



ECC – Kryptographie auf Basis elliptischer Kurven

Eine kurze Einführung



Inhaltsverzeichnis

1	Einleitung	3
1.1	Die Basis: das „elliptic curve discrete logarithm problem“	3
1.2	Die Kurvengleichung	3
1.3	Verwendete Zahlkörper	3
1.4	Graphische Punktaddition	4
1.4.1	Sonderfälle	4
1.4.2	Verallgemeinerung zur Punktaddition über diskrete Zahlkörper	4
2	Public-Key-Verfahren auf Basis elliptischer Kurven	5
2.1	ECDH	5
2.2	ECDSA	5
2.3	Verschlüsselung mit elliptischen Kurven	5
3	Wahl der elliptischen Kurve	6
3.1	Nachweisbar zufällig generierte Kurven	6
3.2	Klassen schwacher Kurven	6
3.3	Andere Klassen spezieller Kurven	6
4	ECC in Standards	7
5	Ein kurzer Vergleich zwischen RSA und ECC	8
5.1	Vergleichbare Schlüssellängen	8
5.2	Performance	8
5.3	Allgemeine Unterschiede	8
5.4	ECC auf Smartcards	9
6	Weitere Informationen	9

cv cryptovision GmbH

Munscheidstr. 14

45886 Gelsenkirchen

Tel.: +49 (209) 167 2450

Fax: +49 (209) 167 2461

info@cryptovision.com<http://www.cryptovision.com>

1 Einleitung

Kryptosysteme auf Basis elliptischer Kurven haben sich in den letzten Jahren immer weiter gegenüber etablierteren Verfahren wie z.B. RSA verbreitet. Dies hat eine Vielzahl von Gründen, auf die später im Detail eingegangen wird.

1.1 Die Basis: das „elliptic curve discrete logarithm problem“

Im Herzen von ECC (*elliptic curve cryptography*) steht die Tatsache, dass man auf einer elliptischen Kurve eine kommutative (oder abelsche) Gruppe definieren kann und dass in dieser Gruppe das diskrete Logarithmus Problem ECDLP (*elliptic curve discrete logarithm problem*) extrem schwer zu lösen ist.

Eine (kommutative) Gruppe im mathematischen Sinne ist dabei eine Menge mit einer dazu gehörenden Verknüpfung. Diese Verknüpfung erfüllt dabei die „normalen“ Rechenregeln. Benutzt man die multiplikative Schreibweise, d.h. schreibt man also „ \cdot “ für die Verknüpfung, so ist das diskrete Logarithmus Problem wie folgt definiert: es seien zwei Gruppenelemente g und h gegeben – bestimme eine ganze Zahl x , so dass $g^x = h$. Hierbei steht g^x für $g \cdot g \cdot \dots \cdot g$. Über „normale“ Zahlen müsste man also lediglich $x = \log_g(h)$, d.h. den Logarithmus, berechnen. Ist die benutzte Gruppe allerdings die Menge von Punkten auf einer elliptischen Kurve, so erweist sich dieses Problem bei einer ausreichend großen Gruppenordnung als enorm schwierig. (Für diese Gruppe verwendet man jedoch i.A. die additive Schreibweise, in der das Problem dann lautet: Gegeben P und Q bestimme x , so dass $Q = xP$. Da dies jedoch lediglich eine andere Art der Notation ist, spricht man auch weiterhin vom diskreten Logarithmus Problem über elliptische Kurven.)

1.2 Die Kurvengleichung

Eine elliptische Kurve über einen allgemeinen Zahlkörper ist definiert durch eine Gleichung der Form:

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

Die Punkte, die diese Gleichung erfüllen, zusammen mit einem weiteren Punkt, dem „Punkt im Unendlichen“, bilden eine Gruppe.

