlaufende \mathcal{N} s nach \mathcal{SSN} beobachtet werden, könnte ein Carry zu \mathcal{SN} oder \mathcal{SSN} weitergereicht werden. In Variable c in Algorithmus 5.1.7.4 wird die Anzahl der fortlaufenden \mathcal{N} s gespeichert.

ALG 5.1.7.4: Angriff auf die Vorzeichenbehaftete Fractional Window Rekodierung:

Eingabe: \mathcal{SN} -Sequenz (v_0, v_1, \dots) mit $v_i \in \{\mathcal{SSN}, \mathcal{SN}, \mathcal{N}\}$ und Integer $w \geq 2$,
ein ungerader Integer m , $1 \leq m \leq 2^w - 3$, $t = \text{Bitlänge von } d$

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

1. $i \leftarrow 0$
2. $state \leftarrow 0$
3. $c \leftarrow 0$
4. while $i < t$ do
5. if $v_i = \mathcal{SSN}$ then
6. if $c \geq w + 1$ und $state = 1$ then $d_i \leftarrow 0$
7. else if $c = w$ und $state = 1$ then $d_i \leftarrow \text{“unknown”}$
8. $d_i \leftarrow 1$
9. for j from $i + 1$ to $i + w$ do $d_j \leftarrow \text{“unknown”}$
10. $i \leftarrow i + w + 1$
11. $c \leftarrow w$
12. $state \leftarrow 1$
13. else if $v_i = \mathcal{SN}$ then
14. if $c \geq w + 1$ und $state = 1$ then $d_i \leftarrow 0$
15. else if $c = w$ und $state = 1$ then $d_i \leftarrow \text{“unknown”}$
16. else $d_i \leftarrow 1$
17. if $i + w + 1 > t$
18. for j from $i + 1$ to $t - 2$ do $d_j \leftarrow \text{“unknown”}$
19. $d_{t-1} \leftarrow 1$
20. $i \leftarrow t$
21. else
22. for j from $i + 1$ to $i + w$ do $d_j \leftarrow \text{“unknown”}$
23. $d_{i+w+1} \leftarrow 0$
24. $i \leftarrow i + w + 2$
25. end if
26. $c \leftarrow w$
27. $state \leftarrow 0$
28. else if $v_i = \mathcal{N}$ then
29. if $state = 1$ then $d_i \leftarrow 1$
30. else if $state = 0$ then $d_i \leftarrow 0$
31. $c \leftarrow c + 1$
32. $i \leftarrow i + 1$
33. end if
34. end while
35. Return $d = (d_{t-1} \dots d_0)_2$

5.1.7.5 Versuchsergebnisse

Es wurden Versuche zu den oben beschriebenen Angriffen wie folgt durchgeführt [SS04]:

1. 10.000 zufällige generierte Exponenten d mit 160-Bits, 512-Bits oder 1024-Bits.
2. Die Algorithmen 5.1.2, 5.1.3a) und 5.1.3b) für die Exponentenrekodierung implementiert in der Programmiersprache C.
3. \mathcal{SN} -Sequenzen Generierung.
4. Versuch mit Hilfe der \mathcal{SN} -Sequenzen und den Algorithmen 5.1.7.2, 5.1.7.3 und 5.1.7.4 auf den geheimen Schlüssel d zu schließen.
5. Zählung der erfolgreich aufgedeckten Bits (= Anzahl der aufgedeckten Bits / Bitlänge des Exponenten d).

Die Ergebnisse sind in Tabelle 5.2 festgehalten [SS04]. Die Experimente wurden mit 160-Bit, 512-Bit und 1024-Bit großen Exponenten durchgeführt. Beinahe in jedem Fall wurde derselbe Prozentsatz erzielt.

Die Fensterbreite w wurde von 2 bis 5 untersucht. In der fractional window Erweiterung wurde der Parameter m von 1 bis hin zur oberen Grenze untersucht, beispielsweise $2^w - 3$. Zwischenliegendes ($2 \leq m \leq 2^w - 4$) wurde aus Platzgründen weggelassen. Die erfolgreich aufgedeckten Bits nahmen mit größerem w ab, weil, wie schon erwähnt, die Bits im mittleren Teil des Fenster nicht eindeutig bestimmt werden können. Bei der fractional window Erweiterung nimmt die Anzahl der erfolgreich aufgedeckten Bits mit größerem m zu.

Hinweis: Es traten keine Fehlannahmen in den Attacken auf. Nur die “unknown”-Bits können unaufgedeckte Bits sein.

Tabelle 5.2: Versuchsergebnisse
Aufgedeckte Bits (%) (=Anzahl aufgedeckter Bits/Bitlänge des Exponenten d)

w	m	w -NAF	vorz.los.fract.	vorz.beh.fract.
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
	29	–	29.1	37.1

5.1.7.6 Schlussfolgerung

Kapitel 4 verdeutlicht, dass obwohl die Exponentenrekodierung sorgfältig implementiert wurde, Kryptosysteme basierend auf elliptischen Kurven verwundbar gegen SPA's sind [SS04].

Während die Effekte eines einzelnen Transistors normalerweise unmöglich durch direkte Beobachtung des Stromverbrauches eines Gerätes zu ermitteln sind, sind die statistischen Operationen in der Lage außergewöhnlich geringe Unterschiede im Stromverbrauch zuverlässig zu erkennen [KuBJ].

Kapitel 6

Kombinierte Schwachstellenanalyse

In diesem Kapitel wollen wir nun unsere Schwachstellenanalyse vorstellen. Wir befassen uns dabei mit Algorithmen für die schnelle Exponentiation. Unsere Schwachstellenanalyse betrachtet die Kombination aus Punktvorbereitung, Exponentenrekodierung und einem quasi “left-to-right”-Algorithmus.

6.1 Konzept

Die Grundidee fußt auf der Kenntnis des verwendeten Algorithmus zur schnellen Exponentiation bei elliptischen Kurven. Darüber hinaus kann der Angreifer in dem jeweiligen Szenario sich einiger Hilfsmittel bedienen. Diese Hilfsmittel sind bekannt als Seitenkanalangriffe [Koc96b].

