# Elliptische Kurven in HBCI - ein Backup zu RSA

Detlef Hühnlein
**secunet** Security Networks AG
Mergenthalerallee 79-81
D-65760 Eschborn
huehnlein@secunet.de

**Zusammenfassung**

Erklärtes Ziel des ZKA ist es mittelfristig *alle* HBCI Transaktionen mit dem RSA-DES-Hybridverfahren (RDH) zu sichern. Allerdings kann niemand *garantieren*, daß nicht plötzlich ein leistungsfähiges Verfahren für das Faktorisieren großer Zahlen gefunden wird. Deshalb halten wir RSA's Monopolstellung in einem auf breiter Basis eingesetzten Verfahren wie HBCI für bedenklich, ja gar gefährlich. In diesem Beitrag soll die Notwendigkeit alternativer Verfahren diskutiert und deren Merkmale hergeleitet werden. Da Verfahren auf Basis Elliptischer Kurven eine gute Wahl zu sein scheinen, wollen wir auf diese Verfahren etwas näher eingehen und notwendige Änderungen für eine mögliche Integration Elliptischer Kurven in die HBCI-Spezifikation [ZKA99] aufzeigen.

## 1 Motivation

Die aktuelle HBCI-Spezifikation sieht als zentrale Sicherheitsmechanismen zwei Verfahren vor: Das DES-DES-Verfahren (DDV) gilt lediglich als Migrationslösung zum mittelfristig favorisierten RDH - Verfahren. Man erwartet also, daß in näherer Zukunft alle HBCI-Transaktionen durch ein einziges Public Key Verfahren, dem auf dem Faktorisierungsproblem beruhenden RSA, geschützt werden. Wenngleich das Faktorisierungsproblem und damit RSA [RSA79] als „wohluntersucht" gilt, so existiert natürlich *keinerlei Garantie*, daß das Faktorisierungsproblem für alle Zeit ein schwer zu lösendes, und damit für die Kryptographie geeignetes, Problem bleibt. Vielmehr unterstreicht die Historie der algorithmischen Zahlentheorie, daß man nie vor der Entdeckung neuer Verfahren, mit geringerer asymptotischer Laufzeit, gefeit ist. So glaubte man selbst in Fachkreisen lange Zeit, daß es wohl keinen besseren Algorithmus zum Faktorisieren von $n$ als den quadratischen Sieb [Pome82] mit einer asymptotischen Laufzeit[1] von $L_n[1/2,c]$ geben könnte. Man vermutete, daß der Parameter „1/2" in der Laufzeitfunktion aufgrund tiefer zahlentheoretischer Zusammenhänge grundsätzlich nicht zu verbessern sei. Daß dem nicht so ist zeigt eine *glückliche Idee* des Mathematikers John Pollard, die letztendlich zur Entwicklung des Zahlkörpersiebs [LeLe93], des heute (asymptotisch und praktisch) leistungsfähigsten Algorithmus zum Faktorisieren [BuLZ93] und Berechnung diskreter Logarithmen

[1] Für die Definition von $L_n[u,v]$ und einen gleichsam aktuellen und anschaulichen Überblick über den Stand der Kunst bei Algorithmen zum Lösen kryptographisch relevanter Probleme sei an [BuMa99] verwiesen.

[Gord93] in *GF(p)* geführt hat. Der Zahlkörpersieb besitzt eine Laufzeit $L_n[1/3,c]$ und wurde beispielsweise kürzlich zur Faktorisierung von RSA-140 [Riel99a] und RSA-155 [Riel99b] verwendet. Während sich diese Resultate, die im wesentlichen aufgrund steigender Rechenleistung und der Verbesserung einer konkreten Implementierung zustandekam, vorhersagen ließ, sind völlig neue Ansätze zum Faktorisieren in diesem Zusammenhang wesentlich gefährlicher. So war es kein geringerer als Adi Shamir, der kürzlich mit TWINKLE [Sham99] eine spezielle, kostengünstige Hardware-Architektur vorstellte, die die Erzeugung der Relationen für das Zahlkörpersieb, und somit das Faktorisieren, etwa um einen Faktor 100 bis 1000 beschleunigen kann. Wenngleich dies Ergebnis gottlob nicht den gefürchteten „Super-GAU“ mit sich brachte, so zeigt sich doch deutlich, daß sich neue Ideen, und damit verbundene Auswirkungen auf die Sicherheit eines Public Key Verfahrens, nicht vorhersehen lassen. Dieses allgemeine Problem gilt natürlich nicht nur für RSA, sondern vielmehr für alle zur Zeit bekannten Public Key Verfahren.

