

Kapitel 1

Effiziente Arithmetik in \mathbb{Z}_M und \mathbb{F}_{p^n}

Die Effizienz eines kryptographischen Verfahrens wird in großem Umfang durch die Implementierung der arithmetischen Operationen bestimmt. Das beste, bekannte Verfahren für die Arithmetik modulo M ist die Montgomery-Darstellung, bei der die Modulo-Reduktion ohne aufwendige Division mit Rest möglich ist. Für endliche Körper lernen wir das Konzept der (optimalen) Normalbasen kennen, die zum Beispiel in \mathbb{F}_{2^n} Quadrieren ohne Multiplikation ermöglicht.¹

1.1 Langzahl-Arithmetik

Die meisten Prozessoren verfügen heute über 32- oder 64-Bit-Register und Instruktionen, um den Inhalt von Registern zu addieren, subtrahieren, multiplizieren oder dividieren. Bei der Multiplikation werden die Faktoren auf die halbe Bitlänge beschränkt, damit das Produkt in einem Register abgelegt werden kann. Man nennt die halbe Bitlänge die sog. *Wortlänge* des Prozessors.²

Betrachtet man den Zeitaufwand der CPU-Instruktionen für arithmetische Operationen, so ist allgemein eine Multiplikation deutlich aufwendiger als eine Addition oder Subtraktion, und eine Division (mit Rest) ist noch zeitintensiver als eine Multiplikation. Prozessoren verfügen über spezielle Befehle, um in einem Register die Bits nach links oder rechts zu verschieben sowie zur bitweisen UND-Verknüpfung. Durch diese Instruktionen sind

- Multiplikation mit 2^i ,
- Division durch 2^i (ohne Rest) und

¹Beweise oder Teile von Beweisen, die nicht in der Vorlesung geführt wurden, sind durch eine kleine Schrift markiert und können übersprungen werden.

²In der Literatur wird der Begriff teilweise für die Bitlänge der Register verwendet.

- Reduktionen modulo 2^i

effizient zu realisieren.

Für kryptographische Anwendungen benötigt man größere Zahlen als 2^{64} , typischerweise Bitlängen zwischen 160 und 2048. Programmpakete für die *Langzahl-Arithmetik* (multi-precision arithmetic), wie zum Beispiel GMP [gmp] oder das Java-Package JAVA.MATH [sun], stellen eine Zahl $x \in \mathbb{N}_0$ zur Basis $b := 2^w$ als

$$x = \sum_{i=0}^n x_i b^i$$

mit $x_i \in [0, b)$ dar. Die arithmetischen Operationen auf Zahlen mit wn Bits werden auf mehrere der Bitlänge w zurückgeführt. Üblicherweise ist w die Wortlänge des Prozessors.

1.2 Montgomery-Darstellung

Die Restklassen modulo M repräsentiert man im allgemeinen jeweils durch das kleinste, nicht-negative Residuum, also $\mathbb{Z}_M := [0, M)$. Beim klassischen Ansatz der modularen Arithmetik werden die Repräsentanten x, y addiert, subtrahiert oder multipliziert und anschließend das Ergebnis modulo M reduziert. Bei der Addition gestaltet sich die Modulo-Reduktion einfach, denn wegen

$$0 \leq x + y < 2M$$

genügt es, M zu subtrahieren, sofern die Summe $x + y \geq M$ ist. Für die Subtraktion ist die Situation ähnlich, bei der Multiplikation gelten hingegen die Schranken

$$0 \leq xy < (M - 1)^2 + 1$$

und der Repräsentant wird mittels (zeitintensiver) Division mit Rest bestimmt.

1.2.1 Montgomery-Reduktion und -Multiplikation

Eine effizientere Methode für die modulare Arithmetik geht auf P. MONTGOMERY [M85] zurück. Zum Modul M wähle eine teilerfremde Zahl $R > M$, so dass der Zeitaufwand für

- die Division durch R und
- die Reduktion modulo R

vernachlässigbar ist. Für die bekannten kryptographischen Anwendungen mit ℓ -Bit-Modul M bietet sich $R := 2^\ell$ an, da der Modul als große Primzahl oder das Produkt zweier großer Primzahlen ungerade ist. Bei der Betrachtung der Algorithmen analysieren wir nur die Anzahl der (zeitintensiven) Multiplikationen, während Additionen modulo M , Division durch R und Reduktionen modulo R unberücksichtigt bleiben.

