In: Horster, P. (Hrsg.): *Digitale Signaturen*. Proceedings der Arbeitskonferenz Digitale Signaturen 96, GMD Darmstadt, 18.-19.09.96, Vieweg-Verlag Braunschweig/Wiesbaden 1996, S. 201-220.

Effiziente Digitale Signatursysteme auf der Basis Elliptischer Kurven

Dirk Fox[1], Alexander W. Röhm[2]

[1]Universität - GH Siegen
Institut für Nachrichtenübermittlung
fox@nue.et-inf.uni-siegen.de

[2]Universität - GH Essen
Wirtschaftsinformatik
roehm@wi-inf.uni-essen.de

Zusammenfassung

Derzeit beruht die Sicherheit der meisten implementierten Digitalen Signatursysteme auf der Komplexität des *Faktorisierungsproblems* oder der des *Diskreten Logarithmusproblems* (DLP) in *GF*(*p*). Für beide zahlentheoretischen Probleme sind heute Algorithmen mit subexponentiellem Aufwand bekannt. Daher wird in den letzten Jahren asymmetrischen Kryptosystemen auf Elliptischen Kurven über endlichen Körpern besondere Aufmerksamkeit gewidmet: Zur Lösung des DLP auf (bestimmten) Elliptischen Kurven sind bisher nur Algorithmen mit exponentiellem Aufwand bekannt.

Der Beitrag stellt unterschiedliche Optimierungsansätze für die Realisierung Digitaler Signatursysteme auf der Basis Elliptischer Kurven in Software vor und schließt mit einem Aufwandsvergleich mit herkömmlichen Digitalen Signatursystemen anhand von Aufwandsabschätzungen und Messungen mit eigenen Implementierungen.

1 Einleitung

Seit den Veröffentlichungen von Miller und Koblitz Mitte der 80er Jahre arbeiten Kryptographen an der Untersuchung und Implementierung asymmetrischer Kryptosysteme auf der Basis Elliptischer Kurven [26, 15]. Für Elliptische Kurven über endlichen Körpern läßt sich analog Galois-Feldern ein *Diskretes Logarithmusproblem* (DLP) definieren, für das – im Unterschied zum DLP über Galois-Feldern – bis heute kein allgemein anwendbarer Lösungsalgorithmus mit subexponentiellem Aufwand bekannt ist. Definiert man nun Digitale Signatursysteme auf der Basis geeigneter Elliptischer Kurven, so können diese bei vergleichbarem Sicherheitsniveau mit erheblich kürzeren Schlüsseln arbeiten als beispielsweise das ElGamal-Verfahren oder der *Digital Signature Algorithm* (DSA) [7, 28].

Geringere Schlüssellängen führen zudem zu kürzeren Signaturen und Zertifikaten. Sie reduzieren so den Speicherbedarf und die erforderliche Übertragungsbandbreite in einer Sicherheitsinfrastruktur. Außerdem ermöglichen sie schnellere arithmetische Operationen und erhöhen damit die Geschwindigkeit der Prüf- und Signieralgorithmen.

Dieser Vorteil kommt besonders bei der weiteren Zunahme von Rechenleistung zum Tragen: Da der Aufwand exponentieller Algorithmen wesentlich schneller steigt als der subexponentieller Algorithmen, führt bei Digitalen Signatursystemen über Elliptischen Kurven eine Verlängerung des Schlüssels um wenige Bits zu einer signifikanten Vervielfachung des Aufwands für einen Angreifer. Herkömmliche Digitale Signatursysteme über Galois-Feldern müssen die Schlüssellängen um deutlich mehr Bits vergrößern, um denselben Grad an Sicherheit zu erreichen.

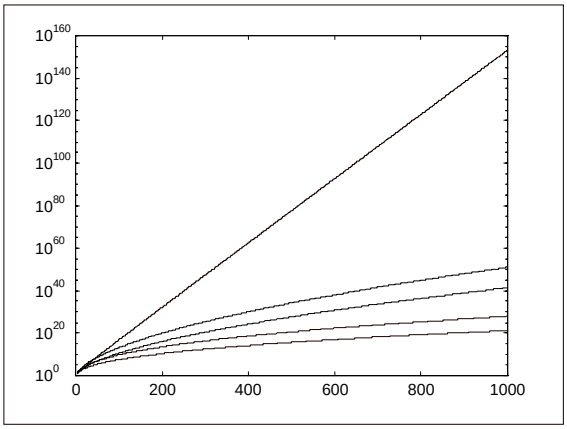

Bild 1-1: Asymptotischer Aufwand zur Bestimmung des DLP

Bild 1-1 zeigt einen Vergleich des asymptotischen Aufwands der heute bekannten Verfahren zur Bestimmung Diskreter Logarithmen.[1] Ein Aufwand von 10^{40} gilt derzeit als nicht zu bewältigen.

Eine weitere Besonderheit der Realisierung Digitaler Signatursysteme auf Elliptischen Kurven ist die Erweiterung des Schlüsselraums: Neben den Parametern des Signatursystems können auch die Kurvenparameter in die Schlüsselgenerierung einbezogen werden. Da zu jedem endlichen Körper eine Vielzahl Elliptischer Kurven existiert, kann dabei das Galois-Feld (und damit die Implementierung der Arithmetik des endlichen Feldes) konstant bleiben.

Dieser Beitrag skizziert zunächst zusammenfassend einige wesentliche Grundkenntnisse über die für kryptographische Anwendungen interessanten Elliptischen Kurven. Anschließend werden unterschiedliche Ansätze zur Optimierung der Implementierung Digitaler Signatursysteme auf Elliptischen Kurven vorgestellt und anhand von Aufwandsabschätzungen und Meßergebnissen eigener Implementierungen bewertet. Schließlich folgt ein Effizienzvergleich der unterschiedlichen Vorschläge zur Realisierung Digitaler Signatursysteme auf Elliptischen Kurven mit „klassischen“ Digitalen Signatursystemen über endlichen Körpern (ElGamal, DSS).

---

1 Die beiden unteren Kurven zeigen den Aufwand der schnellsten probabilistischen Verfahren.

2 Elliptische Kurven über endlichen Körpern

Eine Elliptische Kurve ist definiert als die Menge aller Punkte $P = (x, y)$ des affinen zweidimensionalen Raumes, die die *allgemeine Weierstraß-Gleichung* (*Elliptische Kurvengleichung*)

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (2.1)$$

erfüllen, zusammen mit einem *Punkt im Unendlichen* $\underline{0}$. Dabei sind sowohl die Koeffizienten a_i als auch x und y Elemente (des algebraischen Abschlusses) eines Körpers K. Man spricht daher auch von einer Elliptischen Kurve $E(K)$ über dem Körper K. Auf einer Elliptischen Kurve läßt sich eine Punkte-Addition definieren. Bezüglich dieser Operation bildet die Menge der Lösungspunkte eine abelsche Gruppe.[2] Graphisch veranschaulichen läßt sich diese Operation für Elliptische Kurven über R (Abb. 2-1):

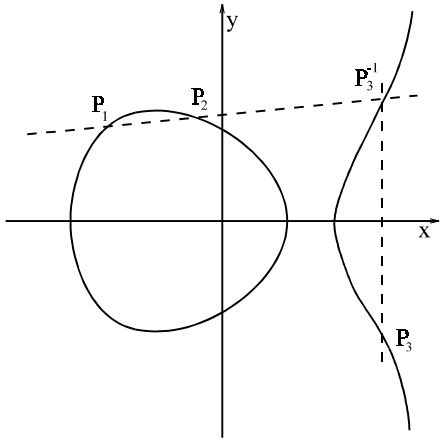

Abb. 2-1: Konstruktion der Punkte-Addition auf Elliptischen Kurven über R

Der Summenpunkt $P_3 = P_1 + P_2$ kann leicht durch die Berechnung des Steigungsdreiecks bestimmt werden [35, 22]:

$$\begin{aligned} x_3 &= \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 \\ y_3 &= -(\lambda + a_1)x_3 - \nu - a_3 \end{aligned} \quad \text{mit} \quad \lambda := \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{falls } P_1 \neq P_2 \\ \dfrac{3x_1^{\,2} + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}, & \text{falls } P_1 = P_2 \end{cases}$$

$$\nu := \begin{cases} \dfrac{x_2y_1 - x_1y_2}{x_2 - x_1}, & \text{falls } P_1 \neq P_2 \\ \dfrac{-x_1^{\,3} + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3}, & \text{falls } P_1 = P_2 \end{cases} \qquad (2.2)$$

---

[2] Die Abschlußeigenschaft läßt sich gut in der dreidimensionalen projektiven Ebene zeigen, in der auch der Punkt im Unendlichen eine explizite Darstellung besitzt (Abschnitt 3.4.1): Nach dem Satz von Bezout schneidet jede beliebige projektive Gerade die Elliptische Kurve E in genau 3 Punkten.