Die Idee dahinter ist, dass es nicht allein der Lösung des mathematischen Problems bedarf, um ein Geheimnis zu ermitteln, sondern auch andere Wege zur Entschlüsselung führen [Koc96b]. Seitenkanalangriffe analysieren Eigenschaften des Algorithmus und damit kann auf das Geheimnis desselben geschlossen werden. Dazu wird eine oder eine große Anzahl von Messungen gemacht, die dann statistisch analysiert werden. Diese Angriffe beziehen sich auf alle messbaren Dinge, über die ein Angreifer verfügen kann. Diese sind, je nach dem welches kryptographische Gerät angegriffen wird, verschieden.

Zu den bekanntesten Seitenkanalangriffen zählen die Timing Attacks (TA) [Koc96b], die Differential Power Analysis (DPA) [KuBJ] [KuBJ99] und die Simple Power Analysis (SPA) [C.W04].

Timing Attacks sind “Chosen Plaintext”- bzw. “Chosen-Ciphertext”-Angriffe [Koc96b]. Dabei wird versucht, für eine bekannte Eingabe die Zeit, die für die Berechnung benötigt wird, exakt zu ermitteln. Dies ist insbesondere bei SmartCards sehr genau möglich. Es wird versucht, für viele Eingaben jeweils die Berechnungsdauer zu ermitteln. Es wird für jede Messung das Tupel (Eingabe, Berechnungsdauer) gespeichert. Zum Beispiel braucht man für das ursprüngliche RSA-Verfahren mindestens 5000 Messungen. Dies ist der erste Schritt des Angriffs, welcher nur sequentiell ausgeführt werden kann. Er benötigt daher sehr viel Zeit. Im zweiten Schritt werden

diese Daten analysiert. Da der Algorithmus standardisiert ist, ist der Ablauf der Berechnung mit Ausnahme der variablen Teile, die vom geheimen Schlüssel bestimmt sind, bekannt. Für jede Eingabe kann man den Algorithmus soweit abarbeiten, bis der erste variable Teil der Berechnung ausgeführt wird. Für diese Ausführung gibt es nur zwei Möglichkeiten, die abhängig davon sind, ob das Bit des geheimen Schlüssels den Wert 1 oder 0 hat. In der folgenden Teilberechnung, muss man nun einen Unterschied in der Berechnungsdauer finden, der nur vom vorher berechneten variablen Teil abhängt. Hier kann für RSA zum Beispiel die Montgomery Multiplikation [MMM03] verwendet werden. Unter der Annahme, dass für den Rest der Berechnung im Mittel die gleiche Berechnungszeit benötigt wird, kann man nun auf das Teilgeheimnis schließen. Die Zeitunterschiede in den Teilberechnungen werden dazu genutzt die Gesamtberechnungsdauern zu gruppieren. Sollten die getroffenen Annahmen richtig sein, wird ein zeitlicher Unterschied zwischen den Gruppen erkennbar sein. Falls die Annahmen falsch sind, ist keine zeitliche Differenz zu erkennen und das Gegenteil wird angenommen. Damit ist das betreffende Schlüsselbit bestimmt. Das Schlüsselbit wird für die folgenden Berechnungen als bekannt vorausgesetzt. Nun beginnt man von Neuem, indem man bis zu dem nächsten variablen, dann ersten, Teil rechnet und die Prozedur wiederholt. Dieser Vorgang wird solange wiederholt, bis der Schlüssel im Ganzen gefunden ist bzw. der Rest des Schlüssels so klein ist, dass eine vollständige Suche die schnellere Angriffsvariante darstellt. Dieses Verfahren kann den geheimen Schlüssel nur ermitteln, wenn von Beginn an alle Schlüsselbits richtig ermittelt werden. Sobald eine falsche Annahme getroffen wird, sind alle weiteren Berechnungen falsch. Daher ist eine Fehlererkennung sehr wichtig. Sie wird in diesem Angriffsszenario gleich mitgeliefert. Da in der Folgeberechnung auf den variablen Teil, der vom Schlüssel abhängt, ein Zeitunterschied festgestellt werden soll, sind diese Berechnungen falsch. Dies ist in der Berechnung begründet, die durch den falsch ermittelten Beginn des geheimen Schlüssels, durchgeführt wird. Daher sind die zu erwartenden Zeitunterschiede nicht zu erkennen und es wird jeweils das Gegenteil der Annahme verwendet, der Schlüssel ist somit ab einer Stelle konstant. Dies lässt auf einen Fehler schließen. Man geht an die Stelle, an der der Fehler vermutet wird zurück und versucht es erneut mit einem neuen Stichprobenraum. Erstmals wurde diese Art von Angriff 1998 von einer Forschergruppe der katholischen Universität Louvain vorgestellt [DKPA⁺98].

Bei der Power Analysis ermittelt man den Strom-/Spannungsverlauf, der bei der Abarbeitung des Algorithmus entsteht. Da SmartCards über eine externe Stromversorgung mit Energie versorgt werden, ist dies ohne weiteres möglich. Aus der resultierenden Kurve versucht man dann auf die ausgeführten Operationen des Algorithmus zu schließen. Die Stromverbrauchsanalyse teilt sich in zwei Bereiche auf. Zum einen gibt es die "Simple Power Analysis" (SPA) und zum anderen die "Differential Power Analysis" (DPA). In der Literatur wird das Verhältnis der beiden zueinander unterschiedlich betrachtet. Die einen sehen die SPA als Teil der DPA, andere sprechen von verschiedenen Angriffen. Daher werden hier lediglich die beiden Varianten beschrieben. Der Leser mag sich dann für eine der beiden Varianten entscheiden. Den ersten Angriff dieser Art stellte wiederum Kocher vor [Koc96a].

Bei der SPA-Methode betrachtet man lediglich eine Kurve des Spannungsverlaufs. Hier versucht man nun, dem Muster einzelne Operationen wie Multiplikationen, Ad-

ditionen oder Shifts zuzuordnen. Aus deren Aneinanderreihung versucht man, den Ablauf des Algorithmus zu rekonstruieren. Zur Verdeutlichung ein Beispiel: Bei RSA wird eine Quadrierung (Q) durchgeführt, wenn das betreffende Schlüsselbit eine 0 ist. Ist es jedoch 1, wird zusätzlich eine Multiplikation (M) erforderlich. Wenn sich an der Spannungskurve nun die Folge QMQQMQQMQMQMQM ablesen lässt, ergibt sich der zugehörige Schlüssel 101001111, denn "QM" bedeutet eine 1 und die verbleibenden "Q" repräsentieren jeweils eine 0 [C.W04].