In der Montgomery-Darstellung wird x modulo M nicht durch $x \bmod M$, sondern durch

$$[x]_R := xR \bmod M$$

repräsentiert. Die Abbildung $x \mapsto [x]_R$ ist verträglich mit der Addition, denn für alle $x, y \in \mathbb{Z}_M$ gilt

$$[x + y]_R \equiv (x + y)R \equiv xR + yR \equiv [x]_R + [y]_R,$$

jedoch unverträglich mit der Multiplikation: $[xy]_R \equiv [x]_R \cdot [y]_R \cdot R^{-1}$.

Definition 1.2.1 (Montgomery-Multiplikation und -Reduktion). *Bezüglich R ist die Montgomery-Multiplikation erklärt als*

$$u \star v := uvR^{-1} \bmod M$$

und die Montgomery-Reduktion als $\text{Mred}_R(u) := uR^{-1} \bmod M$.

Wir nennen $u \star v$ das Montgomery-Produkt von u und v . Die Montgomery-Multiplikation ist die Komposition von Multiplikation (über \mathbb{Z}) mit anschließender Montgomery-Reduktion. Die Montgomery-Reduktion ist die inverse Abbildung zu $x \mapsto [x]_R$ auf \mathbb{Z}_M . Bezogen auf die Montgomery-Multiplikation gilt:

- $R = [1]_R$ ist das Einselement, denn $u \star R \equiv uRR^{-1} \equiv u$.
- Das Inverse von u ist $u^{-1\star} := u^{-1}R^2$ (*Montgomery-Inverses*):

$$u \star u^{-1\star} \equiv u \star u^{-1}R^2 \equiv uu^{-1}R^2R^{-1} \equiv R.$$

Die Abbildung $x \mapsto [x]_R$ ist ein Homomorphismus von (\mathbb{Z}_M^*, \cdot) nach (\mathbb{Z}_M^*, \star) , denn $[1]_R$ ist das Einselement der Montgomery-Multiplikation und für alle $x, y \in \mathbb{Z}_M$ gilt:

$$[xy]_R \equiv (xy)R \equiv (xR \cdot yR)R^{-1} \equiv [x]_R \star [y]_R.$$

Zusammenfassend ist $(\mathbb{Z}_M, +, \star)$ ein kommutativer Ring:

Satz 1.2.2. *Es gilt $(\mathbb{Z}_M, +, \cdot) \simeq (\mathbb{Z}_M, +, \star)$, der Isomorphismus ist gegeben durch $x \mapsto [x]_R$.*

Insbesondere ist die Montgomery-Multiplikation assoziativ, so dass wir analog zur Exponentiation modulo M das k -fache Montgomery-Produkt $u^{k\star}$ von u definieren. Sei $u^{0\star} := R$ und für $k \in \mathbb{N}$:

$$u^{k\star} := \underbrace{u \star u \star \cdots \star u}_{k\text{-Faktoren}}$$

Entsprechend der üblichen Methode bei der Exponentiation $k \mapsto g^k \bmod M$ ist $u^{k\star}$ mit $2 \cdot \lceil \log_2 k \rceil$ Montgomery-Multiplikation zu berechnen [MOV97, 14.6.1].

Aus Satz 1.2.2 erhalten wir folgende Möglichkeit, die modulare Exponentiation $k \mapsto g^k \bmod M$ zu berechnen (vergleiche Abbildung 1.1). Zuerst bestimme $u := [g]_R$ und berechne $u^{k\star}$. Nach Satz 1.2.2 ist $u^{k\star} = [g^k]_R$, und eine Montgomery-Reduktion ergibt $g^k \equiv \text{Mred}_R(u^{k\star})$.

$$\begin{array}{ccc} g & \xrightarrow{\text{Multiplikationen mod } M} & g^k \\ u=[g]_R \downarrow & & \uparrow g^k \equiv \text{Mred}_R(u^{k\star}) \\ u & \xrightarrow{\text{Montgomery-Multiplikationen } \star} & u^{k\star} \end{array}$$

