5. Praktische Konsequenzen

Die in Kapitel 4 vorgestellten Angriffe auf das DL-Problem auf einer elliptischen Kurve haben Konsequenzen für die Auswahl kryptographisch sicherer Kurven. Diese wollen wir im ersten Abschnitt besprechen. Danach werden kurz einige Angriffe auf das RSA-Verfahren und auf das DL-Problem in der multiplikativen Gruppe $\mathbb{F}_q^\times$ beschrieben, die effizienter sind als die allgemeinen Methoden aus Kapitel 4. Im letzten Abschnitt gehen wir noch einmal ausführlicher auf digitale Unterschriften ein.

5.1 Geeignete elliptische Kurven

Wie wählt man eine elliptische Kurve $E(\mathbb{F}_q)$ und einen Punkt $P \in E(\mathbb{F}_q)$, so daß das zugehörige DL-Problem möglichst schwer zu lösen ist? Dazu muß man die in Kapitel 4 vorgestellten Attacken berücksichtigen. Wir haben gesehen, daß die allgemeinen Verfahren 4.1.1, 4.1.2, 4.1.4 und 4.1.5 angewandt auf $E(\mathbb{F}_q)$ und einen Punkt P der Ordnung n Laufzeiten von bestenfalls $O(\sqrt{n})$ Gruppenoperationen haben. Nun ist offenbar $n \leq \#E(\mathbb{F}_q)$, also nach dem Satz von Hasse

$$n \leq q + 1 + 2\sqrt{q} = (\sqrt{q} + 1)^2.$$

Daher gilt $n = O(q)$, so daß diese Verfahren exponentielle Laufzeiten in $\log q$ haben. Man möchte die Laufzeiten hier in $\log q$ ausdrücken, da die Größe der Eingabeparameter logarithmisch in q ist. Diese sind ja Punkte in $E(\mathbb{F}_q)$, werden also durch zwei affine Koordinaten in $\mathbb{F}_q$ beschrieben.

Die beiden speziellen Verfahren, die wir in Kapitel 4 vorgestellt haben, haben dagegen subexponentielle Laufzeiten. Man kann sie allerdings durch geschickte Auswahl der elliptischen Kurve umgehen.

A. Werner, *Elliptische Kurven in der Kryptographie*

Damit der MOV-Algorithmus nicht funktioniert, darf die Ordnung n der Untergruppe

$$\langle P \rangle \subseteq E(\mathbb{F}_q)$$

(oder zumindest ihr größter zu $p = \mathrm{char}\,(\mathbb{F}_q)$ teilerfremder Faktor) kein Teiler von $(q^k - 1)$ sein, falls das DL-Problem in $\mathbb{F}_{q^k}^{\times}$ berechenbar ist. Außerdem darf n kein Vielfaches von $p = \mathrm{char}(\mathbb{F}_q)$ sein, um einen Angriff mit dem verallgemeinerten SSSA-Verfahren auszuschließen.

Damit das DL-Problem in $\langle P \rangle$ auch mit einem allgemeinen Verfahren nicht angreifbar ist, muß n zum Schutz vor Pohlig-Hellman einen Primteiler l haben, der so groß ist, daß das DL-Problem in einer Gruppe mit l Elementen für das Pollard-ρ- oder das BSGS-Verfahren nicht zugänglich ist. In der Praxis wird man meist n selbst schon als Primzahl wählen. Dann gilt $n > 2^{160}$ im Moment als hinreichend sicher.

5.2 Vergleich mit anderen Public Key-Verfahren

5.2.1 RSA

Wir haben in Abschnitt 1.1 schon gesehen, daß man das RSA-Verfahren zum RSA-Modulus

$$n = pq$$

brechen kann, wenn man zu gegebenem n die Primfaktoren p und q bestimmen kann. Allgemein wird die Aufgabe, zu einer gegebenen Zahl n ihre Primfaktorzerlegung zu berechnen, Faktorisierungsproblem genannt. Dieses läßt sich schrittweise lösen, wenn man über einen Algorithmus verfügt, der zu n einen nichttrivialen Teiler d bestimmt. Dann kann man nämlich, falls d und $\frac{n}{d}$ nicht prim sind, nichttriviale Teiler dieser Zahlen bestimmen usw.