Die additive Inverse eines Punktes $P = (x, y)$ ist $-P = (x, -y - a_1 x - a_3)$. Der Punkt im Unendlichen $\underline{0}$ ist das *Neutrale Element* dieser Additionsoperation. Damit gilt:

$$-\underline{0} = \underline{0},\ \underline{0} + P = P \text{ und } P - P = \underline{0}$$

Zu dieser Additionsoperation läßt sich eine Abbildung $k \cdot P$ als k-fache Addition eines Punktes P der Kurve definieren:

$$k \cdot P := \begin{cases} \underbrace{P + \ldots + P}_{k\text{-mal}}, & \text{für } k > 0 \\ \underline{0}, & \text{für } k = 0, \\ -k \cdot (-P), & \text{für } k < 0 \end{cases} \qquad (2.3)$$

Für die Realisierung Digitaler Signatursysteme sind endliche Körper (*Galois-Felder* $GF(q)$) als Basis Elliptischer Kurven von besonderem Interesse. Bezüglich solcher Kurven $E(GF(q))$ wird die (endliche) *Ordnung* eines Punktes P ($Ord(P)$) definiert als der kleinste Wert n, für den gilt: $n \cdot P = \underline{0}$. Der Kurvenpunkt P erzeugt damit eine multiplikative (zyklische) Untergruppe von E der Mächtigkeit $Ord(P)$. Die Ordnung von P kann leicht als Teiler der Ordnung $\#E$ der Elliptischen Kurve, d.h. der Zahl der Lösungspunkte der Kurvengleichung (2.1) bestimmt werden.

Die exakte Bestimmung der Kurvenordnung für Elliptische Kurven ist im allgemeinen allerdings schwierig; der Aufwand des schnellsten bekannten allgemeinen Verfahrens, dem Algorithmus von Schoof, ist mit $O(\log^8 q)$ erheblich [34, 25].[3] Das *Theorem von Hasse* liefert jedoch eine Schranke für die Größe dieses Wertes:

$$q + 1 - 2\sqrt{q} \le \#E \le q + 1 + 2\sqrt{q} \qquad (2.4)$$

Die Ordnung $\#E$ einer Elliptischen Kurve hat also dieselbe Größenordnung wie die des endlichen Körpers $GF(q)$, über dem sie definiert ist.[4]

Mit der Abbildung (2.3) kann auf zyklischen Untergruppen von E analog Galois-Feldern $GF(p)$ ein *Diskretes Logarithmusproblem* (DLP) definiert werden:

> Sind $P, Q \in E$ Generatoren einer zyklischen Untergruppe von E, dann heißt $k \in GF(Ord(P))$ mit $Q = k \cdot P$ *Diskreter Logarithmus* von Q bezüglich P.

Für die Bestimmung Diskreter Logarithmen auf Elliptischen Kurven ist bis heute nur das allgemeine Baby-Step-Giant-Step-Verfahren von Shanks bekannt [29], dessen asymptotischer Aufwand

$$O(\sqrt{Ord(P)} \cdot \log \sqrt{Ord(P)}) \qquad (2.5)$$

Punktadditionen erfordert. Verfahren zur Lösung des DLP in $GF(q)$ wie der Index-Calculus-Algorithmus, dessen asymptotischer Aufwand subexponentiell ist und in vergleichbarer

---

[3] Inzwischen wurden verbesserte Verfahren zur Bestimmung der Kurvenordnung [18, 20] bzw. zur Wahl der Kurve zu einer leicht bestimmbaren Ordnung [33, 17] vorgeschlagen.

[4] Spezielle Klassen Elliptischer Kurven, deren Ordnung in einer geschlossenen Formel angegeben werden kann, werden in Abschnitt 3.4.1 betrachtet.

Größenordnung liegt wie der bekannter Faktorisierungsalgorithmen [22], sind nicht auf das DLP in E übertragbar [26]. Die Lösung des DLP in $E(GF(q))$ ist daher derzeit so aufwendig wie die Lösung des DLP in $GF(q^*)$ für $q^* \gg q$ (siehe Bild 1-1).

Diese relative Komplexität des DLP in $E(GF(q))$ läßt sich für einzelne Kurven präziser bestimmen. Von Menezes, Okamoto und Vanstone wurde 1993 ein Reduktionsalgorithmus vorgestellt, der das DLP in $E(GF(q))$ auf das DLP im Erweiterungskörper $GF(q^r)$ reduziert und dort mit dem Index-Calculus-Verfahren löst [23]. Damit ist das DLP in $E(GF(p))$ für $r < \log q$ subexponentiell [2]. Durch geeignete Wahl von q und $Ord(P)$ kann jedoch ein ausreichend großer Wert für den Reduktionskoeffizienten r erzwungen werden. Dazu ist folgende Bedingung für den größten Primteiler p von $\#E(GF(q))$ zu prüfen:

$$\forall s < r\colon\ p \nmid q^s - 1 \tag{2.6}$$

Die asymptotisch größere Komplexität des DLP auf Elliptischen Kurven erlaubt es, Digitale Signatursysteme auf Elliptischen Kurven (mit genügend großem Reduktionskoeffizienten r) zu realisieren,[5] die bei gleichem Sicherheitslevel mit wesentlich kleineren Schlüssellängen als Digitale Signatursysteme über $GF(q)$ auskommen (siehe Kapitel 4).

Je nach *Charakteristik* des endlichen Körpers $char(GF(q))$, über dem die Elliptische Kurve definiert ist, kann die allgemeine Weierstraß-Gleichung (und damit auch die Punktberechnung) durch Koordinatentransformation vereinfacht werden. Für kryptographische Anwendungen sind insbesondere die endlichen Körper $GF(p)$ (mit Primzahl $p > 3$) und $GF(2^n)$ (mit $n > 0$) von Bedeutung.

2.1 Elliptische Kurven über $GF(p)$

Die Kurvengleichung für Elliptische Kurven über $GF(p)$ (mit p Primzahl größer 3) lautet:

$$y^2 = x^3 + ax + b \tag{2.7}$$

Die Parameter a, b sind dabei so zu wählen, daß die Kurve *nicht-singulär* ist, d.h. die partiellen Ableitungen aller Punkte der Kurve ungleich Null sind.[6] Diese Bedingung ist äquivalent der Forderung, daß Gleichung (2.7) keine Vielfachwurzeln besitzt, daß also gilt:

$$4a^3 + 27b^2 \neq 0 \tag{2.8}$$

Eindeutig bestimmt – bis auf Isomorphie – wird eine Elliptische Kurve über $GF(p)$ durch ihre *j-Invariante*

$$j(E(GF(p)) = -\frac{12^3 (4a)^3}{16 \cdot (4a^3 + 27b^2)} \tag{2.9}$$

Mit der Kurvengleichung (2.7) vereinfacht sich die Punktaddition (2.2) wie folgt:

---

5 In [11] wird $r = 10$ empfohlen.

6 Singuläre Elliptische Kurven $E(GF(q))$ sind isomorph der multiplikativen Gruppe des Galois-Feldes $GF(q)$.

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned}, \text{ mit } \lambda := \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{falls } P_1 \neq P_2 \\ \dfrac{3x_1^2 + a}{2y_1}, & \text{falls } P_1 = P_2 \end{cases} \qquad (2.10)$$

Die additive Inverse erhält man (anschaulich ausgedrückt) durch „Spiegelung an der x-Achse“:

$$-P = -(x, y) = (x, -y)$$

Die Addition zweier (verschiedener und nicht-inverser) Punkte auf E erfordert demnach eine Multiplikation, eine Quadrierung und eine Inversenbestimmung. Die Verdoppelung eines Punktes setzt sich aus zwei Multiplikationen, zwei Quadrierungen und einer Inversenbestimmung in $GF(p)$ zusammen.