Da die SPA mit einfachen Mitteln (Dummy Operationen) unschädlich gemacht werden kann, nutzt die DPA ähnlich wie die Timing Attacks statistische Analysetechniken. Hiermit können auch Algorithmen, die unabhängig vom geheimen Schlüssel die gleiche Berechnungsstruktur haben, aufgrund spezieller Effekte angegriffen werden. Der Angreifer misst zunächst viele Spannungsverläufe, indem er Chosen Plain-/Ciphertexte wählt und das Verfahren durchführt. Anschließend gruppiert er die Spannungsverläufe nach dem von ihm gewählten Prinzip des Angriffs. Dieses bezieht sich auf ausgewählte Operationen, beispielsweise auf eine Multiplikation mit 0, die einen festen vorher ermittelten Ablauf in dem Spannungsverlauf hat. Wenn nun die Annahme richtig ist, wird bei der Überlagerung der Kurven an diesem eben bestimmten Punkt ein Höchstwert zu erkennen sein, wobei der Rest der Überlagerung sich um den Mittelwert einpendelt. Dieser Effekt tritt ein, weil nur an genau dieser Stelle der Algorithmus das einheitliche Verhalten bezüglich des Spannungsverlaufs hat. Sollte das tatsächlich so sein, ist der Angreifer sicher, dass seine Vermutung richtig ist. Ansonsten nimmt er das Gegenteil der Vermutung an und somit ist ein Teil des Geheimnisses bekannt. Darauf aufbauend, kann der Angreifer mit dem gleichen Angriff weitere Teile des Schlüssels herausfinden.

Eine SmartCard bietet wohl die größte Angriffsfläche. Hier kann man die Berechnungsdauer (Timing Attack) und den Stromverbrauch (Power Analysis) eines Algorithmus messen. Darüber hinaus steht die SmartCard meist dem Angreifer beliebig lange zur Verfügung und er kann akribisch an ihr testen. Daher liegt es nahe, alle Emissionen, die vom Prozessor während der Berechnung ausgehen, für einen Angriff zu nutzen.

Die jeweilige richtige Wahl des Analyseverfahrens zu treffen ist Aufgabe des Angreifers. Sie hängt auch davon ab, wo sich unser Angreifer aktuell befindet bzw. welche Art von Leck die nötigen Informationen dem Angreifer zukommen lässt.

Defintion 5.1 (Geburtstagsparadoxon) [Buc04]

Das Geburtstagsparadoxon gehört zur Wahrscheinlichkeitsrechnung. Es gibt n verschiedene Geburtstage für k Personen innerhalb eines Raumes. Ein Elementarereignis besteht aus einem Tupel $(g_1, \dots, g_k) \in \{1, 2, \dots, n\}^k$. Tritt es ein, so hat die i -te Person den Geburtstag g_i , $1 \leq i \leq k$. Es gibt n^k Elementarereignisse. Wir nehmen an, dass alle Elementarereignisse gleich wahrscheinlich sind. Jedes Elementarereignis hat daher die Wahrscheinlichkeit $\frac{1}{n^k}$.

Die Wahrscheinlichkeit, dass wenigstens zwei Personen am gleichen Tag Geburtstag haben wird mit p bezeichnet. $q = 1 - p$ ist die Wahrscheinlichkeit dafür, dass alle Personen an verschiedene Geburtstage haben. Das Ereignis E ist die Menge aller

Vektoren $(g_1, \dots, g_k) \in \{1, 2, \dots, n\}^k$, deren sämtliche Koordinaten verschieden sind. Alle Elementarereignisse habe die gleiche Wahrscheinlichkeit $\frac{1}{n^k}$. Die Wahrscheinlichkeit für E ist demnach die Anzahl der Elemente in E multipliziert mit $\frac{1}{n^k}$. Die Anzahl der Vektoren in $\{1, \dots, n\}^k$ mit verschiedenen Koordinaten bestimmt sich wie folgt: Auf der ersten Position können n Zahlen stehen. Liegt die erste Position fest, so können auf der zweiten Position noch $n - 1$ Zahlen stehen usw. Es gilt also

$$|E| = \prod_{i=0}^{k-1} (n - i).$$

Die gesuchte Wahrscheinlichkeit ist

$$q = \frac{1}{n^k} \prod_{i=0}^{k-1} (n - i) = \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right). \quad (5.1)$$

Nun gilt aber $1 + x \leq e^x$ für alle reellen Zahlen. Daher folgt aus (5.1)

$$q \leq \prod_{i=1}^{k-1} e^{-i/n} = e^{-\sum_{i=1}^{k-1} i/n} = e^{-k(k-1)/(2n)}. \quad (5.2)$$

Ist

$$k \geq (1 + \sqrt{1 + 8n \log 2})/2 \quad (5.3)$$

so folgt aus (5.2), dass $q \leq \frac{1}{2}$ ist. Dann ist die Wahrscheinlichkeit $p = 1 - q$ dafür, dass zwei Personen im Raum am gleichen Tag Geburtstag haben $\geq \frac{1}{2}$. Für $n = 365$ genügt $k = 23$, damit $q \leq \frac{1}{2}$ ist. Sind also 23 Personen im Raum, so haben mit Wahrscheinlichkeit $\geq \frac{1}{2}$ wenigstens zwei am gleichen Tag Geburtstag. Allgemein genügen etwas mehr als \sqrt{n} viele Personen damit zwei am gleichen Geburtstag haben.