Für das Faktorisierungsproblem gibt es eine ganze Reihe von Algorithmen, von denen wir hier nur einige kurz ansprechen wollen. Eine ausführliche Darstellung findet man in [Co]. Ein einfaches Verfahren ist das sogenannte $(p-1)$-Verfahren von Pollard:

$(p-1)$-Verfahren: Hiermit kann man alle Primfaktoren p von n finden, für die $p-1$ nur "kleine" Primteiler hat. Genauer gesagt fixiert man eine Schranke $B \geq 1$ und nennt eine ganze Zahl m

$$B\text{-glatt},$$

falls m nur Primteiler $\leq B$ hat. Pollards $(p-1)$-Algorithmus findet dann alle Primteiler p von n, für die $(p-1)$ eine B-glatte Zahl ist. Dazu sei C das kleinste gemeinsame Vielfache aller Potenzen q^r von Primzahlen $q \leq B$, für die $q^r \leq n$ ist. Wir müssen hier also für alle $q \leq B$ die größte Primzahlpotenz $q^r \leq n$ berücksichtigen, d.h. r ist die größte ganze Zahl mit $r \log q \leq \log n$, mit anderen Worten

$$r = \left[\frac{\log n}{\log q}\right].$$

Daher gilt

$$C = \prod_{q \leq B \text{ Primzahl}} q^{[\log n / \log q]}.$$

Falls nun p ein Primteiler von n ist, für den $(p-1)$ eine B-glatte Zahl ist, so muß $p-1$ ein Teiler von C sein.

Daher gilt nach dem kleinen Satz von Fermat

$$a^C \equiv 1 \bmod p$$

für jede zu p teilerfremde Zahl a. Der Primteiler p von n teilt also für alle zu p teilerfremden Zahlen a den Term

$$a^C - 1.$$

Daraus ergibt sich folgender Algorithmus zur Bestimmung eines Teilers von n:

1) Wähle eine ganze Zahl a mit $2 \leq a \leq n-1$, die teilerfremd zu n ist.
2) Berechne $d = \mathrm{ggT}(a^C - 1, n)$.
3) Falls $d = 1$ oder $d = n$ ist, so ist das Verfahren mit diesen Ausgangsdaten gescheitert. Ansonsten ist d ein echter Teiler von n.

Falls n keinen Primteiler p besitzt, für den $(p-1)$ eine B-glatte Zahl ist, so ist diese Methode natürlich zum Scheitern verurteilt. Entweder man probiert es dann erneut mit einem größeren B oder man geht zu einem anderen Verfahren über.

Es gibt eine Weiterentwicklung des $(p-1)$-Algorithmus, bei der die Gruppe $(\mathbb{Z}/p\mathbb{Z})^\times$ (in der man a auffassen kann) durch eine elliptische Kurve $E(\mathbb{F}_p)$ ersetzt wird. Diese Faktorisierungsmethode mit elliptischen Kurven wurde von H.W. Lenstra entwickelt (siehe [Le] und [BSS], IX.1).

Wir wollen nun noch eine andere Idee zur Faktorisierung von n vorstellen, die für eine Reihe von sehr erfolgreichen Algorithmen grundlegend ist.

Strategie: Suche nach ganzen Zahlen x und y, für die

$$x^2 \equiv y^2 \bmod n,$$

$$\text{aber weder } x \equiv y \bmod n \text{ noch } x \equiv (-y) \bmod n$$

gilt. In diesem Fall ist n ein Teiler von $x^2 - y^2$, aber weder von $x - y$ noch von $x + y$. Da

$$x^2 - y^2 = (x - y)(x + y)$$

ist, muß die Zahl

$$d = ggT(x - y, n)$$

also ein nichttrivialer Teiler von n sein. Falls n etwa wie im RSA-Algorithmus genau zwei ungerade Primteiler hat, so hat die Gleichung $x^2 \equiv y^2 \bmod n$ für gegebenes x genau vier Lösungen y, wie man mit Hilfe des Chinesischen Restsatzes sehen kann. Zwei davon, nämlich x und $-x$, sind für unsere Zwecke unbrauchbar.