2.2 Elliptische Kurven über $GF(2^n)$

Bei der Darstellung Elliptischer Kurven über endlichen Körpern der Charakteristik 2 ($GF(2^n)$) werden zwei Fälle unterschieden: Kurven mit j-Invariante ≠ 0 und solche mit j-Invariante = 0. Letztere werden in Abschnitt 3.4.1 genauer betrachtet, da sie besondere Eigenschaften besitzen.

Für Kurven über endlichen Körpern der Charakteristik 2 lautet die vereinfachte Kurvengleichung (2.1) mit $j(E) = 1/a_6$ ($\neq 0$):

$$y^2 + xy = x^3 + a_2 x^2 + a_6 \qquad (2.11)$$

Die Kurve ist nicht-singulär genau dann, wenn $a_6 \neq 0$. Mit (2.11) vereinfacht sich die Punktaddition (2.2) zu:

$$\begin{aligned} x_3 &= \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)^2 + \dfrac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a_2, & \text{falls } P_1 \neq P_2 \\ x_1^2 + \dfrac{a_6}{x_1^2}, & \text{falls } P_1 = P_2 \end{cases} \\ y_3 &= \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1, & \text{falls } P_1 \neq P_2 \\ x_1^2 + \left(x_1 + \dfrac{y_1}{x_1}\right)x_3 + x_3, & \text{falls } P_1 = P_2 \end{cases} \end{aligned} \qquad (2.12)$$

Die additive Inverse eines Punktes $P = (x, y)$ ist $-P = (x, y + x)$. Die Addition zweier (verschiedener und nicht-inverser) Punkte auf $E(GF(2^n))$ erfordert demnach zwei Multiplikationen, eine Quadrierung und eine Inversenbestimmung. Die Verdoppelung eines Punktes benötigt drei Multiplikationen (falls $a_6 \neq 1$), zwei Quadrierungen und eine Inversenbestimmung in $GF(2^n)$.

3 Effiziente Implementierung

Wie im vorausgegangenen Abschnitt gezeigt, setzt sich die Punktaddition auf E aus mehreren Rechenoperationen im endlichen Körper $GF(q)$ zusammen, über dem die Elliptische Kurve definiert ist. Die k-fache Punktaddition auf E ist damit prinzipiell aufwendiger als eine k-fache Multiplikation (Exponentiation) auf $GF(q)$. Bei der Spezifikation Digitaler Signatursysteme auf Elliptischen Kurven entstehen dennoch Effizienzgewinne gegenüber herkömmlichen Verfahren durch die Möglichkeit, mit wesentlich kürzeren Schlüsseln zu arbeiten. Für die Realisierung asymmetrischer Kryptosysteme auf Elliptischen Kurven werden Moduluslängen von 128 bis 160 bit empfohlen (gegenüber 660 bis 1024 bit in $GF(p)$) [10].[7]

Durch geeignete Implementierung der Verfahren kann der Aufwandsvergleich weiter zugunsten Elliptischer Kurven verbessert werden. Optimierungen können dabei an vier Stellen ansetzen: der Körperarithmetik in $GF(q)$, der Wahl der Elliptischen Kurve, der Darstellung der Kurvenpunkte und dem Algorithmus zur k-fachen Punktaddition auf der Elliptischen Kurve.

Im folgenden werden unterschiedliche Optimierungsansätze vorgestellt. Sie werden durch Aufwandsabschätzungen und vergleichende Messungen für typische Moduluslängen ergänzt, die mit einer C-Implementierung auf PCs (Pentium, 90 MHz) vorgenommen wurden.

3.1 Optimierung der Körperarithmetik in $GF(q)$

Multiplikationen und Quadrierungen in endlichen Körpern $GF(q)$ sind die zentralen arithmetischen Operationen der Implementierung Digitaler Signatursysteme auf zyklischen Gruppen. Das gilt sowohl für Systeme über Elliptischen Kurven als auch für solche über $GF(q)$. Optimierungen dieser Operationen kommen demnach prinzipiell beiden Verfahren zugute. Allerdings greifen einzelne Maßnahmen besonders bei vergleichsweise kleinen Moduluslängen, wie denen endlicher Körper für Elliptische Kurven.

Maß der folgenden Aufwandsbetrachtungen ist eine Wort-Operation. Dabei bezeichnet $n = |q|$ die Länge von q in bit und s die Stellenzahl (Wortanzahl) von q.

3.1.1 Körperarithmetik in $GF(p)$

Elementaroperation aller (wesentlichen) arithmetischen Operationen einer Körperarithmetik in $GF(p)$ ist eine Wort-Multiplikation; sie wird im folgenden als Aufwandsmaß verwendet.

Eine zentrale arithmetische Operation in $GF(p)$ ist die Reduktion der Additions-, Multiplikations- und Quadrierungsergebnisse bezüglich p. Dazu wird meist eine Langzahl-Division verwendet, deren Aufwand bei

$$s \cdot (s + 3{,}5) \tag{3.1}$$

liegt. Eine Division erfordert damit $3{,}5 \cdot s$ Wort-Multiplikationen mehr als eine Multiplikation (Aufwand: s^2) [14, 8, 6]. Von Montgomery stammt ein Vorschlag, diesen Aufwand durch Verwendung einer anderen Zahlenrepräsentation zu verringern [27]. Die Umrechnung der

---

[7] Moduluslängen von weniger als 128 bit verbieten sich wegen der Gefahr eines Birthday-Angriffs für Digitale Signatursysteme ohnehin.

Darstellung erfordert jeweils eine Division. In Montgomery-Darstellung ist statt einer Langzahl-Division lediglich eine spezielle Montgomery-Reduktion mit

$$s \cdot (s+1) \tag{3.2}$$

Wort-Multiplikationen erforderlich. Sie ist damit nur s Wort-Multiplikationen aufwendiger als eine Multiplikation. Die Verwendung der Montgomery-Darstellung führt besonders bei vergleichsweise kurzen Moduluslängen zu einer Beschleunigung, da hier der Mehraufwand einer Division verglichen mit dem einer Multiplikation, d.h. 2,5/s, besonders groß ausfällt [8] (Tab. 3-1).

Operation in Z	n = 128 bit	n = 160 bit
Division [14]	0,14 ms	0,15 ms
Reduktion [27]	0,09 ms	0,10 ms
Multiplikation	0,04 ms	0,04 ms

Tab. 3-1: Aufwandsvergleich Division, Montgomery-Reduktion und Multiplikation

Effiziente Algorithmen für die Multiplikation langer Zahlen werden schon länger intensiv untersucht. Für Softwareimplementierungen kommen insbesondere die Verfahren von Karatsuba [14] und Schönhage-Strassen [1] in Frage. Zwar haben beide Algorithmen asymptotisch mit $O(n^{1.585})$ und $O(n \cdot \log n \cdot (\log (\log n)))$ einen geringeren Aufwand als die klassische „stellenweise“ Multiplikation mit $O(n^2)$. Für die hier betrachteten kleinen Moduluslängen sind allerdings beide Algorithmen aufwendiger [8].

Zusammen mit der anschließenden Reduktion modulo p (Division) hat eine (modulare) Multiplikation in $GF(p)$ damit einen Gesamtaufwand von $2s^2 + c$ Wort-Multiplikationen; c hängt dabei vom verwendeten Reduktionsverfahren ab.

Eine Quadrierung benötigt lediglich die Hälfte der Elementaroperationen einer Multiplikation, bei „klassischer“ (stellenweiser) Multiplikation also

$$s^2/2 \tag{3.3}$$

Wort-Multiplikationen. Damit kann der Aufwand einer (modularen) Quadrierung in $GF(p)$ auf bis zu ¾ des Aufwandes einer (modularen) Multiplikation verringert werden; praktisch erreicht man diesen Wert für kleine Moduluslängen allerdings nicht (Tab. 3-2).