1.3 Verwendete Zahlkörper

Außer der verwendeten Kurve spielt bei einem Kryptosystem auf Basis elliptischer Kurven auch der benutzte Zahlkörper eine Rolle. Dabei gibt es in der Praxis zwei verschiedene Möglichkeiten. Entweder einen Primzahlkörper über eine große Primzahl p , abgekürzt mit $GF(p)$ für „Galois field“, oder einen Körper mit 2^m Elementen, abgekürzt mit $GF(2^m)$. (Theoretisch könnte jedoch jeder endliche Körper gewählt werden.) Bisher wurden noch keine Sicherheitsunterschiede zwischen den beiden Körpern festgestellt, solange die verwendeten Gruppen von Kurvenpunkten eine ähnlich große Ordnung haben. (Dazu später mehr.) $GF(p)$ hat zumeist in Software Vorteile, wohingegen $GF(2^m)$ für Hardware Vorteile mit sich bringt. Lediglich erwähnt sei an dieser Stelle auch, dass es für $GF(2^m)$ verschiedene Darstellungsmöglichkeiten gibt, da man diesen Körper auch als m -dimensionalen Vektorraum betrachten kann. Zwei gängige Wahlen für die Basis dieses Vektorraumes beruhen auf Polynom- bzw. Normalbasen. Der Einfachheit halber wird hier im Folgenden nur der Fall $GF(p)$ behandelt.

Für diesen Körper (sowie für die reellen Zahlen) lässt sich die allgemeine Kurvengleichung einer elliptischen Kurve E durch einfache Transformationen vereinfachen zu:

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

In dieser Form ist die Kurve achsensymmetrisch zur x -Achse.

1.4 Graphische Punktaddition

Über den reellen Zahlen lässt sich die Gruppenverknüpfung auf solch einer Kurve gut veranschaulichen. Dabei ist es allgemein üblich „+“ für diese Verknüpfung zu schreiben. Man sagt also, dass man zwei Punkte „addieren“ kann.

Will man z.B. die Summe zweier verschiedene Punkte P und Q bestimmen, so tut man das graphisch wie folgt: zunächst zeichnet man eine Gerade durch P und Q . Diese schneidet die Kurve E in genau einem weiteren Punkt R^* . Man spiegelt nun R^* an der x -Achse und erhält so einen weiteren Punkt auf der Kurve R . Es gilt: $P + Q = R$.

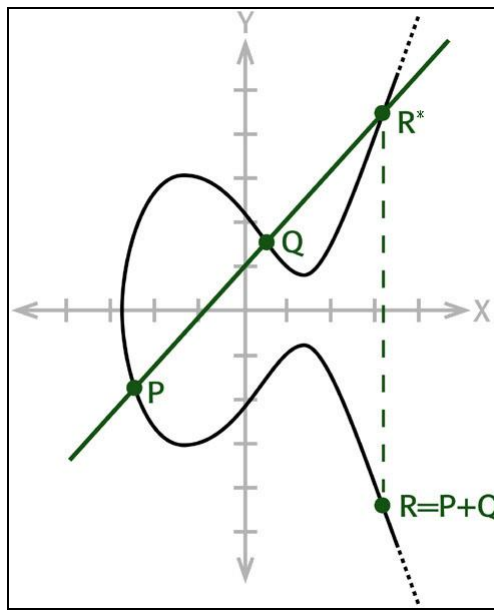


Abbildung 1: Graphische Punktaddition

1.4.1 Sonderfälle

Hätte man zufällig zwei Punkte gewählt, so dass der eine die Spiegelung des anderen an der x -Achse ist, hätte die sich daraus ergebene senkrechte Gerade die Kurve in keinem weiteren Punkt geschnitten. Um aber eine Gruppe zu erhalten, müssen sich zwei beliebige Elemente verknüpfen lassen. Um dies zu erreichen und gleichzeitig ein neutrales Element, also quasi eine „0“, zu erhalten, definiert man einen „Punkt im Unendlichen“ O , den man sich in unendlicher Entfernung entlang der y -Achse vorstellen kann. Aus dieser Definition ergibt sich z.B., dass die Spiegelung eines Punktes an der x -Achse seinem negativen Element entspricht. So ist dann z.B. $P + P^* = P + (-P) = O$. Außerdem sieht man unter der Verwendung der oben angegebenen graphischen Additionsregel auch, dass für alle Punkte $P + O = P$ gilt. Soll der Punkt P zu sich selber addiert werden, will man also berechnen, so muss man die obigen Regel leicht modifizieren. Anstatt die Gerade durch zwei verschiedene Punkte zu zeichnen, benutzt man nun die Tangente im Punkt P . Diese schneidet die Kurve ebenfalls in genau einem weiteren Punkt. Dies kann allerdings der Punkt im Unendlichen sein, falls die Tangente im betroffenen Punkt senkrecht ist.