Wie J. Buchmann in diesem Zusammenhang sehr treffend bemerkte ist diese Situation einer allseitsbekannten Problematik sehr ähnlich: Die *Datensicherung auf einer Festplatte*: Mit dem heutigen Stand der Technik ist die plötzliche Funktionsunfähigkeit einer handelsüblichen Festplatte sehr unwahrscheinlich - aber eben nicht unmöglich. Da nun aber ein möglicher Datenverlust verheerende Folgen hätte, fertigt man ein Backup seines Datenbestandes an - vielleicht einfach auf einer zweiten Festplatte. Die Wahrscheinlichkeit, daß beide Festplatten *gleichzeitig* kaputt gehen ist sehr viel geringer. Daß die Anfertigung von Daten-Backups sinnvoll ist, ist unstrittig. So sollte es auch bei Public Key Verfahren sein.

Ziel dieser Arbeit soll es sein Public Key Verfahren, die möglicherweise diese Backup-Funktion in HBCI erfüllen könnten und notwendige Änderungen für deren Integration in die HBCI-Spezifikation zu diskutieren. Konkrete Maßnahmen für das Revozieren möglicherweise aller ausgestellten Zertifikate im Fall eines GAU‘s, werden hier *nicht* behandelt. Wieso wird hier gerade über HBCI diskutiert? ´Der Grund dafür ist ein recht einfacher: Im Gegensatz zu HBCI, sehen andere bedeutende Standardwerke, wie das SigG/SigV (mit den durch das BSI veröffentlichten Algorithmen), SET und viele IETF-Standards bereits mehrere alternative Public Key Verfahren vor und sind somit prinzipiell besser vor einem möglichen GAU gefeit. Außerdem scheint die Berücksichtigung alternativer Verfahren bei einem Verfahren mit wachsender Verbreitung wie HBCI besonders dringlich.

Die vorliegende Arbeit gliedert sich folgendermaßen: In Abschnitt 2 werden abstrakte Merkmale für alternative Public Key Verfahren für HBCI hergeleitet. In Abschnitt 3 wollen wir (kurz) potentielle Verfahren diskutieren. Da es scheint, daß Verfahren auf der Basis Elliptischer Kurven über endlichen Körpern dafür besonders geeignet sind, wird in Abschnitt 4 etwas näher auf diese Verfahren eingegangen. In Abschnitt 5 werden die von einer möglichen Änderung der HBCI-Spezifikation betrofffenen Passagen identifiziert und erste Änderungsvorschläge, als Grundlage für die Diskussion in den entsprechenden Standardisierungsgremien, angegeben. In Abschnitt 6 wird die vorliegende Arbeit kurz zusammengefasst und weitere Schritte angegeben.

# 2 Merkmale möglicher Alternativen zu RSA

In diesem Abschnitt wollen wir kurz die wichtigsten Merkmale alternativer Verfahren, die die oben geschilderte Backup-Funktion zu RSA realisieren könnten, herleiten und das wohl am besten geeignetste identifizieren.

## 2.1 Zielvorgaben

### 2.1.1 Sicherheit

Die Sicherheit des Verfahrens sollte auf einem Problem basieren, das im Idealfall (vermutlich) „echt schwerer“ ist, wie das Faktorisieren. An dieser Stelle sei angemerkt, daß bislang die Äquivalenz des RSA-Problemes (Berechnung *e*-ter Wurzeln mod *n*) mit dem Faktorisierungsproblem *nicht* bewiesen werden konnte. Eine etwas pragmatischere Forderung ist, daß das zugrundeliegendes Problem wohluntersucht und (zumindest scheinbar) wenig mit dem Faktorisierungsproblem korreliert ist. Das heißt, daß selbst die Entdeckung eines effizienten Algorithmus‘ für das Faktorisieren mit großer Wahrscheinlichkeit nicht zum Brechen des alternativen Verfahrens führt.

### 2.1.2 Effizienz

Da HBCI aus Sicherheitsgründen mittelfristig die breite Verwendung von Chipkarten vorsieht, ist es wichtig, daß die Berechnungen selbst auf einer weniger leistungsfähigen Chipkarte durchgeführt werden können. Durch die Speicherplatzrestriktionen auf der Karte ist vor allem auch die Länge der verwendeten Parameter,Schlüssel und Signaturen eine entscheidende Größe. Hier würde man sicherlich nur ungern ein Verfahren vorschlagen, das bezüglich Speicherplatz- und Laufzeiteffizienz hinter der „alten Lady“ RSA hinterherhinken würde.

# 3 Diskussion möglicher Verfahren

Im folgenden wollen wir kurz einige potentielle Kandidaten (-familien) diskutieren. Die zahlreichen Verfahren, wie ESIGN, Rabin, Fiat-Shamir, NICE, ..., die auf dem Faktorisierungsproblem beruhen wollen wir hier aus naheliegenden Gründen gänzlich außer Acht lassen. Wir betrachten die Systeme in der Reihenfolge ihrer Entdeckung.