Abbildung 1.1: Exponentiation mit Hilfe der Montgomery-Darstellung

Damit diese Transformation zur Berechnung von $k \mapsto g^k$ sinnvoll ist, müssen die Montgomery-Produkte schneller als die modularen Produkte zu berechnen sein. Bei der modularen Multiplikation multipliziert man beide Repräsentanten und reduziert das Produkt z mittels Division mit Rest. Bei der Montgomery-Multiplikation tritt an die Stelle der Modulo-Reduktion die Montgomery-Reduktion $\text{Mred}_R(z)$, die für

$$0 \leq z \leq (M-1)^2 < MR$$

einfach zu bewerkstelligen ist:

Satz 1.2.3 (Montgomery 1985). *Sei $R > M$ teilerfremd zu M und $M' := -M^{-1} \bmod R$. Sei z mit $0 \leq z < MR$. Für $t := zM' \bmod R$ gilt dann, dass $\frac{z+tM}{R}$ ganzzahlig ist,*

$$0 \leq \frac{z+tM}{R} < 2M \tag{1.1}$$

und

$$\frac{z+tM}{R} \equiv zR^{-1} \pmod{M}. \tag{1.2}$$

Beweis. $z + tM$ ist ein Vielfaches von R , denn

$$z + tM \equiv z + zM'M \equiv z + z(-M^{-1})M \equiv z - z \equiv 0 \pmod{R}.$$

Infolge der Voraussetzungen $0 \leq z < MR$ und $0 \leq t < R$ gilt

$$0 \leq z + tM < MR + RM < 2MR,$$

und wir erhalten die Abschätzung (1.1). Aus $z + tM \equiv z \pmod{M}$ folgt nach Multiplikation mit R^{-1} Kongruenz (1.2). \square

Montgomery-Reduktion von z mit $0 \leq z < MR$

```

/* als Preprocessing bestimme  $M' := -M^{-1} \pmod{R}$ . */
1.  $t := zM' \pmod{R}$ 
2.  $z := (z + tM)/R$ 
3. IF  $z \geq M$  THEN  $z := z - M$ 
4. Ausgabe  $z$ .

```

Abbildung 1.2: Montgomery-Reduktion

Der Wert $M' := -M^{-1} \pmod{R}$ hängt nicht von z ab und ist bereits bei der Wahl von R zu berechnen (*Preprocessing*). Algorithmus 1.2 zeigt die Montgomery-Reduktion von z . Nach Wahl von R entspricht die Laufzeit der Montgomery-Reduktion $\text{Mred}_R(z)$ weitgehend der der beiden Multiplikationen zM' und tM mit $z \in [0, RM)$ und $t, M' \in [0, R)$.

Die Montgomery-Reduktion verwenden wir ebenfalls, um die Funktion $x \mapsto [x]_R$ zu berechnen:

$$[x]_R \equiv xR \equiv xR^2R^{-1} \equiv \text{Mred}_R(xR^2),$$

wobei der Faktor $(R^2 \pmod{M}) \in [0, M)$ Teil des Preprocessings ist. Für die Berechnung des Montgomery-Inversen $u^{-1\star}$ von u bestimmen wir $u^{-1} \pmod{M}$, multiplizieren das Resultat mit $(R^3 \pmod{M}) \in [0, M)$ und wenden die Montgomery-Reduktion an:

$$u^{-1\star} \equiv u^{-1}R^2 \equiv u^{-1}R^3R^{-1} \equiv \text{Mred}_R(u^{-1}R^3).$$

Zusammenfassend:

Korollar 1.2.4. *Bei Preprocessing sind*

- die Montgomery-Reduktion mit 2 Multiplikationen,

- $x \mapsto [x]_R$ mit 3 Multiplikationen und
- die Montgomery-Multiplikation mit 3 Multiplikationen

von Werten aus $[0, MR)$ zu bestimmen. Die Berechnung eines Montgomery-Inversen ist mit einer Invertierung modulo M und 3 zusätzlichen Multiplikationen von Werten aus $[0, MR)$ möglich.