Diese Regeln lassen sich zusammen fassen zu:

Die Summe der drei Schnittpunkte einer Geraden mit E ist O , der Punkt im Unendlichen.

1.4.2 Verallgemeinerung zur Punktaddition über diskrete Zahlkörper

Diese anschauliche Regel kann der interessierte Leser zusammen mit der Gleichung verwenden, um sich die dazu gehörenden algebraischen Punktadditionsformeln herzuleiten, also Formeln, die die x - und y -Koordinate von R in Abhängigkeit der x - und y -Koordinaten von P und Q angeben.

Diese Formeln erhalten ihre Gültigkeit, auch wenn die Kurve nicht mehr über den reellen Zahlen, sondern über den diskreten Zahlenkörper $GF(p)$ definiert wird.

Will man sich die Punkte auf solch einer Kurve veranschaulichen, so erhält man höchstens – indem man die einzelnen Punkte in ein Gitter einträgt – eine „Punktwolke“. Die Addition von zwei Punkten entspricht dann einem scheinbar zufälligen Springen von Punkt zu Punkt. Dieses „Chaos“ drückt die Schwierigkeit des diskreten Logarithmusproblems auf einer elliptischen Kurve aus.

2 Public-Key-Verfahren auf Basis elliptischer Kurven

Im Folgenden werden nun einige Protokolle bekannter Public-Key-Verfahren vorgestellt, die auf der Gruppenstruktur der Punkte beruhen.

Bei der Verwendung solch eines Verfahrens müssen sich alle Beteiligten über die benutzte Kurve und den benutzten Körper einig sein. Die Kurve ist dabei so gewählt, dass auf ihr eine Punktgruppe großer Primzahlordnung n existiert. Diese Angaben könnten in allgemeinen Systemparametern stecken oder extra Bestandteile jedes öffentlichen Schlüssels sein. Des Weiteren benötigt man einen allgemein bekannten Basispunkt P auf der Kurve der ebenfalls Teil der Systemparameter sein kann.

Der private Schlüssel von Alice ist dann eine einfache Zufallszahl x_A im Intervall $1 < x_A < n$. Ihr öffentlicher Schlüssel ist dann der Punkt Q_A , wobei $Q_A = x_A P$.

2.1 ECDH

Ein einfaches Protokoll für einen Schlüsselaustausch zwischen Alice und Bob, das auf dem traditionellen Diffie-Hellman Austausch beruht, funktioniert wie folgt:

Alice besorgt sich eine authentische Kopie von Bobs öffentlichem Schlüssel Q_B und berechnet $R_A = x_A Q_B$ unter Verwendung ihres geheimen Schlüssels x_A . Völlig analog besorgt sich Bob eine authentische Kopie von Alice öffentlichem Schlüssel Q_A und berechnet $R_B = x_B Q_A$. Nun ist aber $R_A = x_A Q_B = x_A x_B P = x_B x_A P = x_B Q_A = R_B$ und somit verfügen beide über den gleichen nur ihnen bekannten Schlüssel $R = R_A = R_B$.

Eine Lauscherin Eve steht vor dem Problem aus P , Q_A und Q_B , R berechnen zu wollen. Dies wird für genauso schwer wie das ECDLP (*elliptic curve discrete logarithm problem*) gehalten.