Operation in $GF(p)$	$\|p\|$ = 128 bit	$\|p\|$ = 160 bit
mod. Multiplikation	0,18 ms	0,21 ms
mod. Quadrierung	0,17 ms	0,19 ms

Tab. 3-2: Aufwand von (modularer) Multiplikation und Quadrierung in $GF(p)$

Eine wichtige Rolle spielt die Geschwindigkeit der Berechnung einer multiplikativen Inversen, die sowohl bei der Punktaddition und -verdoppelung in $E(GF(p))$ als auch bei DLP-basierten Digitalen Signatursystemen zu bestimmen ist. Dies leistet in $GF(p)$ der *Erweiterte Euklidsche Algorithmus* (EEA). Der durchschnittliche Aufwand des EEA läßt sich recht genau abschätzen [14]:

$$\lceil (12 \ln 2)/\pi^2 \cdot \ln p + 1{,}47 \rceil \qquad (3.4)$$

modulare Multiplikationen in $GF(p)$. Bei Moduluslängen von 128 und 160 bit erfordert die Bestimmung einer Inversen danach im Mittel etwa 74 bzw. 95 Körpermultiplikationen.

Operation in $GF(p)$	$\|p\|$ = 128 bit	$\|p\|$ = 160 bit
Inversenbestimmung	18,2 ms	22,9 ms
Anzahl Multiplikationen	~74	~95

Tab. 3-3: Aufwand der Inversenbestimmung in $GF(p)$

Wie die Messungen in Tabelle 3-2 und 3-3 zeigen, dominiert die Inversenbestimmung den Aufwand einer Punktaddition in $E(GF(p))$ bei weitem.

3.1.2 Körperarithmetik in $GF(2^n)$

In vielen Arbeiten wurde die Eignung spezieller Elliptischer Kurven über $GF(2^n)$ für Hardwareimplementierungen untersucht. Mit einem 40 MHz-Chip-Prototypen für eine Arithmetik in $E(GF(2^{155}))$ wurde dabei eine Verschlüsselungsrate von 60 kbit/s [2], mit einem 25 MHz-Chip für $E(GF(2^{161}))$ sogar ein Durchsatz von 80 kbit/s erreicht [33].[8]

Eine Arithmetik über $GF(2^n)$ besitzt als Softwareimplementierung nicht so große Vorteile gegenüber einer $GF(p)$-Arithmetik wie im Falle einer Hardwareimplementierung. Viele der möglichen Optimierungen wirken jedoch auch bei einer Softwareimplementierung. Elementaroperation und damit Maß der folgenden Aufwandsabschätzungen ist dabei die Rotation eines Wortes um ein oder mehrere Bits.

Stellt man die Elemente in $GF(2^n)$ bezüglich einer *Normalbasis* (NB) dar, kann die Quadrierung in $GF(2^n)$ auf eine zyklische Bitverschiebung reduziert werden [22].[9] Damit liegt der Aufwand bei s Wort-Rotationen und ist daher – im Vergleich zu einer Multiplikation – praktisch vernachlässigbar. Normalbasen existieren zu jedem endlichen Erweiterungskörper; die Wahl einer NB stellt daher keine Einschränkung dar.

Wählt man zudem einen endlichen Körper, für den eine *Optimale Normalbasis* (ONB) existiert, läßt sich auch der Aufwand einer Multiplikation in $GF(2^n)$ erheblich verringern. Mit

---

8 Zum Vergleich: Die schnellste uns bekannte Hardware-Realisierungen von RSA erreicht bei 25 MHz und einer Schlüssellänge von 1024 bit einen Verschlüsselungsdurchsatz von 50 kbit/s [33].

9 Es gilt sogar allgemeiner, daß die Abbildung $x \mapsto x^p$ in $GF(p^k)$ bezüglich einer Normalbasis einer Rechtsrotation um ein bit entspricht [13].

einigen algorithmischen Verbesserungen sinkt der Aufwand einer Multiplikation in $GF(2^n)$ auf

$$2s\cdot(n-1)+s \tag{3.5}$$

Wort-Rotationen [30]. Optimale Normalbasen existieren allerdings nur für etwa jedes vierte bis fünfte n und schränken damit die Kurvenwahl ein [11]. Der Aufwand einer Multiplikation in $GF(2^n)$ liegt dabei für $n \approx |p|$ näherungsweise um den Faktor n/s (Wortlänge) über dem einer (modularen) Multiplikation in $GF(p)$, sofern die Elementaroperationen Wort-Multiplikation und Wort-Rotation denselben Aufwand besitzten.

Die Berechnung multiplikativer Inversen in $GF(2^n)$ bezüglich einer Normalbasis erfordert erheblich weniger Multiplikationsoperationen als die Inversenbestimmung in $GF(p)$. Dies gelingt mit folgendem eleganten Algorithmus [12, 22]: Es ist leicht zu sehen, daß

$$a^{-1} = a^{2^n-2} = (a^2)^{2^{n-1}-1} \tag{3.6}$$

Zerlegt man nun den verbleibenden Exponenten iterativ nach folgender Vorschrift:

$$2^{n-1}-1 = (2^{(n-1)/2}-1)(2^{(n-1)/2}+1), \text{ für ungerade } n,$$
$$2^{n-1}-1 = 2(2^{(n-2)/2}-1)(2^{(n-2)/2}+1)+1, \text{ für gerade } n$$

reduziert sich der Aufwand einer Inversenbestimmung (bei Vernachlässigung der Quadrierungen) auf exakt

$$\lfloor \log_2(n-1) \rfloor + \omega(n-1) - 1 \tag{3.7}$$

Multiplikationen in $GF(2^n)$. Dabei bezeichnet $\omega(x)$ das *Hamminggewicht* von x, d.h. die Anzahl der von 0 verschiedenen Stellen der Binärdarstellung von x. Für die für Elliptische Kurven empfohlenen Ordnungen endlichen Körper $GF(2^n)$ erfordert damit eine Inversenbestimmung 8 ($n \approx 128$) bzw. 9 ($n \approx 160$) Körpermultiplikationen.

Tabelle 3-4 zeigt den Aufwand der arithmetischen Operationen in $GF(2^n)$ für $n = 130$ und $n = 162$ bezüglich einer optimalen Normalbasis im Vergleich. Um eine schnelle Arithmetik in $GF(2^n)$ zu erhalten, sollte n so gewählt werden, daß eine Optimale Normalbasis existiert und das Hamming-Gewicht von n klein ist.

Operation in $GF(2^n)$	$n = 130$	$n = 162$
Quadrierung	< 0,01 ms	< 0,01 ms
Multiplikation	6,55 ms	8,21 ms
Inversenbestimmung	52,9 ms	74,4 ms

Tab. 3-4: Multiplikation, Quadrierung und Inversenbestimmung in $GF(2^n)$

3.2 Optimierung der Punktdarstellung

Bisher wurden die Punkte einer Elliptischen Kurve in affinen Koordinaten dargestellt. Eine Elliptische Kurve kann jedoch ebenso als Punktemenge einer (zweidimensionalen) projektiven Ebene des dreidimensionalen Raumes betrachtet werden [24, 16, 2]. In dieser Darstellung sind die Punkte der Kurve Lösungspunkte der dreidimensionalen homogenen Weierstraß-Gleichung:

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3 \qquad (3.8)$$

Der Punkt im Unendlichen, für den es im affinen Raum keine explizite Darstellung gibt, hat hier die Koordinaten (0, 1, 0); es ist der einzige Punkt mit Z-Koordinate 0. Bezüglich der dreidimensionalen Kurvengleichung läßt sich analog dem zweidimensionalen Fall eine Punktaddition definieren.[10]

3.2.1 Homogene Koordinaten in $E(GF(p))$

Auf Elliptischen Kurven über $GF(p)$ liegt der Aufwand einer Punktaddition in homogenen Koordinaten bei 13 Multiplikationen und 2 Quadrierungen; der Aufwand einer Verdoppelung bei 6 Multiplikationen und 5 Quadrierungen in $GF(p)$. Damit fallen in beiden Operationen deutlich mehr Multiplikationen und Quadrierungen an als bei der Addition bzw. Verdoppelung in affinen Koordinaten. Dafür wird die – in $GF(p)$ sehr aufwendige – Inversenbestimmung (Abschnitt 3.1.1, (3.4)) eingespart.