Wie findet man ein solches Paar (x, y)? Die Grundstrategie einer Reihe von Algorithmen besteht in der Wahl einer sogenannten Faktorbasis, etwa der Menge

$$S = \{p_1, \ldots, p_t\}$$

der ersten t Primzahlen. Dann werden Paare ganzer Zahlen (a_i, b_i) bestimmt, so daß

$$a_i^2 \equiv b_i \bmod n$$

ist und b_i eine p_t-glatte Zahl, d.h. von der Form

$$b_i = (-1)^{e_{i0}} \prod_{j=1}^{t} p_j^{e_{ij}}$$

ist. Nun sucht man nach Produkten der b_i, die Quadrate einer natürlichen Zahl sind. Dazu genügt es, die Exponenten e_{ij} zu betrachten, deren Summe nämlich gerade sein muß, damit das entsprechende Produkt der b_i ein Quadrat ist.

Daher betrachten wir die Vektoren

$$v_i = (e_{i0} \bmod 2, e_{i1} \bmod 2, \ldots, e_{it} \bmod 2) \in \mathbb{F}_2^{t+1}.$$

Wir brauchen so viele (a_i, b_i), daß die zugehörigen Vektoren v_i linear abhängig in $\mathbb{F}_2^{t+1}$ sind. Dann gibt es eine Teilmenge I von Indizes i, so daß

$$\sum_{i \in I} v_i = 0 \text{ in } \mathbb{F}_2^{t+1}$$

gilt. Das entsprechende Produkt $b = \prod_{i \in I} b_i$ ist also Quadrat einer natürlichen Zahl

$$y = \sqrt{b}.$$

Weiterhin ist die Zahl

$$\prod_{i \in I} a_i^2 \equiv \prod_{i \in I} b_i \equiv b \bmod n$$

definitionsgemäß das Quadrat von

$$x = \prod_{i \in I} a_i.$$

Wir haben also ein Paar (x, y) gefunden mit $x^2 \equiv y^2 \bmod n$.

Falls $x \not\equiv \pm y \bmod n$, so können wir wie oben beschrieben einen nichttrivialen Teiler von n angeben. Ansonsten müssen wir das Verfahren mit einer anderen Linearkombination der v_i oder mit anderen Daten (a_i, b_i) wiederholen.

In dieser Beschreibung der Grundidee haben wir allerdings noch nichts zur Wahl der Menge S, der Zahlen (a_i, b_i) oder zur Bestimmung der Indexmenge I gesagt.

Dazu gibt es verschiedene erfolgreiche Methoden, etwa das quadratische Sieb (siehe [Bu], 8.3) oder das Zahlkörpersieb (siehe [Co], 10.5). Unter einigen plausiblen Annahmen kann von beiden Verfahren gezeigt werden, daß sie subexponentielle Laufzeit haben.

Da es für eine geschickt ausgewählte elliptische Kurve keinen Algorithmus subexponentieller Laufzeit für das DL-Problem gibt, kommt man bei kryptographischen Verfahren mit elliptischen Kurven bei gleicher Sicherheit mit kürzeren Schlüssellängen aus als bei RSA. So gelten z.B. das RSA-Verfahren mit einem Modulus n der Größenordnung 2^{1024} (d.h. es werden 1024 Bits zum Speichern von n gebraucht) und elliptische Kurven $E(\mathbb{F}_q)$ mit einem Element $P \in E(\mathbb{F}_q)$, dessen Ordnung die Größenordnung 2^{139} hat, als berechnungsmäßig äquivalent. Diese und viele weitere Berechnungen von Schlüssellängen finden sich in [Le-Ver]. Zudem wächst die Schlüssellänge bei elliptischen Kurven mit steigendem Sicherheitsbedürfnis wesentlich langsamer an als die Schlüssellänge beim RSA-Verfahren. Das macht elliptische Kurven vor allem in Anwendungen für Medien mit begrenzter Speicherkapazität attraktiv.

5.2.2 DL-Verfahren in $\mathbb{F}_q^\times$

Statt der Punktegruppen $E(\mathbb{F}_q)$ zu elliptischen Kurven über endlichen Körpern kann man auch die multiplikativen Gruppen $\mathbb{F}_q^\times$ endlicher Körper auf ihre kryptographische Verwendbarkeit untersuchen. Dazu muß man das DL-Problem in solchen Gruppen studieren.

Zunächst lassen sich natürlich alle in 4.1 beschriebenen allgemeinen Angriffe auch für die Gruppe $\mathbb{F}_q^\times$ durchführen. Damit gelangt man aber nur zu Algorithmen exponentieller Laufzeit.