2.2 ECDSA

In der Praxis ist aber vor allem der Einsatz von ECC (*elliptic curve cryptography*) für die Erstellung digitaler Signaturen interessant. Das gängigste Verfahren ist hierbei der ECDSA (*elliptic curve digital signature algorithm*), eine Variante des DSA (*digital signature algorithm*).

Die genauen Details werden hier, um einen zu tiefen Einstieg in die Modulo-Arithmetik zu vermeiden, nicht vorgestellt. Wie die meisten Verfahren benutzt es jedoch eine Hashfunktion und eine Zufallszahl (zusätzlich zu dem privaten Schlüssel). Die Hashfunktion dient dabei der Reduktion der Nachricht auf einen kürzeren charakteristischen Wert, der dann aus Effizienzgründen statt der Nachricht selbst in die Signatur eingebaut wird. Die erzeugte Zufallszahl dient dem Schutz des privaten Schlüssels, der ebenfalls in die Signatur einfließt. Da die Signatur direkt von dem Hashwert der Nachricht und dem privaten Schlüssel, der nur einem einzelnen Benutzer bekannt ist, abhängt, kann eine Nachricht eindeutig mit einer bestimmten Person in Verbindung gebracht werden.

2.3 Verschlüsselung mit elliptischen Kurven

Es ist ebenfalls möglich, Verschlüsselungsalgorithmen mit elliptischen Kurven zu benutzen. Diese sind jedoch, wie alle Public-Key-Verschlüsselungsverfahren, deutlich langsamer als symmetrische Chiffren. Zu den bekanntesten ECC Verschlüsselungsalgorithmen gehören die

ElGamal-, Massey-Omura- und Menezes-Vanstone-Schemata. Es bietet sich jedoch an, ein Protokoll wie ECDH zum Austausch eines Schlüssels zu verwenden und diesen Schlüssel dann für ein symmetrisches Verfahren zu benutzen. Diese Mischung aus asymmetrischen und symmetrischen Chiffren nennt man hybride Verfahren.

3 Wahl der elliptischen Kurve

Da das ECDLP für eine bestimmte Kurve der zentrale Sicherheitsfaktor ist, sollte die verwendete elliptische Kurve mit großer Vorsicht gewählt werden. Hierbei gibt es generell die Möglichkeit, entweder „fertige“ Kurven – wie sie z.B. in verschiedenen Standards veröffentlicht werden – zu übernehmen, oder eigene Kurven zu generieren.

Der wichtigste Aspekt einer Kurve ist hierbei die Ordnung ihrer Punktegruppe. Damit eine ausreichende Sicherheit gewährleistet wird, muss es eine Untergruppe mit einer großen Primzahlordnung n geben. In der Praxis ist n zumeist eine 160-Bit große Primzahl.

Bis zum Anfang der 90-er Jahre war die Bestimmung der Kurvenordnung ein ernstes Problem, was die Generierung das Testen von Kurven sehr erschwert hat. Inzwischen sind die Verbesserungen von Schoofs Algorithmus durch Elkes und Atkin das gängigste Verfahren (abgekürzt: SEA), welches ein effizientes Berechnen ermöglicht.

3.1 Nachweisbar zufällig generierte Kurven

Die Einwegeigenschaft von Hashfunktionen bietet jedoch die Möglichkeit, Kurven nachweisbar zufällig zu erzeugen, indem man den seed der Hashfunktion, also das Argument, zusammen mit dem dadurch erzeugten Hashwert veröffentlicht und den Hashwert direkt zur Erstellung der Kurvenparameter verwendet. So kann man i.A. empfohlene Kurven guten Gewissens benutzen, ohne Angst haben zu müssen, dass jemand bewusst eine kryptographisch schwache Kurve generiert hat. In jedem Fall besteht die Möglichkeit, leicht solch eine fertige Kurve und vor allem selbst erzeugte Kurven auf alle bekannten Schwachstellen zu überprüfen.

3.2 Klassen schwacher Kurven

In den folgenden Fällen sind effiziente Algorithmen für das ECDLP bekannt. Solche Klassen von Kurven sind daher für die Kryptographie ungeeignet.