## 3.1 DL-Problem in endlichen Körpern

Die Ursprüngliche Formulierung des DL-Problemes wurde für die multiplikative Gruppe endlicher (Prim-) körper gemacht. Die Effizienz dieser Verfahren ist etwa vergleichbar mit der von RSA. Das ist wenig verwunderlich, da für den wohl weitverbreitesten Fall *GF(p)* die gleiche modulare Arithmetik (in diesem Fall mit einer Primzahl *p* als Modul) verwendet wird. DSA - Signaturen sind nicht von der Größe des Moduls abhängig und deshalb mit 320 Bit deutlich kürzer als RSA-Signaturen. Allerdings scheint jede Verbesserung für einen subexponentiellen Faktorisierungsalgorithmus mehr oder weniger direkt zu einer Verbesserung des analogen Verfahrens zur Berechnung von diskreten Logarithmen in endlichen Körpern zu führen. So ist z.B. der jeweils beste Algorithmus für beide Probleme der Zahlkörpersieb [LeLe93].

## 3.2 DL in elliptischen Kurven über endlichen Körpern

Im Jahr 1985 schlugen Viktor Miller [Mill85] und Neal Koblitz [Kobl87] unabhängig voneinander das DL-Problem in der Punktegruppe elliptischer Kurven (EC) über endlichen Körpern als kryptographisches Primitiv vor. Das Hauptargument für die Verwendung von EC ist, daß bei geschickter Parameterwahl *kein subexponentieller Algorithmus* für die Berechnung diskreter Logarithmen bekannt ist. Deshalb darf der relevante Sicherheitsparameter, d.h. die (Unter-)Gruppenordnung $|E|$, sehr viel kleiner gewählt werden als z.B. bei *GF(q)**. Da nach dem Satz von Hasse $|E| \approx q$ dürfen demzufolge Kurven über "kleineren" endlichen Körpern verwendet werden, was wiederum zu sehr (Speicherplatz- *und* Laufzeit-) effizienten Implementierungen führt. Die meistverbreitetsten Körpertypen für die Kryptographie sind Primkörper *GF(p), p* prim oder Binäre Körper $GF(2^m)$. Während bei Kurven über *GF(p)* die meist bereits vorhandene RSA-Arithmetik verwendet werden kann, erlauben letztere besonders effiziente Implementierungen bei geschickter Darstellung der Körperelemente. In diesem Fall ist es bekanntlich sogar möglich bei akzeptabler Performance (Signatur in 600ms auf 8-Bit Standardcontroller) auf den kryptographischen Koprozessor zu verzichten.

## 3.3 DL-Problem in Zahlkörpern

Ein Problem, das bewiesenermaßen mindestens so schwer und vermutlich echt schwerer ist als das Faktorisierungsproblem ist das DL-Problem in quadratischen *Maximal*ordnungen ([BuWi88] und [BuWi89]) oder in beliebigen Zahlkörpern [BuPa97]. Aus Effizienzgründen, kommen diese Verfahren jedoch nicht in Frage. Es sei angemerkt, daß die äußerst effizienten Verfahren der NICE-Familie in einer Nichtmaximalordnung operieren. D.h. sie basieren auf dem Faktorisierungsproblem und scheiden deshalb aus.

## 3.4 DL in hyperelliptischen Kurven über endlichen Körpern

Eine natürliche Verallgemeinerung, wie 1989 wiederum von N. Koblitz vorgeschlagen, ist die Verwendung von hyperelliptischen Kurven (HEC) vom Geschlecht *g*. Eine HEC ist eine glatte durch

$$F(x, y) : y^2 + h(x)y = f(x), \qquad (1)$$

definierte Kurve, wobei der Grad von *h(x)* höchstens *g* und der Grad von *f(x)* höchstens $2g+1$ ist. Vergleicht man dies mit der Definition von EC's[2], so sieht man sofort, daß EC's auch als "HEC's vom Geschlecht 1" betrachtet werden können. Für eine detailiertere und dennoch lesbare Behandlung der Materie sei an [Kobl99] verwiesen. Auch für HEC's ist, bei geschickter Parameterwahl, kein subexponentieller Algorithmus bekannt. Insgesamt verhält sich die Situation hier sehr ähnlich, wie bei EC's. HEC's sind zugegebenermaßen etwas schwieriger zu vermitteln, sind noch nicht standardisiert und weniger weit verbreitet, was existierende Implementierungen angeht. Aus diesen Gründen scheinen EC's für unsere Zwecke geeigneter als ihre Pendants mit größerem Geschlecht.

[2] im nächsten Kapitel

## 3.5 DL-Problem in Funktionenkörpern

Auch hier kann, wie bei [BuWi89] die Infrastruktur der Hauptidealklasse im reellquadratischen Funktionenkörper zur Konstruktion von Kryptosystemen verwendet werden. Während hier, insbesondere bei Körpern der Charakteristik 2, eine vergleichsweise effiziente Implementierung möglich ist (siehe z.B. [MüVZ98]), so sind sie doch für die Implementierung auf Chipkarten und damit für den praktischen Einsatz weniger geeignet.