Es gibt für die Gruppe $\mathbb{F}_q^\times$ allerdings effektivere Verfahren, nämlich die sogenannten

Indexkalkül Methoden: Wir beschreiben diese Verfahren hier der Einfachheit halber nur für die multiplikativen Gruppen $\mathbb{F}_p^\times$ von Primkörpern der Charakteristik größer als 2, d.h. es sei $q = p$ eine Primzahl > 2.

Wir wollen das DL-Problem in der Gruppe $\mathbb{F}_p^\times$ lösen, also sei g ein Erzeuger dieser zyklischen Gruppe und

$$h = g^k$$

ein beliebiges Element in $\mathbb{F}_p^\times$. Gesucht ist wie immer der Wert

$$k \bmod (p-1).$$

Wir wählen eine Schranke t und bestimmen die Menge $S = \{p_1, \ldots, p_t\}$ der ersten t Primzahlen. In einem ersten Schritt, der für festes G, g und S nur einmal durchgeführt werden muß, werden die diskreten Logarithmen der Restklassen

$$\overline{p}_i = p_i \bmod p$$

bezüglich des erzeugenden Elementes g bestimmt. Wir schreiben hier

$$l = \log_g p_i, \text{ falls } \overline{p}_i = g^l \text{ in } \mathbb{F}_p^\times$$

ist. Dazu versucht man, lineare Abhängigkeiten zwischen den $\log_g p_i$ zu finden, so daß man diese diskreten Logarithmen durch Lösen eines linearen Gleichungssystems bestimmen kann. Hierfür wählt man ein $l \in \{1, \ldots, p-1\}$ und berechnet g^l. Wir bezeichnen der Einfachheit halber mit g^l auch den zugehörigen Vertreter in $\{0, \ldots, p-1\}$. Diesen versucht man nun, allein durch Primfaktoren in S zu faktorisieren. Falls dies gelingt, so ist

$$g^l = \prod_{i=1}^{t} p_i^{e_i}$$

mit gewissen Exponenten $e_i \geq 0$.

Wir gehen nun wieder zu Restklassen modulo p über und erhalten folgende lineare Gleichung:

$$l \equiv \sum_{i=1}^{t} e_i \log_g p_i \bmod (p-1).$$

Dies wiederholen wir so lange für verschiedene l, bis wir ein Gleichungssystem zusammengesammelt haben, das eine eindeutige Lösung besitzt. So können wir die $\log_g p_i$ berechnen.

Nach dieser Vorbereitung können wir nun den diskreten Logarithmus eines beliebigen $h = g^k$ bestimmen, indem wir ein $l \in \{1, \ldots, p-1\}$ wählen und versuchen, den Vertreter von hg^l in $\{0, \ldots, p-1\}$ alleine durch Primfaktoren in S zu faktorisieren. Gelingt dies, so gilt in der Gruppe $\mathbb{F}_p^\times$

$$hg^l = \prod_{i=1}^{t} \overline{p}_i^{m_i} = \prod_{i=1}^{t} g^{(\log_g p_i) m_i}$$

mit gewissen $m_i \geq 0$.

Daraus folgt

$$k + l \equiv \sum_{i=1}^{t} m_i (\log_g p_i) \bmod (p-1),$$

womit wir k berechnen können, da wir die $\log_g p_i$ schon kennen.

Damit dieser Algorithmus funktioniert, muß die Faktorbasis S so gewählt werden, daß einerseits "genug" Elemente in G eine Primfaktorzerlegung haben, in der nur Elemente aus S vorkommen, und sich andererseits diese Zerlegung effektiv berechnen läßt. Dies ist in der Tat so möglich, daß der Indexkalkül-Algorithmus subexponentielle Laufzeit hat. Außerdem läßt sich die Idee, zunächst die diskreten Logarithmen einer geeigneten Faktorbasis zu bestimmen, auch anwenden, um einen subexponentiellen Algorithmus für die Gruppen $\mathbb{F}_{2^m}^{\times}$, $m \geq 1$, und (unter gewissen plausiblen Annahmen) für allgemeine multiplikative Gruppen $\mathbb{F}_q^{\times}$ endlicher Körper zu entwickeln. Für eine Übersicht über diese Verfahren und für Hinweise auf die Originalliteratur siehe [Hb], S. 128f, und [Le-Le], Abschnitt 3.