6.1.1 Das Szenario I

Wir haben eine SmartCard [RE02] mit einem integrierten Elliptischen-Kurven-Kryptosystem zur Verschlüsselung und Entschlüsselung. Der private Schlüssel ist auf der SmartCard gespeichert und kann diese nicht verlassen. Der private Schlüssel kann somit nicht erspäht werden. Da die SmartCard nicht viel physischen Platz bietet und auch von der Rechenleistung sehr begrenzt ist, kommt eine teure Randomisierungsfunktion, im speziellen die Randomisierung des Exponenten, nicht in Betracht. Desweiteren muss die Verschlüsselung / Entschlüsselung aufgrund der geringen Rechenleistung der Karte äußerst effizient ablaufen. Daher muss eine Rekodierung des privaten Schlüssels stattfinden. Die Rekodierung erfolgt in unserem Szenario mit dem w -NAF Algorithmus 4.1.4 mit einer integrierten Punktvorbereitung. Dabei erfolgt die Punktvorbereitung nicht separat. Sie erfolgt zeitnah zum w -NAF Algorithmus, sodass tatsächlich nur die benötigten Punkte vorberechnet werden. Der Angreifer kennt diesen verwendeten Rekodierungsalgorithmus und hat Zugriff auf die Elektronik der Schnittstelle mit der die SmartCard verbunden ist. Mittels SPA kann der Angreifer nun einen Verschlüsselungsvorgang einer Nachricht genauestens ana-

lysieren. Er nutzt das Leck “Strom”, stellt somit geringe Stromunterschiede je nach Berechnungsphase fest und kann so in Kombination mit SPA anhand der Rekodierungsphase und anhand der beobachteten vorberechneten Punkte den originalen, geheimen Schlüssel rekonstruieren.

Tabelle 6.1 zeigt ein mögliches Beispiel einer aus einer SPA gewonnenen binären Lösung in anbetracht der oben genannten Gegebenheiten und unter Verwendung von 4-NAF. D steht dabei für die Punktverdoppelung und A steht für die Punktaddition. Das MSB ist immer 1.

Tabelle 6.1: Analyse mittels SPA I

SPA:		D D D D A A D D D D A A A D				
Sortiert:		D D D	D A A	D D D	D A A A	D
Lösung:	1	0 0 0	$\pm 3P$	0 0 0	$\pm 7P$	0

An obiger SPA Folge sehen wir die Verwundbarkeit eines effizienteren w -NAF Algorithmus. Zusätzlich sei noch erwähnt, dass ein “D...A”-Paar für eine Punktverdoppelung und Punktaddition des eigentlichen w -NAF Algorithmus steht. Alle A’s (Punktaddition) dazwischen, stehen für die Punkt Vorbereitung. Bei der Folge “DAA” bedeutet dies, dass eine Punktverdoppelung und eine Punktaddition mit $\pm 3P$ stattgefunden hat. Für die Folge “DAAA” heißt das, dass eine Punktverdoppelung und eine Punktaddition mit dem vorberechneten Punkt $7P$ statt gefunden hat. Der vorberechnete Punkt lautet hier $7P$, da wir zwei A’s zwischen dem “D...A”-Paar stehen haben. Hieraus sehen wir, dass die Punkt Vorbereitung aktuell zwei mal durchlaufen worden. Insgesamt sind haben wir die Punkt Vorbereitung drei mal durchlaufen, woraus sich schließlich der Punkt $7P$ ergibt.

Unser obiges Beispiel mit 4-NAF bietet also 4 Lösungsmöglichkeiten, unabhängig von den tatsächlich vorberechneten Punkten. Die möglichen Kombinationen sind in Tabelle 6.2 dargestellt:

Tabelle 6.2: Analyse mittels SPA II

...	+	...	+	...
...	+	...	-	...
...	-	...	+	...
...	-	...	-	...

Anhand der Messung durch SPA, der Kenntnis des verwendeten Algorithmus und der nun vorliegenden vorberechneten Punkte, kann der Angreifer auf den geheimen Exponenten schließen. Der Angreifer braucht nur die richtige Kombination der Vorzeichen zu erraten. Das geht in unserem Beispiel mit 2^2 Möglichkeiten sehr schnell. Wenn wir das Ganze für einen 160bit großen Schlüssel, unter Verwendung von 4-NAF, hochrechnen, braucht der Angreifer grob nur 2^{40} Kombinationen zu untersuchen. Die 2^{40} Kombinationen ergeben sich aus der in unserem Fall betrachteten Fenster hochgerechnet auf die Anzahl der Fenster bei einem 160bit großen Schlüssel

mit 4-NAF. Das heißt, ein 160bit großer Schlüssel hat mit 4-NAF in unserem Beispiel 40 Fenster. Wir betrachten immer zwei Fenster gleichzeitig, welche jeweils 2^2 Möglichkeiten der Vorzeichen aufweisen. Daraus ergibt sich die Untersuchung von 20 “Doppelfenstern”, sprich

$$\underbrace{2^2 \times 2^2 \times \dots \times 2^2}_{20\text{-mal}} = 2^{40}$$

Kombinationen.

Unter Hinzunahme des Geburtstagsparadoxon könnte ein Erfolg sich schon bei

$$\sqrt{2^{40}} = 2^{20}$$

zu untersuchenden Kombinationen einstellen. Dies ist in vertretbarer Rechenzeit lösbar.

Die exakte Rechenzeit lässt sich nur grob hochrechnen, da hier noch weitere Faktoren zur Effizienzsteigerung eine bedeutende Rolle spielen. Zu nennen wären hier, das verwendete Koordinatensystem (affin, projektiv, jakobisch . . .), der verwendete Rekodierungsalgorithmus und die für den Angriff verwendete Rechenmaschine.

Wir können mit diesem Angriff jede Fenstergröße w eines w -NAF Algorithmus brechen. Mit Hilfe von SPA können wir genau feststellen welche Punkte vorberechnet werden, jedoch leider vorerst nicht das Vorzeichen eindeutig bestimmen. Hier kommen die oben genannten Überlegungen zur Bedeutung.

Allgemein lässt sich sagen, dass durch den oben aufgeführten Angriff der geheime Exponent vollständig erschlossen werden kann. Aus insgesamt 2^{120} möglichen Kombinationen (siehe dazu Szenario II), welche sich aus der Verwendung mit w -NAF ergeben, brauchen wir nur 2^{20} Kombinationen zu untersuchen. Das sind lediglich $7,88 \times 10^{-29}$ Prozent zu untersuchende Schlüssel.