Für den wichtigen Spezialfall, dass R eine Zweier-Potenz ist, stammt von B.S. KALISKI [K95] ein schnelles Verfahren, das basierend auf dem binären ggT-Algorithmus Montgomery-Inverse berechnet. Der interessierte Leser sei auf die Originalarbeit verwiesen.

Aus Korollar 1.2.4 erhalten wir für das Beispiel der modularen Exponentiation:

Korollar 1.2.5. *Sei M ein ungerader, ℓ -Bit-Modul. Dann kann man die Exponentiation $g^k \bmod M$ mit $5 + 6 \cdot \lceil \log_2 k \rceil$ Multiplikationen von $(2\ell + 1)$ -Bit-Zahlen berechnen (modulare Additionen nicht mitgezählt).*

Montgomery-Reduktion von $z = \sum_{i=0}^{n-1} z_i b^i$ mit $0 \leq z < MR$

```

/* als Preprocessing bestimme  $m' := -M^{-1} \bmod b$ . */
1. FOR  $i = 0$  TO  $n - 1$  DO
    (a)  $t_i := z_i m' \bmod b$ .
    (b)  $z := z + t_i M b^i$ .
    END for
2.  $z := z/R$ .
3. IF  $z \geq M$  THEN  $z := z - M$ .
4. Ausgabe  $z$ .

```

Abbildung 1.3: Montgomery-Reduktion bei Langzahl-Arithmetik

1.2.2 Montgomery-Reduktion bei Langzahl-Arithmetik

Wir betrachten die Montgomery-Reduktion für Langzahl-Arithmetik zur Basis $b = 2^w$, der (ungerade) Modul habe die Darstellung $M = \sum_{i=0}^{n-1} m_i b^i$. Als R wähle $R = b^n$, da der Reduktionsalgorithmus mit $-M^{-1}$ modulo b statt modulo R operieren kann. Der Aufwand für Multiplikation mit b , Divisionen durch b , Reduktion modulo b und Additionen modulo M sei vernachlässigbar. Dann gilt:

Satz 1.2.6. Zur Eingabe z mit $0 \leq z < MR$ berechnet Algorithmus 1.3 die Montgomery-Reduktion $\text{Mred}_R(z)$ mit $n(n+1)$ Multiplikationen von Faktoren aus $[0, b)$.

Beweis. Sei \bar{z} die Eingabe. Aus den Schleifeninvarianten

- $0 \leq Rz < RM + \sum_{j=0}^{i-1} b^j M$,
- b^i teilt z (äquivalent $z_0 = z_1 = \dots = z_{i-1} = 0$) und
- $z \equiv \bar{z} \pmod{M}$.

folgt, dass nach Schritt 1 dann R ein Teiler von $z \equiv \bar{z} \pmod{M}$ ist. Ferner gilt wegen $b^n = R$:

$$0 \leq Rz < RM + \sum_{i=0}^{n-1} b^i M = RM + (b^n - 1)M < 2RM.$$

In Schritt 4 gibt der Algorithmus $z = \text{Mred}_R(\bar{z}) = \bar{z}R^{-1} \pmod{M}$ aus. \square

Um das Montgomery-Produkt $u \star v$ zu berechnen, fassen wir Langzahl-Multiplikation $z := uv$ und die Montgomery-Reduktion von z zusammen. Wegen

$$uv = \left(\sum_{i=0}^{n-1} u_i b^i \right) v = \sum_{i=0}^{n-1} u_i b^i v$$

können wir das Produkt $z := uv$ wie folgt berechnen:

1. $z := 0$
2. FOR $i = 0$ TO $n - 1$ DO $z := z + u_i b^i v$.

Die Werte z_0, z_1, \dots, z_i bleiben nach der Iteration i unverändert. Bei der Montgomery-Reduktion in Algorithmus 1.3 hängt der Wert t_i nur von z_i ab. Bei der Montgomery-Reduktion von uv können wir t_i bereits nach der Zuweisung $z := z + u_i b^i v$ bestimmen. Auf diese Weise lassen sich die Schleifen der Multiplikation und der Montgomery-Reduktion zusammenfassen:

1. $z := 0$.
2. FOR $i = 0$ TO $n - 1$ DO
 - (a) $z := z + u_i b^i v$.
 - (b) $t_i := z_i m' \pmod{b}$.
 - (c) $z := z + t_i M b^i$.