- Die Ordnung der Kurve hat keine große Primzahl als Faktor: Pohlig und Hellman haben gezeigt, wie das ECDLP in diesem Fall in kleinere Probleme unterteilt werden kann. Dieser Angriff kann in der Praxis leicht vermieden werden, indem man eine Kurve und einen Basispunkt benutzt, so dass die von diesem Punkt erzeugte Untergruppe von großer Primzahlordnung ist.
- Die Kurve ist super-singulär: Für diese spezielle Klasse von Kurven haben Menezes, Okamoto und Vanstone gezeigt, wie man das ECDLP auf das einfache DLP in Erweiterungskörpern von reduzieren kann. Es lässt sich aber sehr leicht feststellen, ob eine bestimmte Kurve super-singulär ist und sich dieser Angriff somit vermeiden.
- Die Kurve ist anomal, d.h. es gibt genau p Punkte auf einer Kurve über \mathbb{F}_p : Satoh und Araki haben gezeigt, dass für diese Kurven ein Algorithmus mit linearer Laufzeit für das ECDLP existiert.

3.3 Andere Klassen spezieller Kurven

Außerdem gibt es die weit verbreitete Ansicht, dass jede Art von Kurve, die auf irgendeine Weise speziell ist, für kryptographische Zwecke gemieden werden sollte, auch wenn bisher noch kein besonders effizienter Angriff entdeckt worden ist.

Zu dieser Klasse von Kurven gehören z.B. die Koblitz-Kurven über $\text{GF}(2^m)$, deren Kurvengleichungen nur die Koeffizienten „0“ und „1“ enthalten. Diese Kurven sind besonders

interessant, da es für sie hoch effiziente Algorithmen für die Punktarithmetik auf ihnen gibt. Jedoch weiß man inzwischen, dass es möglich ist, das Lösen des ECDLP auf ihnen um einen Faktor \sqrt{m} zu beschleunigen, was aber immer noch zu vernachlässigen ist.

Ein Verfahren, das sich komplexe Multiplikation nennt, ermöglicht es, Kurven mit einer bestimmten Punktordnung zu erzeugen. Für Kurven, die so erzeugt worden sind, besteht zwar bisher außer den allgemein anwendbaren und extrem langsamen Verfahren noch keine besondere Angriffsmöglichkeit, dennoch bilden sie ein „besondere“ Klasse von Kurven, so dass man solch eine Angriffsmöglichkeit in der Zukunft nicht ausschließen kann.

4 ECC in Standards

Mit der Verbreitung von ECC und deren Akzeptanz in verschiedenen Gremien, ist ECC inzwischen auch in vielen wichtigen Standards vertreten, um eine Kompatibilität zwischen unterschiedlichen Systemen zu gewährleisten. Im Folgenden seien exemplarisch einige dieser Standards aufgeführt:

IEEE P1363

Das Dokument Standard Specifications for Public Key Cryptography des Institute of Electrical and Electronics Engineers [P1363] umfasst Verfahren der Public-Key-Kryptographie sowie geeignete Parameter und Schlüssel. Die aufgeführten ECC-Protokolle beinhalten die Schlüsselerzeugung sowie verschiedene Varianten des Schlüsselaustauschs und der Digitalen Signatur.

Der aktuelle Stand und begleitende Informationen zu diesem Standard findet man unter <http://grouper.ieee.org/groups/1363/index.html>.

ANSI x9.62 und x9.63

Diese beiden Dokumente des American National Standards Institute liegen bisher als Draft vor und enthalten Standardisierungsvorschläge zur Public-Key-Kryptographie für die amerikanische Finanzdienstleistungsindustrie. Das Dokument x9.62 umfasst Vorschläge zum ECDSA (Elliptic Curve Digital Signature Algorithm), während x9.63 den Schlüsselaustausch mittels elliptischer Kurven enthält. In beiden Texten sind außerdem Beispiele für geeignete Parameter und Schlüssel angegeben.