6.1.2 Das Szenario II

Das zweite Szenario bezieht sich auf eine mit dem Elliptischen-Kurven-Kryptosystem erzeugte Signatur [Vau04]. Mit unserem privaten Schlüssel signieren wir die Nachricht und der Empfänger kann die Signatur mit dem dazugehörigen öffentlichen Schlüssel verifizieren. Als privaten Schlüssel wählen wir ein zufälliges e und berechnen damit den öffentlichen Schlüssel $Q = e \cdot P$. Der öffentliche Schlüssel ist meist von einer vertrauenswürdigen CA (certificate authority, Zertifizierungsstelle) zertifiziert und ermöglicht so erst eine vernünftige Verifizierung der Signatur. Die Zertifizierung gilt aber nur für einen öffentlichen Schlüssel. Das bedeutet, dass wir für jeden neuen öffentlichen Schlüssel ein neues Zertifikat benötigen. Der Zertifizierungsakt ist allerdings sehr zeitaufwändig und kostenintensiv. Bei der Signaturerzeugung mit einem Elliptischen-Kurven-Kryptosystem muss vorher ein fester Basispunkt P gewählt werden, um einen eindeutigen, einzigartigen öffentlichen bzw. privaten Schlüssel zu generieren. In Folge dessen und aufgrund des mühsamen Zertifizierungsaktes, wird auch hier der Einsatz der Randomisierung, im speziellen die Randomisierung des Basispunktes, nicht in Erwägung gezogen.

Wir nehmen für unser Szenario eine SmartCard Architektur an. Da wir uns auch bei Signaturen einen performanten Einsatz mit der SmartCard wünschen, wird hier die Rekodierung des privaten Schlüssels parallel zur Signierung stattfinden. Nehmen wir an, es sei wieder w -NAF zum Einsatz gekommen, konkret 4-NAF. Dies ist dem Angreifer bekannt. Darüber hinaus kennt er auch schon die hierfür benötigten, tatsächlich vorberechneten Punkte. Überdies ist unser Angreifer in den Besitz dieser SmartCard gelangen und kann sie nun ausgiebig testen. Der Angreifer besitzt ein SmartCard Lesegerät, welches mit seinem Rechensystem verbunden ist. Jetzt kann die Analyse mittels Timing Attack beginnen. Der Angreifer versucht geringste Zeitunterschiede der einzelnen Berechnungsphasen zu untersuchen. Nun signiert der Angreifer mehrere bekannte Klartexte, erhält jeweils die dazugehörige Signatur und misst die Laufzeitunterschiede der jeweiligen Berechnungsphase. Im speziellen wird versucht Additionsunterschiede zu messen. Es soll festgestellt werden, wann und wo ein Carrybit aufgetreten ist und welche Auswirkungen es auf den Rest des Schlüssels hat. Die Tabelle 6.3 zeigt zwei Fenster mit 4-NAF und Vorbereitung bis P wo Additionsunterschiede aufgetreten sein könnten.

Tabelle 6.3: Analyse mittels TA

Urspr.Schlüssel:	1 1 1 1 0 1 1 1 1		
T:	1 0 0 0 0 -1 0 0 0 -1		
Sortiert:	1 0	0 0 0 -1	0 0 0 -1
		Fenster 1	Fenster 2

In der Tabelle 6.3 sehen wir die Rekodierung des Ursprünglichen Schlüssels. Unser Angreifer hat feststellen können, dass in Fenster 1 $1/-1$ und in Fenster 2 auch eine $1/-1$ auftauchen könnte. Da der Angreifer auch noch die Additionsunterschiede, verursacht durch das jeweilige Carrybit, feststellen konnte, kann der Angreifer sich sicher sein, dass in Fenster 1 eine -1 und in Fenster 2 auch eine -1 stehen muss. Das MSB ist wie immer 1. Demzufolge lässt sich der original Schlüssel unseres Beispiels aus 6.3 eindeutig bestimmen und es müssen keine anderen Möglichkeiten berücksichtigt werden. Hochgerechnet für einen 160bit großen EKK Schlüssel können wir auch hier im Idealfall eine eindeutige Kombination bestimmen, gemäß obiger Annahme. Falls der Angreifer keine signifikanten Additionsunterschiede feststellen konnte, können für unser Beispiel die Inhalte der Fenster 1 und Fenster 2 wie folgt sein:

1.	1	1
2.	1	-1
3.	-1	1
4.	-1	-1
	Fenster 1	Fenster 2

Damit hätten wir schon 2^2 Möglichkeiten. Für einen 160bit großen EKK Schlüssel

sind das

$$(2^2)^{20} = 2^{40}$$

mögliche Kombinationen.

Ziehen wir das Geburtstagsparadoxon hinzu kommen wir auf

$$\sqrt{2^{40}} = 2^{20}$$

zu untersuchende Kombinationen.

Dies ist in polynomieller Rechenzeit lösbar.

Betrachten wir das Ganze für alle möglichen Kombinationen die es bei 4-NAF geben kann, so kämen wir auf insgesamt 2^{120} Kombinationen. Nachfolgende Tabelle 6.4 zeigt diese Kombinationen.

Tabelle 6.4: Mögliche Kombinationen für 4-NAF

Fenster 1	Fenster 2	‡ Kombinationen
± 1	± 1	4
± 1	± 3	4
± 1	± 5	4
± 1	± 7	4
± 3	± 1	4
± 3	± 3	4
± 3	± 5	4
± 3	± 7	4
± 5	± 1	4
± 5	± 3	4
± 5	± 5	4
± 5	± 7	4
± 7	± 1	4
± 7	± 3	4
± 7	± 5	4
± 7	± 6	4

Insgesamt kommen wir auf 64 Möglichkeiten für zwei Fenster. Für einen 160bit großen Schlüssel wären wir demnach bei

$$(2^6)^{20} = 2^{120}$$

möglichen Kombinationen.

Hier wird nochmal die “Effizienz” unseres Angriffes deutlich. Wir müssen nicht alle 2^{120} Möglichkeiten untersuchen, sondern nur 2^{20} .

Falls der Angreifer keine Additionsunterschiede feststellen kann und bis $3P$ vorberechnet worden ist, kommen wir wegen

1.	± 1	± 1
2.	± 1	± 3
3.	± 3	± 1
4.	± 3	± 3
	Fenster 1	Fenster 2

auf

$$2^4 = 16$$

Kombinationen.

Für einen 160bit großen Schlüssel folgt daraus

$$(2^4)^{20} = 2^{80}.$$

Mit Geburtstagsparadoxon kommen wir auf

$$\sqrt{2^{80}} = 2^{40}$$

Möglichkeiten.