END for

Bei genauerer Betrachtung ist zu erkennen, dass man den Schleifenrumpf durch zwei Anweisungen wie in Algorithmus 1.3 ersetzen kann:

Satz 1.2.7. *Zur Eingabe u, v mit $0 \leq u, v < M$ berechnet Algorithmus 1.3 das Montgomery-Produkt $z := u \star v$ mit $2n(n+1)$ Multiplikationen von Faktoren aus $[0, b)$.*

Montgomery-Multiplikation $z := u \star v$

/ als Preprocessing bestimme $m' := -M^{-1} \bmod b$. */*

1. $z := 0$.
 2. FOR $i = 0$ TO $n - 1$ DO
 - (a) $t_i := (z_i + u_i v) m' \bmod b$
 - (b) $z := z + (u_i v + t_i M) b^i$
 - END for
 3. $z := z/R$.
 4. IF $z \geq M$ THEN $z := z - M$.
 5. Ausgabe z .
-

Abbildung 1.4: Montgomery-Multiplikation bei Langzahl-Arithmetik

1.3 Optimale Normalbasen

Für effiziente Implementierungen von kryptographischen Verfahren, die auf dem diskreten Logarithmus basieren, sind elliptische Kurven über endlichen Körpern \mathbb{F}_{p^n} , speziell \mathbb{F}_{2^n} , eine Alternative zu Untergruppen von \mathbb{F}_p [BSS99]. Bei der Darstellung von \mathbb{F}_{2^n} durch eine Normalbasis ist Quadrieren ohne Multiplikation möglich, für optimale Normalbasen ist zusätzlich bei der Multiplikation die Anzahl von Operationen über \mathbb{F}_2 minimal. Teile dieser Konstruktionen sind jedoch durch Patente geschützt [MOV97].

1.3.1 Normalbasen

Der endliche Körper \mathbb{F}_{p^n} mit p^n Elementen ist ein n -dimensionaler Vektorraum über seinem Grundkörper \mathbb{F}_p . Es existiert eine Basis \mathcal{B} bestehend aus

n Elementen $\beta_0, \beta_1, \dots, \beta_{n-1} \in \mathbb{F}_{p^n}$, so dass jedes $A \in \mathbb{F}_{p^n}$ eine Linearkombination der β_i mit Koeffizienten a_0, a_1, \dots, a_{n-1} aus \mathbb{F}_p ist:

$$A = \sum_{i=0}^{n-1} a_i \beta_i.$$

$A \in \mathbb{F}_{p^n}$ wird durch seinen Koeffizienten-Vektor $\mathbf{a} := (a_0, a_1, \dots, a_{n-1}) \in \mathbb{F}_p^n$ bezüglich der gewählten Basis \mathcal{B} dargestellt.³ Addition und Subtraktion sind elementar: $A \pm B$ von $A, B \in \mathbb{F}_{p^n}$ entspricht der Summe bzw. Differenz der Koeffizientenvektoren, also $\mathbf{a} \pm \mathbf{b}$. Die Multiplikation ist aufwendiger:

$$C := A \cdot B = \sum_{k=0}^{n-1} c_k \beta_k = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j \beta_i \beta_j$$

Sei T_k die $n \times n$ -Matrix $[t_{i,j}^{(k)}]_{0 \leq i,j < n}$ über dem Grundkörper \mathbb{F}_p mit:

$$\beta_i \beta_j = \sum_{k=0}^{n-1} t_{i,j}^{(k)} \beta_k. \quad (1.3)$$

Für die Einträge des Koeffizientenvektors \mathbf{c} gilt:

$$c_k := c_k(A, B) = \sum_{i,j} a_i b_j t_{i,j}^{(k)} = \mathbf{a} \cdot T_k \cdot \mathbf{b}^T \quad (1.4)$$

Die n Matrizen T_0, T_1, \dots, T_{n-1} hängen nicht von den Faktoren A und B ab. Sie sind die sog. *Multiplikationstabellen* bezüglich der Basis \mathcal{B} . Obwohl man die Multiplikationstabellen vorab (Preprocessing) bestimmen kann, ist der Ansatz mit n^3 zu speichernden Werten aus \mathbb{F}_p nicht praktikabel. Wir werden sehen, dass für Normalbasen die Darstellung der Multiplikation durch die Multiplikationstabelle T_0 bzw. durch einen Schaltkreis für die Berechnung des Koeffizienten c_0 eines Produktes ausreichend ist.