Operation in $E(GF(p))$	$\|p\|$ = 128 bit		$\|p\|$ = 160 bit	
	affin	*homogen*	*affin*	*homogen*
Addition	18,6 ms	2,7 ms	23,3 ms	3,1 ms
Verdoppelung	18,9 ms	1,9 ms	23,7 ms	2,2 ms

Tab. 3-5: Aufwand von Punktaddition und -verdoppelung in $E(GF(p))$

Wie Tabelle 3-5 zeigt, sind für die Moduluslängen 128 und 160 bit Addition und Verdoppelung eines Punktes in homogenen Koordinaten erheblich effizienter als in affinen Koordinaten.

3.2.2 Homogene Koordinaten in $E(GF(2^n))$

Durch den Wechsel der Darstellung von affinen Koordinaten in die projektive Ebene kann auch in Elliptischen Kurven über $GF(2^n)$ bei der Punktaddition und -verdoppelung die Inversenbestimmung eingespart werden [24, 2, 22].

---

[10] Auf die Wiedergabe der Gleichungen zur Koordinatenbestimmung wird hier aus Platzgründen verzichtet; eine Darstellung für Elliptische Kurven über $GF(p)$ findet sich in [16], für $E(GF(2^n))$ in [22].

Der Effizienzgewinn ist hier allerdings, wie Tabelle 3-6 zeigt, wesentlich geringer als bei dem in Abschnitt 3.2.1 betrachteten Fall von Kurven über $GF(p)$: Der Aufwand einer Punktaddition liegt bei 13 Multiplikationen, die Verdoppelung wird mit 7 Multiplikationen in $GF(2^n)$ deutlich effizienter.

Operation in $E(GF(2^n))$	$n = 130$		$n = 162$	
	affin	*homogen*	*affin*	*homogen*
Addition	66 ms	85,2 ms	90,8 ms	106,7 ms
Verdoppelung	72,6 ms	45,9 ms	99 ms	57,5 ms

Tab. 3-6: Aufwand von Punktaddition und -verdoppelung in $E(GF(2^n))$

Mit der Darstellung eines Punktes in homogenen Koordinaten steigt jedoch der Speicheraufwand für Schlüssel und Signaturen eines Digitalen Signatursystems über Elliptischen Kurven. Um den Vorteil einer kompakten Darstellung nicht zu verlieren, sollte die Darstellung eines Punktes vor Beginn und nach Abschluß der Berechnungen auf der Elliptischen Kurve transformiert werden. Dazu läßt sich eine einfache Transformation T (mit Umkehrtransformation T^{-1}) definieren, die einen Punkt $P = (x, y)$ in die projektive Ebene abbildet:

$$T\colon (x,y) \mapsto (x,y,1) \quad \text{und} \quad T^{-1}\colon (x,y,z) \mapsto \left(\frac{x}{z},\frac{y}{z}\right) \qquad (3.9)$$

Die Transformation in die dreidimensionale projektive Ebene ist dabei „frei"; die Umkehrtransformation erfordert zwei Inversenbestimmungen. Da die Darstellung eines Punktes P in der projektiven Ebene des dreidimensionalen Raums nicht eindeutig ist, erfordert der Vergleich zweier Punkte in homogenen Koordinaten für $z_1 \neq z_2$ zusätzlich vier Körpermultiplikationen in $GF(q)$:

$$(x_1, y_1, z_1) = (x_2, y_2, z_2) \ \underline{gdw}.\ x_1 \cdot z_2 = x_2 \cdot z_1 \wedge y_1 \cdot z_2 = y_2 \cdot z_1 \qquad (3.10)$$

3.3 Optimierung der *k*-fachen Punktaddition

Einen weiteren Ansatzpunkt für Effizienzsteigerungen bieten Auswertungsstrategien bei der in Kapitel 2 definierten Abbildung einer k-fachen Punktaddition $k \cdot P$ auf einer Elliptischen Kurve $E(GF(q))$. Das Standard-Verfahren „Double&Add" wertet den Faktor k bitweise aus. Es erfordert $|k|$-1 Verdoppelungen eines Punktes und durchschnittlich $|k|/2$ Punktadditionen.

Dieser Aufwand läßt sich auf unterschiedliche Weise reduzieren. Ein Ansatz, der zur Optimierung der modularen Exponentiation vorgeschlagen wurde, setzt an der Darstellung von k an: Durch Zerlegung von k in eine Additionskette minimaler Länge kann die Zahl der Punktadditionen auf ein Minimum reduziert werden. Da das Finden von in diesem Sinne optimalen Additionsketten ein NP-vollständiges Problem ist [14], muß man sich bei genügend großem k auf suboptimale Lösungen beschränken, die allerdings (z.T. erheblichen) Vorausberechnungsaufwand erfordern [21, 32]. Diese Verfahren lohnen immer dann, wenn der Faktor k fester Parameter des Signatursystems ist, d.h. die Vorausberechnungen zur optimierten Darstellung einmalig, z.B. zusammen mit der Generierung von k vorgenommen werden können. Das gilt

für alle RSA-artigen Verfahren, bei denen der Exponent *k* fest bleibt, nicht aber für Signatursysteme, die auf dem DLP beruhen.

Eine Alternative sind Heuristiken, die minimale Additionsketten für Abschnitte von *k* bestimmen, den Wert *k* komprimieren, nach der Bitgruppen- und Fenstertechnik Tabellen mit Teilergebnissen vorausberechnen oder durch eine dreiwertige Darstellung von *k* die Zahl der erforderlichen modularen Multiplikationen bzw. Punktadditionen verringern [5, 36, 31, 4]. Mit diesen Verfahren läßt sich der durchschnittliche Rechenaufwand einer *k*-fachen Punktaddition analog einer modularen Exponentiation um bis zu 40% senken [9, 16].

Ein Vorschlag zur Reduktion der erforderlichen (modularen) Quadrierungen, der sich leicht auf die *k*-fache Punktaddition auf Elliptischen Kurven übertragen läßt, stammt von Lim und Lee [19]. Sie nutzen die bekannte Tatsache, daß bei einer „Doppelexponentiation" die Hälfte der Quadrierungen und Vorausberechnungen gespart werden kann, wenn beide Exponenten gleichzeitig (z.B. nach der Bitgruppentechnik) ausgewertet werden [9]:

$$x^a \cdot y^b = ((...(x^{a_1} \cdot y^{b_1})^{2^l} \cdot ...\cdot)^{2^l} \cdot x^{a_r} \cdot y^{b_r})^{2^l} \qquad (3.11)$$

mit $a = a_1 a_2 ... a_r$, $b = b_1 b_2 ... b_r$ und $|b_i| = |a_i| = l$. Dies kann durch Zerlegung von $k = k_1 + 2^{|k|/2} \cdot k_2$ genutzt werden: Mit Vorausberechnung des Punktes $P' := 2^{|k|/2} \cdot P$ vereinfacht sich damit die *k*-fache Punktaddition (2.3) zu

$$k_1 \cdot P + k_2 \cdot P' = ((...(P \cdot k_{11} + P' \cdot k_{21}) \cdot 2^l ...) \cdot 2^l + P \cdot k_{1r} + P' \cdot k_{2r}) \cdot 2^l \qquad (3.12)$$

Damit läßt sich die Hälfte der bei einer k-fachen Punktaddition erforderlichen Verdoppelungen einsparen (genauer: vorausberechnen). Durch die Bestimmung weiterer (Hilfs-)Punkte kann man die Zahl der erforderlichen Punktverdoppelungen weiter verringern. Die Tabellen 3-7 und 3-8 zeigen eine Gegenüberstellung des Aufwands einiger wichtiger Auswertungsverfahren für *k* und den Ansatz von Lim&Lee kombiniert mit der Bitgruppentechnik bezüglich der für Digitale Signatursysteme auf Elliptischen Kurven empfohlenen Längen.