## 3.6 Gitter-Systeme

Das in [AjDw97] vorgestellte Verfahren ist von großem theoretischen Interesse, da das Brechen des Verfahrens bewiesenermaßen im Durchschnitt genauso schwierig ist, wie im schlimmsten Fall. Zum Brechen des Verfahrens müßte man kurze Gittervektoren berechnen, was (im worst case) ein sehr, sehr schwieriges (NP-vollständiges) Problem ist. Allerdings sind die nötigen Schlüssel um das einige hundertfache größer als beispielsweise bei RSA. Deshalb können diese Verfahren nicht im Zusammenhang mit Chipkarten eingesetzt werden. Das Verfahren [GoGH97] *schien* mit viel kürzeren Schlüsseln auszukommen und erlaubte zudem sehr effiziente Operationen. Leider wurde dieses Verfahren kürzlich gebrochen [Nguy99]. Etwas besser verhält es sich (noch?) bei NTRU [HoPS98]. Während dieses System etwas günstigere Laufzeit als RSA hat, sprechen die relativ großen öffentlichen Schlüssel (z.B. 1841 Bits vs. 1024 Bit RSA) und vor allem die Tatsache, daß NTRU auf einem neuen und noch relativ wenig untersuchten Problem beruht gegen eine Verwendung dieses Primitivs. Insgesamt scheinen Gittersysteme aus diesen Gründen kein sinnvolles Backup-Verfahren für RSA zu sein.

## 3.7 Zusammenfassung

Nach obiger (sehr kurzer) Diskussion potentiell interessanter Verfahren scheinen Kryptosysteme auf Basis Elliptischer Kurven das ideale Backup-Verfahren für RSA in HBCI zu sein. Die wichtigsten Argumente für EC sind, daß das DL-Problem scheinbar nicht mit dem Faktorisierungsproblem korelliert ist. Insbesondere scheint überhaupt kein subexponentieller Algorithmus dafür möglich zu sein. Außerdem sind sehr effiziente Implementierungen (z.B. gar auf Chipkarten ohne Koprozessor) vorhanden. Deshalb werden im folgenden möglicherweise notwendige Änderungen an der HBCI Spezifikation aufgezeigt.

# 4 Kryptosysteme auf Basis Elliptische Kurven

In diesem Abschnitt wollen wir lediglich die wichtigsten Notationen und Sicherheitsrelevanten Parameter Elliptischer Kurven angeben. Für eine ausführlichere Behandlung sei auf das vollständige Papier und z.B. [Mene93] verwiesen.

Der große Löwenanteil existierender Implementierungen verwendet Primkörper $GF(p)$ oder binäre Körper $GF(2^m)$. Während verschiedenste Standardisierungsgremien z.B. der IEEE, ISO und DIN zur Zeit Signatur- und Verschlüsselungsverfahren auf Basis Elliptischer Kurven (für $GF(p)$ und $GF(2^m)$ ) standardisieren, scheinen die ANSI-Standards X9.62 [ANSI98] und X9.63 [ANSI99] sehr ausgereift und besonders gut auf die Bedürfnisse von Banken zugeschnitten.

Sei $K=GF(q)$ ein endlicher Körper, wobei $q=p>3$ oder $q=2^m$. Eine Elliptische Kurve über $K$ ist definiert[3] als die Lösungsmenge $(x,y)\in K\times K$ der Gleichung

$$F(x,y):\begin{cases} y^2 = x^3 + ax + b & , falls\ q = p \\ y^2 + xy = x^3 + ax + b & , falls\ q = 2^m \end{cases}. \qquad (2)$$

mit einem zusätzlichen Punkt $\mathcal{O}=(\infty,\infty)$ im Unendlichen. Außerdem wird gefordert, daß die Kurven keine Singularitäten (beide partielle Ableitungen nach $x$ und $y$ gleichzeitig 0) besitzt. Formeln für die Gruppenverknüpfung findet man z.B. in [Mene93]. Weiterhin sei $P=(x_P,y_P)$ ein Punkt auf der Kurve, der eine prime (Unter-) gruppe der Ordnung $q_o$ erzeugt.