Dies ist nicht mehr mit vertretbarem Rechenaufwand lösbar. Dies bedeutet aber keine absolute Sicherheit, sondern kann als "gerade noch Sicher" qualifiziert werden.

6.1.3 Das Szenario III

Wir wollen nun einen Hochleistungsrechner angreifen, welcher aber keine Randomisierungsfunktion benutzt. Der Angreifer hat auch hier Zugang zu den Leiterbahnen, um den Stromverbrauch messen zu können. Zur Verschlüsselung / Entschlüsselung wird hier w -NAF mit separater Rekodierung verwendet. Konkret handelt es sich um 4-NAF. Zuerst misst der Angreifer viele Spannungsverläufe. Dabei wählt er verschiedene Chosen Plain-/Ciphertexte und führt den Algorithmus aus. Anschließend gruppiert er die Spannungsverläufe nach Punktaddition und Punktverdoppelung. Nun überlagert er die jeweiligen Spannungskurven übereinander. Wie oben schon erwähnt, werden an bestimmten Punkten Höchstwerte zu erkennen sein. Der Rest der Überlagerung wird sich um den Mittelwert einpendeln.

Unser Angreifer kennt die vorberechneten Punkte. In unserem Beispiel wurde nur bis $3P$ vorbereitet. Die sich aus den Messungen ergebenden Werte für zwei Fenster stehen in Tabelle 6.5.

Tabelle 6.5: Analyse mittels DPA I

DPA:	1 0 0 0 0 x 0 0 0 x		
Sortiert:	1 0	0 0 0 x	0 0 0 x
		Fenster 1	Fenster 2

Nun gilt es herauszufinden, was an den mit x markierten Positionen der Bitfolge gestanden haben könnte. Mögliche Werte inklusive der vermuteten, ursprünglichen Bitfolgen stehen in Tabelle 6.6.

Tabelle 6.6: Analyse mittels DPA II

	Fenster 1	Fenster 2	mögliche Bitfolge
1.	1	-1	1 0 0 0 0 0 1 1 1 1
2.	1	-3	1 0 0 0 0 0 1 1 0 1
3.	3	-1	1 0 0 0 1 0 1 1 1 1
4.	3	-3	1 0 0 0 1 0 1 1 0 1

Anhand der Tabelle 6.6 sehen wir, dass uns vier Möglichkeiten für die x Einträge in dem jeweiligen Fenster zur Auswahl stehen. Das erste Fenster wird hierbei immer einen positiven x Eintrag haben und das zweite Fenster einen negativen x Eintrag. Der Grund hierfür liegt in der Beschaffenheit des w -NAF Algorithmus und in der zufälligen, für uns glücklichen Anordnung der Bitfolge. Im ersten Fenster haben wir eine Wahrscheinlichkeit von $\frac{1}{2}$, dass an der Position des x eine 1 oder 3 steht. Im zweiten Fenster haben wir auch eine Wahrscheinlichkeit von $\frac{1}{2}$, dass hier eine -1 oder -3 für das x in Frage kommt. Betrachten wir nun beide Fenster zusammen, kommen wir auf eine Wahrscheinlichkeit von $\frac{1}{4}$ ($=\frac{1}{2} \times \frac{1}{2}$), dass eine der in Tabelle 6.6 gezeigten Möglichkeiten eintritt.

Wenn wir uns das Ganze nun für einen 160bit großen EKK Schlüssel überdenken und von einer Gleichverteilung ausgehen, kommen wir zum einen auf 2^{40} mögliche Kombinationen für den privaten Schlüssel und zum anderen auf eine Gesamtwahrscheinlichkeit von $(\frac{1}{2})^{40}$. Unter Hinzunahme des Geburtstagsparadoxon brauchen wir lediglich 2^{20} Kombinationen zu berücksichtigen und kommen auf eine Gesamtwahrscheinlichkeit $(\frac{1}{2})^{20}$. Die Wahrscheinlichkeit fällt zwar sehr gering aus, trotzdem ist es aber möglich auf den privaten Schlüssel in vertretbarer Rechenzeit zu schließen. Falls bis $5P$ vorberechnet worden ist, dann würden sich für unser obiges 4-NAF Beispiel die in der Tabelle 6.7 gelisteten Werte / Kombinationen ergeben.

Tabelle 6.7: Analyse mittels DPA III

	Fenster 1	Fenster 2
1.	1	-1
2.	1	-3
3.	1	-5
4.	3	-1
5.	3	-3
6.	3	-5
7.	5	-1
8.	5	-3
9.	5	-5

Das wären dann schon 3^2 mögliche Werte. Wenn wir diese Tatsache nun für einen 160bit großen Schlüssel hochrechnen, würden wir, aufgrund der 20 Fenster, auf

$$(3^2)^{20} = 3^{40}$$

Kombinationen kommen.

Unter Hinzunahme des Geburtstagsparadoxon

$$(\sqrt{3^{40}}) = 3^{20}$$

$$3^{20} < 2^{40}$$

könnten wir das Problem noch in vertretbarer Rechenzeit lösen bzw. auf den privaten Schlüssel schließen.

Bei einer Vorberechnung bis $7P$ bei 4-NAF wären wir bei

$$(4^2)^{20} = 4^{40}$$

Kombinationen.

Mit dem Geburtstagsparadoxon

$$(\sqrt{4^{40}})$$

kommen wir auf

$$2^{40}$$

Kombinationen.

Hier ist eine Lösung in polynomieller Zeit noch nicht möglich. Die Betonung liegt hier auf "noch", da 2^{40} Kombinationen eine gerade noch angemessene Grenze der Sicherheit bilden.

Auch hier zeigt sich unser effizientes Vorgehen. Für unser 2-Fenster Beispiel mit 4-NAF kommen wir auf 64 Kombination. Die Tabelle 6.4 aus Szenario II zeigt alle potentiellen Möglichkeiten. Für einen 160bit großen EKK Schlüssel wären wir auch hier bei

$$(2^6)^{20} = 2^{120}$$

Kombinationsmöglichkeiten.

Wir betrachten aber im oben beschriebenen Idealfall für $3P$ nur 2^{20} Möglichkeiten. Das sind auch hier lediglich $7,88 \times 10^{-29}$ Prozent zu untersuchende Schlüssel aus 2^{120} möglichen.