$\|k\|$ = 128 bit		Vor.	Tabellen		'online'(max.)		Tabellen+'online' (max.)			Speicher	
Verfahren*	*l*	Dbl.	Dbl.	Add	Dbl.	Add	Dbl.	Add	Σ	fest	Tab.
Double&Add	1	-	-	-	127	63 (127)	127	63 (127)	210 (254)	-	1
Bitgruppen	4	-	7	7	124	30 (31)	131	37 (38)	168 (169)	-	15
Fenstertechnik	4	-	3	7	124	27 (33)	127	34 (49)	161 (176)	-	9
Lim&Lee (1)	3	64	6	6	61	37 (43)	67	43 (49)	110 (116)	1	14
Lim&Lee (2)	3	86	9	9	40	36 (43)	49	45 (52)	94 (101)	2	21
Lim&Lee (3)	3	96	12	12	29	33 (43)	41	46 (55)	87 (96)	3	28
Lim&Lee (4)	3	104	15	15	23	29 (43)	38	51 (58)	89 (96)	4	35

Tab. 3-7: Aufwand der k-fachen Punktaddition für |k| = 128 bit

* Die zweite Spalte der Tabelle gibt die jeweilige Größe *l* der Bitgruppe bzw. des Fensters in bit an.

Die Werte zeigen, daß schon die Vorausberechnung eines einzigen Hilfswertes $2^{|k|/2} \cdot P$ im Verfahren nach Lim&Lee den durchschnittlichen Gesamtaufwand auf weniger als 70% der sehr effizienten Fenstertechnik und auf weniger als 55% der Double&Add-Methode senkt. Durch die Vorausberechnung weiterer Hilfswerte kann der Aufwand unter 55% der Fenstertechnik (45% der Double&Add-Methode) gesenkt werden.[11]

$\|k\|$ = 160 bit		Vor.	Tabellen		'online'(max.)		Tabellen+'online' (max.)			Speicher	
Verfahren*	l	Dbl.	Dbl.	Add	Dbl.	Add	Dbl.	Add	Σ	fest	Tab.
Double&Add	1	-	-	-	159	79 (159)	159	79 (159)	238 (318)	-	1
Bitgruppen	4	-	7	7	156	37 (39)	163	44 (46)	207 (209)	-	15
Fenstertechnik	4	-	3	7	156	33 (40)	159	40 (47)	199 (206)	-	9
Lim&Lee (1)	3	80	6	6	77	33 (38)	83	40 (44)	123 (127)	1	14
Lim&Lee (2)	3	108	9	9	51	45 (51)	60	54 (60)	114 (120)	2	21
Lim&Lee (3)	3	120	12	12	37	46 (53)	49	59 (66)	108 (115)	3	28
Lim&Lee (4)	3	128	15	15	29	47 (53)	44	62 (68)	106 (112)	4	35

Tab. 3-8: Aufwand der k-fachen Punktaddition für $|k|$ = 160 bit

Dies gilt allerdings nur, wenn Punktaddition und -verdoppelung auf der verwendeten Elliptischen Kurve etwa denselben Aufwand besitzen. Wie in Abschnitt 3.4.1 gezeigt wird, läßt sich auf bestimmten (supersingulären) Kurven über $GF(2^n)$ der Aufwand einer Punktverdoppelung auf Quadrierungen (Rotationen) reduzieren und ist damit vernachlässigbar. In diesem Fall führt die Vorausberechnung von Hilfswerten nach Lim&Lee daher zu einer Vergrößerung des Aufwands (s. dunkel hinterlegte Spalten).

Mit der Anzahl der vorausberechneten Hilfswerte nimmt auch der Aufwand für die (dynamische) Berechnung der für die Bitgruppenmethode erforderlichen Tabellen zu. Damit steigt der Gesamtaufwand des Verfahrens bei der Vorausberechnung von mehr als 4 Hilfswerten (bei typischen Moduluslängen von 128-160 Bit) wieder an.

3.4 Optimierung durch Kurven- und Schlüsselwahl

Wie in Abschnitt 3.1.2 gezeigt, hat die Wahl des Endlichen Körpers der Charakteristik 2, über dem die Elliptische Kurve $E(GF(2^n))$ definiert wird, erheblichen Einfluß auf die Effizienz einer Implementierung. Auch mit der Wahl der Kurvenparameter und der Gestalt des Schlüssels läßt sich die Geschwindigkeit eines auf einer Elliptischen Kurve spezifizierten Digitalen Signaturverfahrens beeinflussen.

---

[11] Die Auswertung des Faktors k kann weiter verbessert werden, indem statt der statischen Bitgruppentechnik dynamische Verfahren zur gleichzeitigen Auswertung der k_i verwendet werden.

3.4.1 Verwendung supersingulärer Elliptischer Kurven

Eine für kryptographische Anwendungen sehr interessante Klasse Elliptischer Kurven bilden sogenannte *supersinguläre* Kurven: Die Mächtigkeit dieser Kurven läßt sich in einer geschlossenen Formel angeben und damit leicht bestimmen. Dadurch vereinfacht sich die Kurven- und Schlüsselgenerierung erheblich. Aus diesem Grund wurden insbesondere supersinguläre Kurven als Basis Digitaler Signatursysteme vorgeschlagen [3]. Es gilt:

E ist supersingulär *gdw.* j-Invariante $j(E) = 0$.

Elliptische Kurven über $GF(p)$ sind supersingulär, wenn für die Mächtigkeit der Kurve gilt: $\#E(GF(p)) = p + 1$. Nach (2.9) ist $j(E) = 0$ *gdw.* $a = 0$. Damit vereinfacht sich die Kurvengleichung (2.7) für supersinguläre Kurven zu:

$$y^2 = x^2 + b, \text{ mit } b \neq 0\,. \tag{3.13}$$

Der Aufwand von Punktaddition und -verdoppelung auf $E(GF(p))$ bleibt davon unberührt. Das ist anders bei supersingulären Kurven über $GF(2^n)$. Hier vereinfacht sich die Kurvengleichung (2.1) zu

$$y^2 + a_3 y = x^3 + a_4 x + a_6\,, \text{ mit } a_3 \neq 0\,. \tag{3.14}$$

Die Berechnung der Punktaddition (2.2) ändert sich dabei wie folgt [24]:

$$x_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)^2 + x_1 + x_2, & \text{falls } P_1 \neq P_2 \\ \dfrac{x_1^{\,4} + a_4^{\,2}}{a_3^{\,2}}, & \text{falls } P_1 = P_2 \end{cases}$$

$$y_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + y_1 + a_3, & \text{falls } P_1 \neq P_2 \\ \left(\dfrac{x_1^{\,2} + a_4}{a_3}\right)(x_1 + x_3) + y_1 + a_3, & \text{falls } P_1 = P_2 \end{cases} \tag{3.15}$$

Die Verdoppelung eines Punktes auf $E(GF(2^n))$ erfordert in diesem Fall keine Inversenbestimmung und ist daher erheblich effizienter als auf nicht-supersingulären Kurven. Es lassen sich für ungerade n drei und für gerade n sieben Isomorphieklassen supersingulärer Elliptischer Kurven über GF(2^n) unterscheiden [24, 22]. Für diese 10 Klassen läßt sich die Mächtigkeit von E bestimmen als:

$$\#E(GF(2^n)) = \begin{cases} 2^{n+1} \pm 2^{\frac{n+1}{2}}, & \text{für } n \text{ ungerade} \\ 2^{n+1} \pm 2^{\frac{n}{2}}, & \text{für } n \text{ gerade} \end{cases} \tag{3.16}$$

Im Hinblick auf die Effizienz ist die Isomorphieklasse für ungerade n mit $a_3 = 1$, $a_4 = a_6 = 0$, besonders interessant, d.h. die Elliptische Kurve mit der Kurvengleichung

$$y^2 + y = x^3 \tag{3.17}$$

Für diese Klasse Elliptischer Kurven gilt $\#E(GF(2^n)) = 2^{n+1}$. Der Berechnungsaufwand einer Punkteverdoppelung verringert sich damit auf vier Quadrierungen in $GF(2^n)$ und ist daher vernachlässigbar.[12]

Das Verfahren nach Lim&Lee (s. Abschnitt 3.3) zur Auswertung des Faktors *k*, das die Zahl der zu berechnenden Punktverdoppelungen durch Vorausbestimmung verringert, ist damit für supersinguläre Kurven über $GF(2^n)$ nicht interessant.