Um ausreichende Sicherheit (analog zu RSA mit 1024 Bit Modul) zu gewährleisten, sollten wie in [BSI99] angegeben, folgende Punkte bei der Parameterwahl berücksichtigt werden:

1. $q_o$ sollte ausreichend groß sein um "generische Algorithmen", wie Shank's Baby-Step-Giant-Step- oder Pollard's Rho-Algorithmus unmöglich zu machen. Da diese Algorithmen eine Laufzeit von $O(q_o^{1/2})$ haben ist $\log_2 q_o>159$ ausreichend.
2. Um im Fall $q=p$ die $p$-adische Berechnung des diskreten Logarithmus [Smar99] auszuschließen, muß $q_o \neq p$ sein. Da $p=q_o$ ist, ist notwendigerweise die Punktegruppe isomorph zu $GF(p)^+$, wo der DL bekanntlich mit dem Euklidischen Algorithmus (in Polynomialzeit) berechnet werden kann. Dieser Isomorphismus wird essentiell durch "Liften der Kurve" mod $p^2$ berechnet.
3. Es darf nicht möglich sein, die Kurve über $GF(q)$ in die multiplikative Gruppe eines Erweiterungskörpers $GF(q^r)$ mit relativ kleinem Erweiterungsgrad $r$ einzubetten [MeOV91]. Hier ist $r_0>10^4$, wobei $r_0:=\min(r:q_o \mid q^r-1)$, sicherlich ausreichend.
4. In [BSI99] sind zwei weitere Klassen *möglicherweise* schwacher Kurven angegeben:

*4a) Koblitz-Kurven über $GF(2^m)$*: Dies sind Kurven, deren Koeffizienten bereits in Teilkörpern definiert sind. Durch Ausnutzen des Frobenius-Automorphismus ist hier eine sehr effiziente Implementierung möglich.

*4b) kleine Klassenzahl:* Kurven, wobei die Maximalordnung des Endomorphismenringes eine sehr kleine Klassenzahl ($h(\Delta)<200$) besitzt. Diese Kurven haben den Vorteil, daß deren Erzeugung mit Methoden der komplexen Multiplikation sehr schnell möglich ist.

Wie bereits angedeutet, so ist auch für diese Kurven kein besserer Algorithmus zur DL-berechnung als Pollard's Rho[4] bekannt. Es ist vielmehr so, daß diese Kurven zusätzliche Strukturen aufweisen, die vielleicht zu einem besseren Algorithmus in den angegebenen Spezialfällen führen *könnten*. Nach dem heutigen Wissensstand sind jedoch auch diese Kurven bei ausreichender Gruppenordnung (d.h. $\log_2 q > 159$) als sicher einzustufen. Die Forderungen 4a) und 4b) sind deshalb *nicht* in den ANSI X9.{62,63} - Standards enthalten.

[3] Im Fall $q=2^m$ sind lediglich die für die Kryptographie interessanten nicht-supersingulären von der angegebenen Form (2).

[4] Für Koblitz-Kurven ist es nach [GaLV98] und [WiZu98] möglich Pollard's Rho-Algorithmus um einen konstanten Faktor (ca. 20) zu beschleunigen.

# 5 Notwendige Änderungen an der HBCI-Spezifikation

In diesem Abschnitt wollen wir nun die von einer möglichen Integration Elliptischer Kurven betroffenen Passagen in der aktuellen HBCI-Spezifikation 2.1 [ZKA99] identifizieren und erste (grobe) Formulierungsvorschläge angeben.

Da die HBCI-Spezifikation bereits sehr modular aufgebaut ist, werden lediglich an den folgenden sechs Punkten Änderungen nötig. Die erste Änderung betrifft die gesamte Spezifikation und ist eher redaktioneller, denn inhaltlicher Natur. Die weiteren fünf Änderungen betreffen ausschließlich Teil B (Kapitel VI - Sicherheit) der HBCI - Spezifikation; es müssen konkrete Kennungen und Unterkapitel für die neuen Verfahren auf Basis Elliptischer Kurven eingefügt werden.

## 5.1 ASH = (RDH oder ECD) statt RDH

An vielen Stellen an denen vom RDH - Verfahren die Rede ist, ist es in der Tat unerheblich, ob RSA und DES oder jede andere Kombination von asymmetrischen und symmetrischen Verschlüsselungsverfahren verwendet wird. Im Hinblick auf die kommende AES-Standardisierung, deren Entscheidung mitte nächsten Jahres erwartet werden darf, sollte hier deshalb ein weiterer Freiheitsgrad in die HBCI-Spezifikation eingefügt werden. Deshalb sollte, außer an den im folgenden näher spezifizierten Stellen, RDH durch ASH-Verfahren (für **A**symmetrisch - **S**ymmetrisch - **H**ybrid) ersetzt werden. Die Kennung ASH referenziert damit das bekannte RDH-Verfahren oder die im folgenden ausführlicher diskutierten neuen Sicherheitsverfahren **ECD** (für **E**lliptic **C**urve mit **D**ES-3)[5]. Eine mögliche (künftige) Erweiterung um **A**ES würde demnach die Kennungen RAH und ECA verwenden. Um Hersteller und Banken nicht unnötig unter Druck zu setzen, sollte das neue Verfahren ECD, zumindest in der Einführungszeit, „optional“ sein. Wie bereits weiter oben angesprochen, sollte auf die ANSI - Standards X9.62 und X9.63 verwiesen werden. Diese Standards decken bereits beide Kurventypen (über *GF(p)* und $GF(2^m)$) ab.