Definition 1.3.1 (Normalbasis). Eine Basis $\beta_0, \beta_1, \dots, \beta_{n-1} \in \mathbb{F}_{p^n}$ von \mathbb{F}_{p^n} über \mathbb{F}_p heißt *Normalbasis*, wenn $\beta_i = \beta^{p^i}$ für ein $\beta \in \mathbb{F}_{p^n}$. Man sagt, β erzeugt die Normalbasis.⁴

Die Potenzen $\beta^p, \beta^{p^2}, \dots, \beta^{p^{n-1}}$ sind die sog. *Konjugierten* von β über \mathbb{F}_p . Es gilt [LN86, Theorem 2.35]:

Satz 1.3.2 (Normal-Basis-Theorem). Zu jedem endlichen Körper \mathbb{F}_{p^n} existiert eine Normalbasis über \mathbb{F}_p .

³Für ein irreduzibles Polynom $f(\beta) \in \mathbb{F}_p[\beta]$ vom Grad n ist \mathbb{F}_{p^n} isomorph zum Faktorring $\mathbb{F}_p[\beta]$ modulo $f(\beta)$ und $1, \beta, \beta^2, \dots, \beta^{n-1}$ bilden eine Basis, die sog. *Polynomialbasis*.

⁴Beachte den Unterschied: Erzeugt β eine Normalbasis, so bezieht sich dies auf p -Potenzen $\beta, \beta^p, \beta^{p^2}, \beta^{p^3}, \dots$ von β . Erzeugt α eine Gruppe, so bezieht sich dies auf die Potenzen $\alpha, \alpha^2, \alpha^3, \alpha^4, \dots$ von α .

Sei $\mathcal{N} := \{\beta_0, \beta_1, \dots, \beta_{n-1}\}$ eine von β erzeugte Normalbasis von \mathbb{F}_{p^n} über \mathbb{F}_p . Zur Vereinfachung der Notation seien Indizes im weiteren jeweils modulo n . Aufgrund $\beta^{p^n} = \beta$ gilt:

$$\beta_i^{p^k} = \beta_{i+k} \quad (1.5)$$

Aufgrund $(U + V)^p = U^p + V^p$ für $U, V \in \mathbb{F}_{p^n}$ und $w^p = w$ für $w \in \mathbb{F}_p$ gilt

$$A^p = \left(\sum_{i=0}^{n-1} a_i \beta_i \right)^p = \sum_{i=0}^{n-1} a_i^p \beta_i^p = \sum_{i=0}^{n-1} a_i \beta_{i+1} = \sum_{i=0}^{n-1} a_{i-1} \beta_i.$$

Bezogen auf den Koeffizientenvektor entspricht das Erheben in die p -te Potenz dem zyklischen Verschieben (Rotation) der Einträge

$$(a_0, a_1, \dots, a_{n-1}) \mapsto (a_{n-1}, a_0, a_1, \dots, a_{n-2}),$$

so dass $A \mapsto A^{p^k}$ und $A \mapsto A^{p^{-k}}$ effizient zu implementieren sind.

Betrachten wir die Multiplikationstabellen T_0, T_1, \dots, T_{n-1} bezogen auf die Normalbasis \mathcal{N} . Nach Gleichung (1.5) gilt

$$\beta_i \beta_j = \beta_{i-1}^p \beta_{j-1}^p = (\beta_{i-1} \beta_{j-1})^p$$

und mit Gleichung (1.3) folgt:

$$\sum_{k=0}^{n-1} t_{i,j}^{(k)} \beta_k = \left(\sum_{k=0}^{n-1} t_{i-1,j-1}^{(k)} \beta_k \right)^p = \sum_{k=0}^{n-1} t_{i-1,j-1}^{(k)} \beta_{k+1} = \sum_{k=0}^{n-1} t_{i-1,j-1}^{(k-1)} \beta_k.$$