ISO/IEC

ECC wurde in die folgenden ISO/IEC-Standards aufgenommen:

- ISO/IEC 14888-3: „Digital Signature with Appendix Part 3: Certificate-based Mechanisms“
- ISO/IEC 9796-4: „Digital Signature with Message Recovery, Discrete Logarithm-based Mechanisms“
- ISO/IEC 15946: „Cryptographic Techniques Based on Elliptic Curves“, I-III,

FIPS (Federal Information Processing Standard)

Das National Institute of Standards and Technologie (NIST) der US-Regierung hat den Standard FIPS 186-2 um das ECDSA-Signaturverfahren erweitert. Als Konsequenz ist es nun Regierungsinstitutionen der USA ohne weitere Genehmigung möglich, Sicherheitsprodukte mit ECC-Komponenten zu verwenden. Dies hat beträchtliche Signalwirkung sowohl auf andere US-Märkte als auch über die USA hinaus.

Neben diesen Standards, die sich unmittelbar mit Kryptographie beschäftigen, ist ECC auch Bestandteil vieler anwendungsorientierter Standards. Exemplarisch seien Standards der Internet Engineering Task Force (IETF) wie SSL/TLS, IPSEC, PKIX, S/MIME sowie das WAP-Protokoll genannt.

5 Ein kurzer Vergleich zwischen RSA und ECC

5.1 Vergleichbare Schlüssellängen

Wie schon erwähnt liegt der Nutzen von ECC gegenüber anderen gängigen Verfahren wie z.B. RSA vor allem in erheblichen Geschwindigkeitsvorteilen. Diese basieren darauf, dass die benötigten Schlüssellängen bei gleichem Sicherheitsniveau für ECC deutlich kürzer sind als bei klassischen asymmetrischen Verfahren, wie RSA.

Lenstra und Verheul (siehe auch Kapitel *Weitere Informationen*) entwickeln ein detailliertes Model zur Ermittlung geeigneter Schlüssellängen für zukünftige Anwendungen, das sich auch zum direkten Vergleich verschiedener kryptographischer Verfahren eignet. Dabei lassen sie neben den bereits beschriebenen Faktoren auch Prognosen über den Fortschritt in der Kryptoanalyse sowie den Einsatz speziell entwickelter Hardware eingehen. In der folgenden Tabelle ist ein Auszug dieser Empfehlungen aufgeführt. Der zum Berechnen eines privaten Schlüssels der jeweiligen Länge notwendige Aufwand ist in Rechenzeiten auf einem PC mit 450-Mhz-CPU und einer DEC VAX 11/780 (MIPS) angegeben.

Jahr	Schlüssellänge symmetrischer Verfahren	Asymmetrische Schlüssellänge (z.B. RSA)	Schlüssellängen von ECC	Erforderliche MIPS-Jahre	Erforderliche Jahre auf 450 Mhz PC
2000	70	952	132	$7.13 * 10^9$	$1.58 * 10^7$
2002	72	1028	137	$2.06 * 10^{10}$	$4.59 * 10^7$
2004	73	1108	141	$5.98 * 10^{10}$	$1.33 * 10^8$
2006	75	1191	145	$1.73 * 10^{11}$	$3.84 * 10^8$
2008	76	1279	149	$5.01 * 10^{11}$	$1.11 * 10^9$
2010	78	1369	153	$1.45 * 10^{12}$	$3.22 * 10^9$
2012	80	1464	157	$4.19 * 10^{12}$	$9.32 * 10^9$
2014	81	1562	162	$1.21 * 10^{13}$	$2.70 * 10^{10}$
2016	83	1664	166	$3.51 * 10^{13}$	$7.81 * 10^{10}$
2018	84	1771	170	$1.02 * 10^{14}$	$2.26 * 10^{11}$
2020	86	1881	175	$2.94 * 10^{14}$	$6.54 * 10^{11}$