## 5.2 ECD-Signatur

Es sollte ein Kapitel „VI.2.1.3 ECD - Signatur“ eingefügt werden. Das Hashing könnte wie bei DDV und RDH mit RIPEMD160 erfolgen. Das wäre dann die einzige Abweichung vom ANSI X9.62 Standard [ANSI98], der lediglich SHA-1 als Hash-Algorithmus vorsieht. In Hinblick auf die angedachte internationale Entwicklung von HBCI sollte jedoch über die Integration von SHA-1 in HBCI und somit die komplette Unterstützung von ANSI X9.62 beraten werden.

In jedem Fall sollte die „Formatierung des Hashwertes“, die hier bei Verwendung von 160 Bit-Kurven trivial ist, sowie die Berechnung und Überprüfung der Signatur wie in ANSI X9.62 durchgeführt werden.

[5] Daß es sich hierbei um ein „hybrides“ Verfahren handelt ergibt sich implizit durch die Verwendung von EC und DES-3.

## 5.3 ECD-Verschlüsselung

Auch hier sollte ein Kapitel „VI.2.2.3 ECD - Verschlüsselung“ eingefügt werden. Als Verschlüsselungsalgorithmus sollte das EC-Analog des ElGamal Verschlüsselungsverfahrens, wie im ANSI X9.63, Kapitel 5.8.1 [ANSI99] verwendet werden.

## 5.4 Das Format für öffentliche Schlüssel

In Kapitel VI.5.1.5 von [ZKA99] ist das Format des öffentlichen RSA-Schlüssels festgelegt. Da zur Zeit noch keine Zertifikate zum Einsatz kommen wird dieses Format zum Transport des öffentlichen Schlüssels zwischen Kunde und Bank verwendet. Da auch im Fall Elliptischer Kurven nicht damit gerechnet werden kann, daß sofort eine Zertifizierungsinfrastruktur zur Verfügung steht, sollte auch hier ein Format für den Transport öffentlicher Schlüssel und Kurvenparameter festgelegt werden.

In der aktuellen Spezifikation besteht die DEG „öffentlicher Schlüssel“ aus den folgenden 7 Feldern: 1. Verwendungszweck, 2. Operationsmodus, 3. Verfahren, 4. Wert für Modulus, 5. Bezeichner für Modulus, 6. Wert für Exponent und 7. Bezeichner für Exponent.

Hier wird vorgeschlagen, die Möglichkeiten um EC-Parameter zu erweitern und die Namen der Felder etwas abstrakter zu formulieren. Konkret sollte man das erste Element (z.Zt. für Modulus verwandt) z.B. „Verfahrensparameter“ und das zweite Element (z.Zt. für Exponent verwandt) z.B. „öffentlicher Schlüssel“ nennen. Das heißt, daß die neue Spezifikation der DEG öffentlicher Schlüssel folgendermaßen aussehen könnte: (Die vorgeschlagenen Änderungen sind **fett** markiert)

| Nr. | Name | Typ | Format | Länge | Status | Anzahl | Restriktionen |
|---|---|---|---|---|---|---|---|
| 1 | Verwendungszweck für öffentlichen Schlüssel | GD | an | ..3 | M | 1 | 5, 6 |
| 2 | Operationsmodus, kodiert | GD | an | ..3 | M | 1 | 16, **26** |
| 3 | Verfahren Benutzer | GD | an | ..3 | M | 1 | 10, **20** |
| 4 | Wert für **Verfahrensparameter** | GD | bin | ..512 | M | 1 | |
| 5 | Bezeichner für **Verfahrensparameter** | GD | an | ..3 | M | 1 | 12, **22** |
| 6 | Wert für **öffentlicher Schlüssel** | GD | bin | ..512 | M | 1 | 65537 |
| 7 | Bezeichner für **öffentlicher Schlüssel** | GD | an | ..3 | M | 1 | 13, **23** |

**Erläuterungen:**

Nr. 1: keine Änderung, d.h. „5“ für OCF - Owner Ciphering, „6“ für OSG - Owner Signing.

Nr. 2: neben „16“ für DSMR (ISO9796)[6] sollte nun auch „26“ für ECC nach ANSI X9.62 [ANSI98] bzw. X9.63 [ANSI99] zugelassen werden.

Nr. 3: neben „10“ für RSA sollte nun auch „20“ für ECC zugelassen werden.