6.2 Ergebnis

Wir haben drei Szenarien gesehen, die uns gezeigt haben, dass aufgrund bestimmter Gegebenheiten, ein Angriff auf effiziente Verschlüsselungsalgorithmen Erfolg haben kann. Dabei entsprechen die Gegebenheiten durchaus der Realität. Vor allen Dingen spielt die Effizienz für SmartCards eine wichtige Rolle. Diese, wie schon erwähnt, sind in ihren physischen Eigenschaften und in ihrer Performanz eingeschränkt. Hier bietet es sich an effiziente Algorithmen zu verwenden. Infolge der Effizienz muss auch auf Sicherheitsmechanismen, beispielsweise eine Randomisierungsfunktion, verzichtet werden. Hier setzt unser Angreifer an. Er bedient sich verschiedener Techniken und besitzt gute bis sehr gute Kenntnisse über die verwendete Hardware und über die verwendeten Algorithmen. Diese Eigenschaften eines Angreifers sind äußerst realitätsnah und beschreiben nicht den omnipotenten Angreifer.

Wir müssen uns dieser Schwachstellen bewusst werden und eine gewisse Awareness entwickeln. Der nächste Schritt wäre geeignete Gegenmaßnahmen zu den oben

gezeigten Szenarien zu finden.

Kapitel 7

Resümee

Wir haben gesehen, dass auch sorgfältig implementierte Rekodierungsalgorithmen, bei auf elliptische Kurven basierten Kryptosystemen, anfällig für Stromverbrauchsanalysen sind. Wir haben charakteristische Eigenschaften und Schemen von Rekodierungsalgorithmen kennengelernt und gesehen wie diese ausgenutzt werden können - sowohl für den reinen Angriff auf die Rekodierungsphase, als auch für die kombinierte Schwachstellenanalyse.

Während Effekte eines einzelnen, schaltenden Transistors normalerweise nur schwer durch direktes Monitoring des Stromverbrauchs zu identifizieren sind, können statistische Methoden zuverlässig außergewöhnliche geringe Stromdifferenzen feststellen. Deshalb nehmen wir an, dass auf einer SmartCard, auch trotz einer eventuellen Verschleierung durch mehrere gleichzeitig geschaltet Co-Prozessoren, kein wahrer Schutz gegen DPA erreicht werden kann.

Die nächste, sich an diese Arbeit anschließende Aufgabe wäre, die konkrete Umsetzung der beschriebenen Szenarien aus Kapitel 6 in einer Art Laborversuch. Bei zu erwartendem Erfolg müssten dann nachfolgend wirkungsvolle Gegenmaßnahmen entwickelt werden.

Abbildungsverzeichnis

2.1	$E: y^2 = x^3 - 5x + 7$	11
2.2	$E: y^2 = x^3 - 7x + 6$	11
2.3	Punktaddition: $P \neq Q$	14
2.4	Punktverdoppelung von P	14
2.5	Singuläre Kurve $E: y^2 = x^3 - 3x + 2$	16

Tabellenverzeichnis

4.1	Komplexitätsvergleich I	30
4.2	Komplexitätsvergleich II	36
5.1	Relation zwischen Fensterinhalt & \mathcal{SN} -Werten für 3-NAF	42
5.2	Versuchsergebnisse	48
6.1	Analyse mittels SPA I	54
6.2	Analyse mittels SPA II	54
6.3	Analyse mittels TA	56
6.4	Mögliche Kombinationen für 4-NAF	57
6.5	Analyse mittels DPA I	58
6.6	Analyse mittels DPA II	59
6.7	Analyse mittels DPA III	59

Literaturverzeichnis

- [AK97] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. *In Proc. of 1997 Security Protocols Workshop*, pages 125–136, Volume 1361 of Lecture Notes in Computer Science, 1997. Springer-Verlag.
- [Bec06] Anja Becker. Methods of fault analysis attacks on elliptic curve cryptosystems. *Comparison and Combination of Countermeasures to resist SCA*, pages 1–17, 2006. Darmstadt University of Technology, Diploma Thesis.
- [Ble05] Soeren Bleikertz. Das elliptische-kurven-kryptosystem. 2005. Ruhr-University Bochum.
- [Buc04] Professor Dr. Johannes Buchmann. Einfuehrung in die kryptographie. pages 95–96, 2004. Springer-Verlag.
- [CJ01] Clavier and M. Joye. Universal exponentiation algorithm a first step towards provable spa-resistance. *In Cryptographic Hardware and Embedded Systems - CHES 2001*, page 30030, Volume 2162 of Lecture Notes in Computer Science 2001. Springer-Verlag.
- [CKN00] J.-S. Coron, P. Kocher, and D. Naccache. Statistics and secret leakage. pages 157–162, Volume 1962 of Lecture Notes in Computer Science, 2000. Springer-Verlag.
- [CMO98] Henri Cohen, Atsuko Miyaji, and Takatoshi Ono. Efficient elliptic curve exponentiation using mixed coordinates. *In ASIACRYPT 98: Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security*, pages 51–65, London, UK, 1998. Springer-Verlag.
- [Cor99] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. *CHES'99, LNCS 1717*, pages 292–302, 1999. Springer-Verlag.
- [C.W04] C.Walter. Simple power analysis of unified code for ecc double and add. *In M. Joye and J.-J. Quisquater, editors, Cryptographic Hardware and Embedded Systems (CHES) 2004*, pages 191–204, Volume 3156 of Lecture Notes in Computer Science, 2004. Springer- Verlag.