5.2 Performance

Faire Zahlen zu finden, die die Geschwindigkeit eines ECC basierten Systems mit der eines RSA basierten Systems für dieselben Operationen (z.B. Schlüsselgenerierung, Erstellung/Überprüfung einer digitalen Signatur) und bei äquivalenter Sicherheit vergleichen, ist aus verschiedenen Gründen schwierig. Zum einen gibt es für beide Systeme verschiedene Beschleunigungsmöglichkeiten wie z.B. der Einsatz von Koblitzkurven (für ECC), des chinesischen Restsatzes (für RSA) oder der Verwendung von einem kleinen öffentlichen Exponenten (für RSA). Solche „Abkürzungen“ bergen aber stets zumindest theoretische Sicherheitsprobleme. Zum anderen kommt es natürlich auf die Implementierung an. Hier kann eine bestimmte Implementierung z.B. sehr maschinennah geschrieben sein, womit sie allerdings schlechter auf andere Systeme übertragbar ist, wohingegen eine andere Implementierung vielleicht sehr abstrakt und flexibel ist, dadurch aber auch deutlich langsamer wird. Zusätzlich kommt hinzu, dass es Hardware gibt, die speziell für die jeweiligen Systeme optimiert ist.

5.3 Allgemeine Unterschiede

Dennoch liegt es in der Natur der beiden Systeme, dass ECC vor allem für die Aufgaben, die den Einsatz des privaten Schlüssels bedürfen, also digitales Signieren und Entschlüsseln, i.A. um wenigstens einen Faktor 4 schneller ist. Bei der Überprüfung einer Signatur oder dem

Verschlüsseln ist hingegen RSA um einen ähnlichen Faktor schneller. Dieser Unterschied rührt vor allem daher, dass der öffentliche RSA Exponent meist bewusst so gewählt wird, dass möglichst effektiv gerechnet werden kann, wohingegen der sich daraus ergebende private Exponent keine Geschwindigkeitsvorteile mit sich bringt.

5.4 ECC auf Smartcards

In der Praxis ist es aber gerade von Interesse, die Operationen, die den privaten Schlüssel involvieren, auf eine in ihrer Rechenkapazität beschränkte Umgebung wie z.B. eine Smartcard auszulagern, wohingegen die Operationen mit dem öffentlichen Schlüssel i. A. auf einem gewöhnlichen PC stattfinden und dort selbst in ihrer „schlimmsten“ Form keine Probleme bereiten. So erscheint ECC gerade für den Einsatz in solchen Umgebungen vorteilhaft, da auch die verwendeten Schlüssel und der daraus resultierende Speicherbedarf deutlich kleiner sind als im Falle von RSA. An dieser Stelle sei nur kurz darauf hingewiesen, dass es genügt die x-Koordinate eines Punktes sowie ein weiteres Bit zu speichern, um einen beliebigen Punkt eindeutig identifizieren zu können und so zusätzlich Speicherplatz zu sparen. Dieses Verfahren wird als Punktkompression bezeichnet.

6 Weitere Informationen

Bücher zum Thema ECC

J. Silverman: *The Arithmetic of Elliptic Curves*, Springer Verlag, 1986.

A. Menezes, P. Oorschot, S. Vanstone: *Handbook of applied Cryptography*, CRC Press, 1996.

A. Lenstra, E. Verheul: *Selecting cryptographic Key Sizes*, in The Journal of Cryptology, Springer Verlag; online erhältlich unter <http://www.cryptosavvy.com>, 2002.

I. Blake, G. Seroussi, N. Smart: *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.

Links zum Thema ECC

„Elliptic Curve Cryptography“ (engl.); Integrity Sciences

<http://world.std.com/~dpj/elliptic.html>

„Elliptische Kurven“; Franz Lemmermeyer

<http://www.rzuser.uni-heidelberg.de/~hb3/ec00.ps>

„Elliptic Curves and Cryptology“ (engl.); Marc Joye

http://www.geocities.com/marcjoye/biblio_ell.html

„Elliptic Curves and Their Applications to Cryptography“; Andreas Enge

<http://www.math.uni-augsburg.de/~enge/buch/buch/buch.html>

cv cryptovision GmbH

<http://www.cryptovision.com>