Nr. 4: anstatt „Wert für Modulus“ sollte das Feld mit „Wert für Verfahrensparameter“ abstrakter benannt werden. Ist das Feld Nr. 3=“20“ (und das Feld Nr. 5=“22“ für „EC-Parameters“) so ist dieses Feld als „EC-Parameters“, wie in ANSI X9.62 [ANSI98], Kapitel 6.3, Seite 20 definiert zu interpretieren.

Nr. 5: anstatt „Bezeichnung für Modulus“ sollte das Feld „Bezeichner für Verfahrensparameter“ benannt werden. Neben „12“ (für MOD) sollte auch „22“ für „EC-Parameters“ zugelassen sein.

Nr. 6: anstatt „Wert für Exponent“ sollte das Feld „Wert für öffentlicher Schlüssel benannt werden. Ist nun Nr. 3=“20“ (und Nr. 5=“22“) so ist das Feld als öffentlicher Schlüssel (Punkt) auf der in EC-Parameters definierten Kurve zu interpretieren. Die Definition dieses öffentlichen Schlüssels ist in „SubjectPublicKeyInfo“ in ANSI X9.62 [ANSI98], Kapitel 6.4, Seite 21 gegeben.

Nr. 7: anstatt „Bezeichnung für Exponent“ sollte das Feld „Bezeichner für öffentlicher Schlüssel“ benannt werden. Zulassen sollte man neben „13“ für „Exponent“ auch „23“ für EC-Punkt.

## 5.5 Die zugelassenen Signaturalgorithmen

Im Kapitel VI.5.2.3 der HBCI-Spezifikation [ZKA99] sollte neben „1“ für DES und „10“ für RSA auch „20“ für ECDSA, wie in ANSI X9.62 [ANSI98] definiert zugelassen werden. Alle anderen Werte bleiben hiervon unberührt.

## 5.6 Die zugelassenen Verschlüsselungsalgorithmen

Im Kapitel VI.5.4.2 der HBCI-Spezifikation [ZKA99] sollte im Feld Nr. 5 neben „5“ (für DES-verschlüsselter DES-Schlüssel) und „6“ (für mit RSA-verschlüsselter DES-Schlüssel) auch „7“ für „mit ECC verschlüsselter DES-Schlüssel“ zugelassen werden. Der zu verwendende Verschlüsselungsalgorithmus sollte wie in ANSI X9.63 [ANSI99], Kapitel 5.8.1 sein. Alle anderen Werte bleiben hiervon unberührt.

# 6 Zusammenfassung

In diesem Beitrag wurde die Notwendigkeit für die Integration alternativer Public Key Verfahren in HBCI diskutiert. Da Verfahren auf Basis Elliptische Kurven das ideale Backup-Verfahren zum bereits vorhandenen RSA zu sein scheinen, wurden die nötigen Änderungen an der HBCI-Spezifikation in Kapitel 5, als Diskussionsgrundlage für die HBCI-Standardisierungsgremien, angegeben. Da diese Änderung im wesentlichen die in Kapitel 5 identifizierten sechs Punkte betreffen würde, sollten einer zügigen Integration in die HBCI-Spezifikation wenige *technische* Hürden im Wege stehen.

[6] An dieser Stelle sollte evtl. um Mißverständnissen vorzubeugen in der aktuellen HBCI-Spezifikation geklärt werden, welche Bedeutung das Tag „16“ für *Chiffrier*schlüssel (d.h. Nr. 1 = „5“) hat.

## Literatur:


[AjDw97] M. Ajtai, C. Dwork: „A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence“, in Proceedings of the 29th annual ACM Symposium on Theory of Computing, 1997, SS. 284-293

[ANSI98] ANSI-X9.62 (WD): „Public Key Cryptography for the Financial Service Industry - The Elliptic Curve Digital Signature Algorithm (ECDSA)“, 20.09.1998, via http://grouper.ieee.org/groups/1363/index.html

[ANSI99] ANSI-X9.63 (WD): „Public Key Cryptography for the Financial Service Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography“, 08.01.1999, via http://grouper.ieee.org/groups/1363/index.html

[BuLZ93] J. Buchmann, J. Loho, J. Zayer: „An implementation of the general number field sieve“, Advances in Cryptology Crypto (1993), Lecture Notes in Computer Science, 773, SS. 159-165

[BuMa99] J. Buchmann, M. Maurer: „Wie sicher ist die Public Key Kryptographie?“, unveröffentlichtes Manuskript, 28.01.1999, via http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/Welcome.html#1999

[BuPa97] J. Buchmann, S. Paulus: „A One Way Function Based on Ideal Arithmetic in Number Fields“, Advances in Cryptology: CRYPTO '97, Springer LNCS 1294, 1997. SS. 384-394.

[BuWi88] J. Buchmann, H.C. Williams: „A key-exchange system based on imagninary quadratic fields“, Journal of Cryptology, 1, 1988, SS. 107-118

[BuWi89] J. Buchmann, H.C. Williams: „A Key Exchange System Based on Real Quadratic Fields“, Advances in Cryptology: CRYPTO '89, Springer, LNCS 435, 1989, SS. 335-343.