Bisher ist es nicht gelungen, ein Analogon zum Indexkalkül-Algorithmus zu entwickeln, um das DL-Problem auf einer elliptischen Kurve anzugreifen. Es gibt sogar theoretische Argumente, die gegen den Erfolg eines solchen Projektes sprechen, siehe [Ko], Abschnitt 5.

5.3 ECDSA

ECDSA steht für "Elliptic Curve Digital Signature Algorithm", es handelt sich hierbei also um ein Verfahren für digitale Signaturen mit elliptischen Kurven. In den letzten Jahren ist es von verschiedenen Institutionen standardisiert worden. Wir wollen hier kurz einige Grundzüge dieser Standards erläutern. Eine ausführliche Beschreibung findet sich in [JMV].

Zunächst müssen die Ausgangsparameter festgelegt werden, die angeben, auf welcher elliptischen Kurve gearbeitet werden soll. Genauer gesagt, handelt es sich bei den Ausgangsparametern um ein Tupel

$$(q, a, b, x, y, n, h)$$

mit folgenden Eigenschaften:

- q ist entweder eine ungerade Primzahl $q = p$ oder eine Zweierpotenz $q = 2^m$ und gibt den Grundkörper $\mathbb{F}_q$ an.
- a und b sind Elemente in $\mathbb{F}_q$, die die affine Weierstraßgleichung der zugrundeliegenden elliptischen Kurve festlegen:

  Im Fall $q = p > 2$ ist $E(\mathbb{F}_q)$ gegeben durch

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

  und im Fall $q = 2^m$ durch

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

  Nach 2.3.2 können wir für $q = p > 3$ immer annehmen, daß $E(\mathbb{F}_q)$ durch eine Weierstraßgleichung dieser Form gegeben wird. (Der Fall $p = 3$ ist für praktische Zwecke uninteressant, da $E(\mathbb{F}_3)$ viel zu wenig Elemente hat, nach dem Satz von Hasse nämlich höchstens 7.)

  Falls $q = 2^m$ und $E(\mathbb{F}_q)$ nicht supersingulär ist, so können wir nach 2.3.2 und 3.4.4 ebenfalls annehmen, daß die Weierstraßgleichung von der oben angegebenen Form ist. Dies ist ausreichend, da supersinguläre Kurven für kryptographische Zwecke ohnehin ungeeignet sind (siehe 4.2.1).
- x und y sind Elemente in $\mathbb{F}_q$, die die affinen Koordinaten eines Punktes $P = (x, y)$ in $E(\mathbb{F}_q)$ angeben.
- n sei die Ordnung dieses Punktes P. Wir verlangen außerdem, daß n eine Primzahl ist und den Abschätzungen

  $$n > 2^{160} \text{ und } n > 4\sqrt{q}$$

  genügt. Darüberhinaus soll n für alle $k = 1, 2, \ldots, 20$ kein Teiler von $q^k - 1$ und ungleich q sein.
- Die Zahl h sei der sogenannte Kofaktor

  $$h = \frac{\#E(\mathbb{F}_q)}{n}.$$

Die Bedingung $n > 2^{160}$ sorgt hier dafür, daß das DL-Problem in der Untergruppe $\langle P \rangle$ nicht mit dem Pollard-ρ-Verfahren angreifbar ist. Da n kein Teiler von $(q^k - 1)$ ist für genügend viele k, kann man den MOV-Algorithmus nicht einsetzen, und da $n \neq q$ ist (das ist natürlich

nur eine Bedingung für $q = p > 2$), greift auch der SSSA-Algorithmus nicht.

Die Abschätzung $n > 4\sqrt{q}$ rührt daher, daß wir letztendlich nur die Untergruppe $\langle P \rangle$ von $E(\mathbb{F}_q)$ in kryptographischen Verfahren benutzen, unsere Berechnungen (wie etwa von kP) aber in $E(\mathbb{F}_q)$, d.h. in affinen Koordinaten in $\mathbb{F}_q \times \mathbb{F}_q$ durchführen müssen. Dies erfordert um so mehr Speicherplatz und Rechenzeit, je größer q ist. Auf der anderen Seite soll n natürlich für festes q möglichst groß sein, um das DL-Problem in $\langle P \rangle$ so schwer wie möglich zu machen. Die Bedingung $n > 4\sqrt{q}$ sorgt dafür, daß dies der Fall ist. Der Kofaktor h ist somit "klein", wir können ihn mit dem Satz von Hasse abschätzen durch