Für die Verwendung supersingulärer Elliptischer Kurven als Basis Digitaler Signatursysteme sprechen also zwei Punkte: die sehr einfache Bestimmung der Mächtigkeit Elliptischer Kurven und die erhebliche Beschleunigung der Verdoppelung eines Punktes auf geeigneten Kurven $E(GF(2^n))$.

Leider greift der MOV-Reduktionsalgorithmus besonders gut auf supersingulären Kurven [22]. So ist der Reduktionskoeffizient *r* von supersingulären Kurven über $GF(2^n)$ kleiner 5; der besonders interessante Fall (3.17) hat den Reduktionskoeffizienten $r = 2$.

Um vor den bekannten Algorithmen zur Bestimmung Diskreter Logarithmen ebenso sicher zu sein wie eine nicht-supersinguläre Kurve mit $r = 10$, müssen supersinguläre Elliptische Kurven über $GF(2^n)$ daher auf einem Galois-Feld mit einem Modulus wenigstens 2,5-facher Länge arbeiten. Damit verlieren supersinguläre Kurven die angeführten Effizienzvorteile; sie erzeugen zudem längere Signaturen.

3.4.2 Generierung von *k* mit kleinem Hamming-Gewicht

Wie in Abschnitt 3.3 gezeigt, hängt der Aufwand einer *k*-fachen Punktaddition in $E(GF(q))$ entscheidend von der Anzahl der durchzuführenden Additionen und damit vom Hamming-Gewicht des Faktors *k* ab. Schränkt man die Auswahl von $k \in [1..Ord(P)-1]$ auf Werte mit einem maximalen Hamming-Gewicht ein, kann ein konstanter Aufwand für die *k*-fache Punktaddition erzwungen werden:

Wählt man $\omega(k) = 30$, dann sinkt der Aufwand der Punktaddition bei Verwendung des Double&Add-Verfahrens auf 29 Additionen und, mit einer Vorausberechnung nach Lim&Lee, für $|k| = 128$ auf 63 Verdoppelungen (Gesamtaufwand: 92 Punktadditionen/-verdoppelungen) bzw. für $|k| = 160$ auf 79 Verdoppelungen (Gesamtaufwand: 108 Punktadditionen/-verdoppelungen).

Bisher ist kein Angriff bekannt, der diese Einschränkung von *k* nutzt.

---

[12] Die Quadrierungen sind bei Verwendung einer optimalen Normalbasis in $GF(2^n)$ „frei“ (siehe Abschnitt 3.1.2).

4 Digitale Signatursysteme

Die Vielzahl von Möglichkeiten, Kryptosysteme und Digitale Signatursysteme auf Elliptischen Kurven zu realisieren, hat zu einer großen Zahl an Verfahrensvorschlägen geführt. Ein von Menezes und Vanstone verfolgter Ansatz hat sich in der Normung durchgesetzt: Von IEEE wird seit Herbst 1994 ein Vorschlag für Elliptische-Kurven-Systeme, Teil eines „Standard for RSA, Diffie-Hellman and related Public-Key Cryptography“ (IEEE P1363, Part 4) diskutiert. Der Entwurf wurde in die ISO-Normung einbezogen und liegt inzwischen als Draft 8 vor [11]. Dort werden eine Variante des Diffie-Hellman-Verfahrens für die Schlüsselvereinbarung, ein Verschlüsselungsverfahren und insbesondere zwei Digitale Signatursysteme spezifiziert: Ein *Elliptic Curve Digital Signature Algorithm* (ECDSA) – Variante des vom amerikanischen NIST standardisierten *Digital Signature Algorithm* (DSA) [28] für Elliptische Kurven – und ein an das Originalverfahren von ElGamal [7] angelehntes *Elliptic Curve Signature Scheme* (ECSS).

Tabelle 4-1 zeigt einen Vergleich des Rechenaufwands für ECSS und ECDSA für unterschiedliche Elliptische Kurven. Dabei werden die Darstellungen in affinen und in homogenen Koordinaten unterschieden.

Bei der für die Messungen verwendeten Implementierung wurde die Auswertung des Faktors k nach Lim&Lee mit genau einer Vorausberechnung gewählt (Abschnitt 3.3); das Hamming-Gewicht von k wurde nicht beschränkt. Der Aufwand einer Hashfunktion wurde bei den Messungen nicht berücksichtigt und es wurde angenommen, daß Systemparameter (wie der zufällige Faktor k) vorausberechnet wurden.

Kurve		$E(GF(p))$				$E(GF(2^n))$			
		$\|p\| = 128$ bit		$\|p\| = 160$ bit		$n = 130$		$n = 162$	
Verfahren		*affin*	*hom.*	*affin*	*hom.*	*affin*	*hom.*	*affin*	*hom.*
ECSS/	*sign*	0,4 ms	0,4 ms	0,5 ms	0,5 ms	0,4 ms	0,4 ms	0,5 ms	0,5 ms
ECDSA	*verify*	4,3 s	0,55 s	5,9 s	0,65 s	15,5 s	13,7 s	23,9 s	18,3 s

Tab. 4-1: Aufwandsvergleich für ECSS/ECDSA

Da es genügt, ein Bit anstatt der gesamten y-Koordinate eines Kurvenpunktes zu speichern [22], besteht eine Signatur in allen Fällen aus einem Zahlenpaar (r, s) mit $|r| \approx |s| \approx |q|$ mit $q = p$ bzw. 2^n. Die Länge einer Signatur liegt also zwischen 256 und 320 bit.

Tabelle 4-2 zeigt im Vergleich dazu den Aufwand „klassischer“ Digitaler Signaturverfahren für übliche Moduluslängen. Auch bei diesen Messungen wurden Hashfunktionen und vorausberechnete Werte nicht berücksichtigt. Für das Testen einer RSA-Signatur wurde ein Expo-

nent voller Länge angenommen;[13] die Signatur wurde mit Hilfe des Chinesischen-Reste-Algorithmus bestimmt.

Verfahren		|p| = 512 bit	|p| = 640 bit	|p| = 768 bit
ElGamal	*sign*	0,002 s	0,003 s	0,003 s
	verify	0,56 s	0,95 s	1,54 s
DSS	*sign*	< 0,001 s	< 0,001 s	< 0,001 s
	verify	0,32 s	0,52 s	0,85 s
RSA	*sign*	0,16 s	0,24 s	0,33 s
	verify	0,29 s	0,48 s	0,77 s

Tab. 4-2: Aufwandsvergleich „klassischer" Digitaler Signatursysteme

Die Länge einer RSA-Signatur entspricht der Moduluslänge (d.h. 512 bis 768 bit). Eine DSS-Signatur besteht aus einem Paar (r, s) mit $|r| \approx |s| \approx 160$ bit (d.h. 320 bit); eine ElGamal-Signatur aus (r, s) mit $|r| \approx |s| \approx |p|$ (d.h. 1024 bis 1536 bit).

5 Bewertung

Für Digitale Signatursysteme auf Elliptischen Kurven wurden unterschiedliche Optimierungsansätze betrachtet und anhand von Aufwandsabschätzungen und Messungen mit Softwareimplementierungen bewertet. Es wurde gezeigt, daß Signatursysteme auf Elliptischen Kurven über $GF(p)$ ab einer Schlüssellänge von etwa 700 bit klassischen Signatursystemen (im Hinblick auf die Effizienz) überlegen sind. Dieser Effizienzvorteil nimmt für größere Schlüssellängen deutlich zu, da für die Lösung des Diskreten Logarithmusproblems (DLP) auf Elliptischen Kurven bis heute kein subexponentieller Algorithmus bekannt ist.

[13] Die Verwendung eines kürzeren öffentlichen Exponenten bei RSA erscheint inzwischen problematisch.

Literatur

[1] Aho, Alfred V.; Hopcroft, John E.; Ullman, Jeffrey D.: *The Design and Analysis of Computer Algorithms*. Addison Wesley, Massachusetts 1974.

[2] Agnew, G.B.; Mullin, R.C.; Vanstone, S.A.: *An Implementation of Elliptic Curve Cryptosystems Over $F_{2^{155}}$*. IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, June 1993, S. 804-813.