- [DKPA⁺98] J.-F. Dhem, F. Koeune, Leroux P.-A., P. Mestre, J.-J. Quisquater, and J.-L. Willems. A practical implementation of the timing attack. pages 167–182, Volume 1820 of *Lecture Notes in Computer Science*, 1998. Springer-Verlag.
- [FR94] G. Frey and H. G. Rueck. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of computation*, 62, pages 865–874, 1994.
- [Hen03] Birgit Henapl. Zur effizienz von elliptische kurven kryptographie. pages 67–84, 2003.
- [IEE] The institute of electrical and electronics engineers, inc. (ieee) p1363. standard specification for public key cryptography, 2000. draft version d13, 1999. <http://cnscenter.future.co.kr/resource/crypto/standard/p1363/P1363-11-12-99-pdf.zip>.
- [IIT03] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. A practical countermeasure against address-bit differential power analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2003, Lecture Notes in Computer Science*, pages 382–396, 2003. Springer-Verlag.
- [JT01] Marc Joye and Christophe Tymen. Protections against differential analysis for elliptic curve cryptography. In *CHES 01: Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, pages 377–390, London, UK, 2001. Springer-Verlag.
- [Kob87] N. Koblitz. Elliptic curve cryptosystems. In *Mathematics of Computation*, pages 203–209, 1987. Volume 48.
- [Koc96a] P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. *Advances in Cryptology - CRYPTO'96, LNCS 1109*, pages 104–113, 1996. Springer-Verlag.
- [Koc96b] P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO96*, pages 104–113, 1996. Springer-Verlag.
- [KR02] V. Klima and T. Rosa. Further results and considerations on side channel attacks on rsa. In *Cryptographic Hardware and Embedded Systems (CHES) 02*, pages 244–259, Volume 2523 of *Lecture Notes in Computer Science* 2002. Springer-Verlag.
- [KSWH98] J. Kelsey, B. Schneier, D. Wagner, and C. Hall. Side channel cryptanalysis of product ciphers. In *ESORICS 98*, pages 97–110, 1998. Springer-Verlag.
- [KuBJ] P. C. Kocher and J. Jaffe und B. Jun. Introduction to differential power analysis and related attacks. Cryptography Research, <http://www.cryptography.com>.

- [KuBJ99] P. C. Kocher and J. Jaffe und B. Jun. Differential power analysis. *Advances in Cryptology - CRYPTO'99, LNCS 1666*, pages 388–397, 1999. Springer-Verlag.
- [LL94] C. H. Lim and P. J Lee. More flexible exponentiation with precomputation. In *Yvo G. Desmedt, editor, Advances in Cryptology CRYPTO 94*, pages 108–113, Volume 839 of Lecture Notes in Computer Science, August 1994. Springer-Verlag.
- [MDS99] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Power analysis attacks of modular exponentiation in smartcards. In *CHES 99: Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems*, pages 144–157, London, UK, 1999. Springer-Verlag.
- [Mil85] V. Miller. Use of elliptic curves in cryptography. *CRYPTO: Proceedings of Crypto, 85*, 1985.
- [MMM03] Ciaran McIvor, Maire McLoone, and John V McCanny. Fast montgomery modular multiplication and rsa cryptographic processor architectures. *The Institute of Electronics, Communications and Information Technology. School of Electrical and Electronic Engineering*, 2003. Ireland.
- [MOC97] A. Miyaji, T. Ono, and H. Cohen. Efficient elliptic curve exponentiation. In *S. Quing Y. Han, T. Okamoto, editor, International Conference on Information and Communications Security - ICIS 97*, 1334 of Lecture Notes in Computer Science:282–290, 1997.
- [Moe01] Bodo Moeller. Algorithms for multi-exponentiation. In *Serge Vaudenay and Amr M. Youssef, editors, Selected Areas in Cryptography - SAC 2001*, pages 165–180, 2001. Springer-Verlag.
- [Moe02] B. Moeller. Improved techniques for fast exponentiation. *ICSIC 2002, LNCS 2587*, pages 298–312, 2002. Springer-Verlag.
- [MOV91] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 80–89, 1991.
- [MvOV97] A. J. Menezes, P. C. van Oorschot, , and S. A. Vanstone. Handbook of applied cryptography. *CRC Press, Boca Raton*, page 616, 1997. Florida.
- [OT03a] K. Okeya and T. Takagi. A more flexible countermeasure against side channel attacks using window method. *CHES 2003, LNCS 2779*, pages 397–410, 2003. Springer-Verlag.
- [OT03b] K. Okeya and T. Takagi. The width-w naf method provides small memory and fast elliptic scalar multiplications secure against side channel attacks. *CT-RSA 2003, LNCS 2612*, pages 328–342, 2003. Springer-Verlag.

- [RE02] Wolfgang Rankl and Wolfgang Effing. Handbuch der chipkarten. 4. Auflage, 2002. Hanser Verlag.
- [Rei60] G. W. Reitwiesner. Binary arithmetic. *Advances in Computers*, pages 231–308, Volume 1, 1960.
- [SA98] T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Commentarii Mathematici Universitatis Sancti Pauli*, 47, pages 81–92, 1998.
- [Sil86] Joseph H. Silverman. The arithmetic of elliptic curves. page 130, Volume 106 of Graduate Texts in Mathematics, 1986. Springer-Verlag.
- [Sma99] N. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12, pages 193–196, 1999.
- [Sol00] Jerome A. Solinas. Efficient arithmetic on koblitz curves. *j- DESIGNS-CODES-CRYPTOGR*, 19(23):195–249, March 2000.
- [SS04] Yasuyuki Sakai and Kouichi Sakurai. A new attack with side channel leakage during exponent recoding computations. In *M. Joye and J.-J. Quisquater (Eds.): CHES 2004, LNCS 3156, International Association for Cryptologic Research 2004*, pages 298–311, 2004. Mitsubishi Electric Corporation, 5-1-1 Ofuna, Kamakura, Kanagawa 247-8501, Japan. Kyushu University, 6-10-1 Hakozaki, Higashi-Ku, Fukuoka 812-8581, Japan.
- [Str64] E. Straus. Problems and solutions: Addition chains of vectors. *American Mathematical Monthly*, 71, pages 806–808, 1964.
- [Vau04] S. Vaudenay. Digital signature schemes with domain parameters: Yet another parameter issue in ecdsa. In *H. Wang, J. Pieprzyk, and V. Varadharajan, editors, Australasian Conference on Information Security and Privacy, ACISP 2004*, pages 188–199, Volume 3108 of Lecture Notes in Computer Science, 2004. Springer-Verlag.
- [X9.99] ANSI X9.63. *Public Key Cryptography For The Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography*, January 1999. <http://grouper.ieee.org/groups/1363/private/x9-63-01-08-99.pdf>.
- [YLL94] S.-M. Yen, C.-S. Laih, and A. Lenstra. Multi-exponentiation. In *IEE Proceedings - Computers and Digital Techniques*, pages 325–326, Volume 141, 1994.