$$h < \frac{q + 1 + 2\sqrt{q}}{4\sqrt{q}}.$$

Man kann zu diesen Parametern auch noch einen Eintrag hinzunehmen, der spezifiziert, wie Elemente in $\mathbb{F}_q$ dargestellt werden sollen. Falls $q = p > 2$ ist, so nimmt man üblicherweise das Vertretersystem $\{0, 1, \ldots, p-1\}$ von $\mathbb{F}_p$. Falls $q = 2^m$ ist, so kann man $\mathbb{F}_{2^m}$ mit Bitstrings der Länge m identifizieren, wenn man z.B. das zugehörige irreduzible Polynom kennt (siehe 6.6). Dieses kann den Parametern hinzugefügt werden.

Die Kurvenparameter a und b können entweder speziell so gewählt werden, daß sich Berechnungen auf der elliptischen Kurve $E(\mathbb{F}_q)$ einfach durchführen lassen, oder sie können zufällig erzeugt werden. Letzteres läßt sich sogar so durchführen, daß eine andere Instanz nachprüfen kann, ob a und b wirklich zufällig gewählt sind. Wird das gewünscht, so kann man den Parametern noch entsprechende Daten hinzufügen, mit denen dies möglich ist. (Für weitere Einzelheiten sei auf [JMV] verwiesen.)

Zufällig gewählte elliptische Kurven sollten resistent sein gegenüber zukünftig vielleicht entwickelten Algorithmen, die das DL-Problem auf Kurven mit besonderen Eigenschaften lösen (so wie heute MOV oder SSSA). Wenn die zufällige Erzeugung nachprüfbar ist, so kann ausgeschlossen werden, daß ein Betrüger spezielle schwache Kurven einschleust, um die privaten Schlüssel zu erbeuten.

Auf der Basis dieser Ausgangsparameter wird nun für jeden Nutzer A zufällig eine Zahl

$$d \in \{1, 2, \ldots, n-1\}$$

gewählt und der Punkt

$$Q = dP \text{ in } E(\mathbb{F}_q)$$

berechnet. Die Zahl d ist dann As privater Schlüssel, der Punkt Q ist As öffentlicher Schlüssel.

Um die Nachricht m zu unterschreiben, die als Bitstring gegeben sei, geht A alias Alice nun folgendermaßen vor:

1) Sie wählt zufällig eine Zahl $k \in \{1, 2, \ldots, n-1\}$.

2) Sie berechnet $kP = (x, y)$ und konvertiert die erste Koordinate x folgendermaßen in eine ganze Zahl x':

   Wenn $q = p > 2$ ist, so ist $x \in \mathbb{F}_p$, und x' ist einfach der Vertreter von x in $\{0, 1, \ldots, p-1\}$.

   Wenn $q = 2^m$ ist, so ist $x \in \mathbb{F}_{2^m}$ gegeben durch einen Vektor $(a_{m-1}, \ldots, a_0)$ über $\mathbb{F}_2$ der Länge m. Hier sei x' die Zahl

$$x' = \sum_{i=0}^{m-1} a_i 2^i.$$

3) Sie berechnet die Restklasse

$$r = x' \bmod n \text{ in } \mathbb{Z}/n\mathbb{Z}.$$

   Falls $r = 0$ ist, so beginnt sie erneut bei 1). (Würde man hier trotzdem die Schritte 4) - 7) ausführen, so käme Alice' privater Schlüssel d in der Unterschrift nämlich gar nicht vor.)

4) Ansonsten berechnet sie k^{-1} in $\mathbb{Z}/n\mathbb{Z}$. (Da n eine Primzahl ist, ist k invertierbar modulo n.)