[BSI99] Bundesamt für Sicherheit in der Informationstechnik (BSI): "Geeignete Kryptoalgorithmen gemäß §17 (2) SigV", 29.05.99, via http://www.bsi.de/aufgaben/projekte/pbdigsig/index ... Literatur

[GaLV98] R. Gallant, R. Lambert, S. Vanstone: „Improving the parallelized Pollard lambda search on binary anomalous curves“, manuscript 1998, via http://grouper.ieee.org/groups/1363/contrib.html#papers-cryptanalysis

[GoGH97] O. Goldreich, S. Goldwasser, S. Halevi: „Public Key Cryptosystems from Lattice Reduction Problems“, Advances in Cryptology: CRYPTO '97, Springer LNCS 1294, 1997. SS. 112-131

[DoLe95] B. Dodson and A.K. Lenstra: "NFS with four large primes: An explosive experiment", Proceedings of Crypto '95, Springer-Verlag, 1995, SS. 372-385

[Gord93] D. Gordon: "Discrete Logarithms in GF(p) Using the Number Field Sieve", Siam Journal on Discrete Mathematics 6, 1993, SS. 124-138

[HoPS98] J. Hoffstein, J. Pipher, J. Silverman: „NTRU: A ring based public key cryptosystem“, in Proceedings of ANTS III, Springer LNCS 1423, 1998, SS. 267-288

[Kobl87] N. Koblitz: "Elliptic Curve Cryptosystems", Math. Comp., vol. 48, 1987, 203-209

[Kobl89] N. Koblitz: "Hyperelliptic Cryptosystems", Journal of Cryptology, 1, 1989, SS. 139-150

[Kobl99] N. Koblitz: "Algebraic Aspects of Cryptography", Algorithms and Computation in Mathematics, vol. 3, Springer, 1998, ISBN 3-540-63446-0

[LeLe93] A.K. Lenstra, H.W. Lenstra: "The Development of the Number Field Sieve (LNM 1554), Springer, 1993

[MeOV91] A.J. Menezes, T. Okamoto und S.A. Vanstone: "Reducing elliptic curve logarithms to logarithms in finite fields", Proceedings of 23rd Annual ACM Symposium on Theory of Computing (STOC), 1991, SS. 80-89

[Mene93] A.J. Menezes: "Elliptic Curve Public Key Cryptosystems", Kluwer Academic Publishers, 1993, ISBN 0-7923-9368-6

[Mill85] V. Miller: "Use of Elliptic Curves in Cryptology", Advances in Cryptology: Proceedings of Crypto '85, LNCS 218, Springer, 1986

[MüVZ98] V. Müller, S. Vanstone, R. Zuccherato: „Discrete Logarithm Based Cryptosystems in Quadratic Function Fields of Characteristic 2“, Designs, Codes and Cryptography, 14 (2), 1998, SS. 159 - 178

[Nguy99] P. Nguyen: "Cryptoanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '99", Advances in Cryptography - Proceedings of Crypto '99, LNCS 1666, Springer, 1999, SS. 288-304

[Pome82] C. Pomerance: „Analysis and Comparison of Some Integer Factoring Algorithms“, in H.W. Lenstra, Jr. and R. Tijdeman, „Computational Methods in Number Theory“, Part I, Mathematisch Centrum Tract 154, Amsterdam 1982, SS. 89-139

[Riel99a] H.J.J. te Riele & al: „Factorization of RSA-140 with the Number Field Sieve“, Posting in sci.crypt.research, Februar 1999, erscheint auch bei ASIACRYPT, Nov. 1999

[Riel99b] H.J.J. te Riele & al: „Factorization of RSA-155“, Posting in sci.crypt.research, August 1999

[RSA79] R. Rivest, A. Shamir, L. Adleman: „A method for obtaining Digital Signatures and Public-Key-Cryptosystems", Communications of the ACM, v.21,n.2, Feb 1978, SS. 120-126

[Sham99] A. Shamir: „Factoring integers with the TWINKLE device“, erschienen bei CHES’99, August 1999, erscheint auch in Springers LNCS

[Smar99] N. P. Smart: "The Discrete Logarithm Problem on Elliptic Curves of Trace One", Journal of Cryptology, 1999, SS. 193-196

[WiZu98] M. Wiener, R. Zuccherato: „Faster Attacks on Elliptic Curve Cryptosystems“, erschienen bei SAC’98, Springer Verlag, oder via http://grouper.ieee.org/groups/1363/contrib.html#papers-cryptanalysis

[ZKA99] ZKA: „Home Banking Computer Interface - Spezifikation“, Version 2.1 vom 02.03.1999, via http://www.hbci-zka.de/spezifikation/2.1_body.html