[3] Bender, Andreas; Castagnoli, Gui: *On the Implementation of Elliptic Curve Cryptosystems*. In: Brassard, G. (Hrsg.): Proceedings of Crypto ‘89, LNCS 435, Springer, Berlin 1990, S. 186-192.

[4] Brickell, E.; Gordon, D.M.; McCurley, K.S.; Wilson, D.: *Fast Exponentiation with Precomputation*. In: Rueppel, R.A. (Hrsg.): Proceedings of Eurocrypt ‘92, LNCS 658, Springer, Berlin 1993, S. 193-201.

[5] Bos, Jurjen; Coster, Mattijs: *Addition Chain Heuristics*. In: Brassard, G. (Hrsg.): Proceedings of Crypto ‘89, LNCS 435, Springer, Berlin 1989, S. 377-386.

[6] Bosselaers, Antoon; Govaerts, René; Vandewalle, Joos: *Comparision of three modular reduction functions*. In: Stinson, D.R. (Hrsg.): Proceedings of Crypto ‘93, LNCS 773, Springer, Berlin 1994, S. 175-186.

[7] ElGamal, Taher: *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*. IEEE Trans. on Inform. Theory, Bd. IT-31, Nr. 4, 7/1985, S. 469-472.

[8] Fox, Dirk: *Effiziente Softwareimplementierung asymmetrischer Kryptosysteme und der zugrundeliegenden modularen Langzahlarithmetik*. Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, 1991.

[9] Fox, Dirk: *Der 'Digital Signature Standard': Aufwand, Implementierung und Sicherheit*. In: Weck, G.; Horster, P. (Hrsg.): Verläßliche Informationssysteme, Proceedings der GI-Fachtagung VIS ‘93, DuD-Fachbeiträge 16, Verlag Vieweg, Braunschweig 1993, S. 333-352.

[10] Fumy, Walter; Hess, Erwin: *What are Today’s Alternatives for a Digital Signature Scheme?* Proceedings of Securicom ‘94, Paris, S. 23-33.

[11] IEEE P1363 Standard: *Standard for RSA, Diffie-Hellman and related public-key cryptography. Part 4: Elliptic Curve Systems (Draft 8).* 14.05.1996.

[12] Itoh, T.; Teechai, O., Tsujh, S.: *A fast algorithm for computing multiplicative inverses in GF(2^t) using normal bases*. J. Society for Electronic Communications (Japan), 44, 1986, S. 31-36.

[13] Jungnickel, Dieter: *Finite Fields. Structure and Arithmetics*. BI Wissenschaftsverlag, Mannheim 1993.

[14] Knuth, Donald Erwin: *The Art of Computer Programming. Bd. 2: Seminumerical Algorithms*. 2. Auflage, Addison-Wesley, Massachusetts 1981.

[15] Koblitz, Neal: *Elliptic Curve Cryptosystems*. Mathematics of Computation, Vol. 48, No. 177, Jan. 1987, S. 203-209.

[16] Koyama, K.; Tsuruoka, Y.: *A signed binary window method for fast computing over elliptic curves*. In: Brickell, E.F. (Hrsg.): Proceedings of Crypto ‘92, LNCS 740, Springer, Berlin 1993, S. 345-357.

[17] Lay, Georg-Johann; Zimmer, Horst G.: *Constructing Elliptic Curves with Given Group Order over Large Finite Fields*. In: Adleman, L.M.; Huang, M.-D. (Hrsg.): Proceedings of 1. Symposium on Algorithmic Number Theory, LNCS 877, Springer, New York 1995, S. 250-263.

[18] Lercier, R.; Morain, F.: *Counting the number of points on elliptic curves over finite fields: strategies and performances*, In: Guillou, L.; Quisquater, J.-J. (Hrsg.): Proceedings of Eurocrypt ‘95, LNCS 921, Springer, Berlin 1995, S. 79-94.

[19] Lim, C.H.; Lee, P.J.: *More Flexible Exponentiation with Precomputation.* In: Desmedt, Y. G. (Hrsg.): Proceedings of Crypto ‘94, LNCS 839, Springer, Heidelberg 1994, S. 95-107.

[20] Lehmann, Frank; Mauerer, Markus; Müller, Volker; Shoup, Victor: *Counting the Number of Points on Elliptic Curves over Finite Fields of Characteristic Greater than Three*. In: Adleman, L.M.; Huang, M.-D. (Hrsg.): Proceedings of 1. Symposium on Algorithmic Number Theory, LNCS 877, Springer, New York 1995, S. 60-70.

[21] McCarthy, D.P.: *Effect of Improved Multiplication Efficiency on Exponentiation Algorithms Derived from Addition Chains*. Mathematics of Computation, Vol. 46, No. 174, April 1986, S. 603-608.

[22] Menezes, Alfred J.: *Elliptic Curve Public Key Cryptosystems*. SECS 234, Kluwer Academic Publishers, Massachusetts, 1993.

[23] Menezes, Alfred J.; Okamoto, Tatsuaki; Vanstone, Scott A.: *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field*. IEEE Transactions on Information Theory, Vol. 39, No. 5, Sept. 1993, S. 1639-1646.

[24] Menezes, Alfred J.; Vanstone, Scott A.: *The Implementation of Elliptic Curve Cryptosystems*. In: Seberry, J.; Pieprzyk, J. (Hrsg.): Proceedings of Auscrypt ‘90, LNCS 453, Springer, Berlin 1990, S. 2-13.

[25] Menezes, Alfred J.; Vanstone, Scott, A.; Zuccherato, Robert J.: *Counting Points on Elliptic Curves over* F_{2^m}. Mathematics of Computation, Vol. 40, No. 201, Jan. 1993, S. 407-420.

[26] Miller, Victor: *Use of Elliptic Curves in Cryptology*. In: Williams, H.C. (Hrsg.): Proceedings of Crypto ‘85, LNCS 218, Springer, Berlin 1986, S. 417-426.

[27] Montgomery, Peter L.: *Modular Multiplikation without Trial Division*. Mathematics of Computation, Bd. 44, Nr. 170, 4/1985, S. 519-521.

[28] National Institute of Standards and Technology (NIST): *Digital Signature Standard (DSS)*. Federal Information Processing Standards Publication 186 (FIPS-PUB), 19. Mai 1994.

[29] Odlyzko, A.: *Discrete logarithms in finite fields and their cryptographic significance*. In: Beth, T.; Cot, N.; Ingemarsson, J. (Hrsg.): Proceedings of Eurocrypt ‘84, LNCS 209, Springer, Berlin 1995, S. 224-314.

[30] Röhm, Alexander: *Digitale Signaturverfahren unter Verwendung Elliptischer Kurven über* $GF(2^m)$. Diplomarbeit, Universität Siegen, April 1996.

[31] Sauerbrey, Jörg; Dietel, Andreas: *Ressource Requirements for the Application of Addition Chains in Modulo Exponentiation*. In: Rueppel, R.A. (Hrsg.): Proceedings of Eurocrypt ‘92, LNCS 658, Springer, Berlin 1993, S. 159-167.

[32] Sauerbrey, Jörg: *Langzahl-Modulo-Arithmetik für kryptographische Verfahren. Konzepte zur effizienten Implementierung*. Dissertation, Universität München, Deutscher Universitäts-Verlag, Wiesbaden 1993.

[33] Schaefer-Lorinser, Frank J.: *Arithmetik auf elliptischen Kurven zur Konstruktion von kryptographischen Einwegfunktionen*. Dissertation, Universität Karlsruhe, Verlag Shaker, Aachen 1993.

[34] Schoof, René: *Elliptic Curves over Finite Fields and the Computation of Square Roots mod p.* Math. of Comp., Vol. 44, No. 170, April 1985, S. 483-494.

[35] Silverman, Joseph H.: *The arithmetic of Elliptic Curves*. Graduate Texts in Mathematics 106, Springer-Verlag, New York, 1986.

[36] Yacobi, Yacov: *Exponentiating faster with Addition Chains.* In: Damgård, I. B. (Hrsg.): Proceedings of Eurocrypt ‘90, LNCS 473, Springer, Berlin 1991, S. 222-229.