5) Dann berechnet sie $h(m)$ für eine vorher festgelegte Hashfunktion h (z.B. die Funktion h = SHA-1, siehe [Hb], §9.4), die ihre Werte in der Menge $\{0,1\}^N$ annimmt. Indem man den Bitstring

$$h(m) = (a_0, \ldots, a_{N-1})$$

   als die Binärentwicklung einer ganzen Zahl auffasst, erhält man

$$e = \sum_{i=0}^{N-1} 2^i a_{N-1-i}.$$

6) Nun berechnet sie die Restklasse

$$s = k^{-1}(e + rd) \bmod n$$

mit Hilfe ihres privaten Schlüssels d. Falls $s = 0$, also nicht invertierbar in $\mathbb{Z}/n\mathbb{Z}$ ist, so beginnt sie erneut bei 1). (Die Invertierbarkeit von s wird bei der Verifikation der Unterschrift gebraucht.)

7) Falls nicht, so ist Alice' Unterschrift für m das Paar (r, s).

Um diese Unterschrift zu prüfen, muß der Nutzer B alias Bob zunächst die Ausgangsparameter (q, a, b, x, y, n, h) und Alice' öffentlichen Schlüssel Q erhalten. Hier muß Bob allerdings sicher sein, daß er wirklich Alice' öffentlichen Schlüssel bekommt. Gelingt es nämlich der Betrügerin Eva, ihm ihren öffentlichen Schlüssel als Alice' Schlüssel unterzuschieben, so kann Eva in Alice' Namen gültige Unterschriften erzeugen.

Dieses Problem läßt sich dadurch umgehen, daß sich jeder Teilnehmer bei einer vertrauenswürdigen Instanz, einer sogenannten "Certification Authority" (abgekürzt CA) registrieren läßt. Von dieser erhält Bob dann quasi eine beglaubigte Kopie von Alice öffentlichem Schlüssel. Wie das genauer funktioniert, kann man in [Bu], Abschnitt 14.2 nachlesen.

Um Alice' Unterschrift (r, s) unter m zu prüfen, geht Bob nun folgendermaßen vor:

1) Er prüft nach, ob r und s in der Menge $\{1, 2, \ldots, n-1\}$ liegen.

2) Er berechnet den Hashwert $h(m)$ und daraus die Zahl e.

3) Dann berechnet er das Inverse $w = s^{-1}$ in $\mathbb{Z}/n\mathbb{Z}$ sowie den Punkt

$$R = w(eP + rQ)$$

in $E(\mathbb{F}_q)$.

4) Falls $R = O$ ist, so ist die Unterschrift ungültig. Andernfalls ist $R = (x_1, y_1)$ ein affiner Punkt auf $E(\mathbb{F}_q)$. Bob wandelt die erste Koordinate x_1 wie oben beschrieben in eine ganze Zahl x_1' um.

5) Falls $(x_1' \bmod n) = r$ ist, so akzeptiert er Alice' Unterschrift, andernfalls nicht.

Falls die Unterschrift wirklich von Alice stammt, so ist

$$s = k^{-1}(e + rd) \bmod n,$$

also gilt $wk^{-1}(e + rd) \equiv 1 \bmod n$, somit auch $w(e + rd) \equiv k \bmod n$. Daher folgt

$$R = w(eP + rQ) = w(e + rd)P = kP,$$

so daß die x-Koordinaten beider Punkte und damit auch x_1' mod n und r gleich sind.

Damit Bob Alice' Unterschrift akzeptiert, muß umgekehrt das Bild des Punktes $R = w(eP + rQ)$ unter der Abbildung

$$\begin{aligned} \psi : E(\mathbb{F}_q)\backslash\{O\} &\longrightarrow \mathbb{Z}/n\mathbb{Z} \\ (x_1, y_1) &\longmapsto x_1' \bmod n \end{aligned}$$

gleich r sein, wobei x_1' die ganze Zahl zu $x_1 \in \mathbb{F}_q$ ist (für $q = p$ also der Vertreter in $\{0, 1, \ldots, p-1\}$ und für $q = 2^m$ die Zahl, deren Binärentwicklung durch x_1 gegeben ist).

Diese Abbildung ψ ist zwar nicht bijektiv (so haben ja z.B. die Punkte P und $-P$ dieselbe x-Koordinate), jedoch ist die Urbildmenge jedes Wertes klein genug, so daß es unwahrscheinlich ist, daß $\psi(R)$ und $r = \psi(kP)$ übereinstimmen, ohne daß $R = kP$ gilt.

Wenn $R = kP$ ist, so erhalten wir wie oben

$$w(e + rd) \equiv k \bmod n,$$

woraus $s \equiv k^{-1}(e + dr) \bmod n$ folgt, was nur Alice berechnet haben kann.