Weil $\beta_0, \beta_1, \dots, \beta_{n-1}$ als Basis linear unabhängig sind, erhalten wir

$$t_{i,j}^{(k)} = t_{i-1,j-1}^{(k-1)}$$

durch einen Koeffizientenvergleich. Ein einfacher Induktionsbeweis nach k ergibt

$$t_{i,j}^{(k)} = t_{i-k,j-k}^{(0)},$$

so dass es für Normalbasen ausreichend ist, statt der n Multiplikationstabelle T_0, T_1, \dots, T_{n-1} lediglich die Matrix T_0 zu speichern oder wegen

$$c_k(A, B) = c_0(A^{p^{-k}}, B^{p^{-k}})$$

ein Verfahren zur Berechnung von c_0 zu implementieren.

1.3.2 Konstruktion optimaler Normalbasen

Man bezeichnet als *die* Multiplikationstabelle einer von β erzeugten Normalbasis von \mathbb{F}_{p^n} über \mathbb{F}_p die $n \times n$ -Matrix T erklärt durch

$$\beta \cdot \beta_i = \sum_{j=0}^{n-1} t_{ij} \beta_j. \quad (1.6)$$

Wegen $t_{ij} = t_{0,i}^{(j)} = t_{n-j,i-j}^{(0)}$ unterscheiden sich T und T_0 lediglich durch die Anordnung der Koeffizienten. Im weiteren werden wir sehen, dass für optimale Normalbasen T eine einfache Struktur aufweist.

Je mehr Nulleinträge T bzw. T_0 hat, desto einfacher ist die Multiplikation zu realisieren. Daher ist man an Normalbasen interessiert, deren Multiplikationstabelle T möglichst wenig Einträge ungleich Null aufweist. Deren Anzahl wird als *Komplexität* $c_{\mathcal{N}}$ der Normalbasis \mathcal{N} bezeichnet:

Satz 1.3.3. *Sei \mathcal{N} eine Normalbasis von \mathbb{F}_{p^n} über \mathbb{F}_p . Dann gilt $c_{\mathcal{N}} \geq 2n-1$.*

Beweis. Sei $\mathcal{N} := \{\beta_0, \beta_1, \dots, \beta_{n-1}\}$ eine von β erzeugte Normalbasis von \mathbb{F}_{p^n} über \mathbb{F}_p . Wir setzen w gleich der Spur von β über \mathbb{F}_p

$$w := \sum_{i=0}^{n-1} \beta^{p^i} = \sum_{i=0}^{n-1} \beta_i,$$

wobei aufgrund $w^p = w$ die Spur w in \mathbb{F}_p liegt. Summiere die Gleichungen (1.6) für $i = 0, 1, \dots, n-1$ auf:

$$\beta \cdot \underbrace{(\beta_0 + \beta_1 + \dots + \beta_{n-1})}_{=w} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} t_{ij} \beta_j = \sum_{j=0}^{n-1} \left(\sum_{i=0}^{n-1} t_{ij} \right) \beta_j.$$

Wir halten durch einen Koeffizientenvergleich:

$$\sum_{i=0}^{n-1} t_{ij} = \begin{cases} w & \text{falls } j = 0 \\ 0 & \text{sonst.} \end{cases} \quad (1.7)$$

Da $\beta \neq 0$, ist $\beta\beta_0, \beta\beta_1, \dots, \beta\beta_{n-1}$ ebenfalls eine Basis von \mathbb{F}_{p^n} und die Transformationsmatrix T invertierbar.

In jeder Spalte von T stehen mindestens zwei Einträge ungleich Null, d.h. zu jedem j gibt es zwei t_{ij} ungleich Null. Wir wissen aus (1.7), dass die Summe der Einträge in jeder bis auf Spalte $j = 0$ Null ist, also jeweils zwei Einträge nicht Null sind. In der Spalte $j = 0$, deren Einträge sich zu w summieren, muß in jedem Fall ein Eintrag ungleich Null sein. Mindestens $1 + 2(n-1) = 2n-1$ Einträge der Matrix T sind ungleich Null. \square

Die untere Schranke des Satzes 1.3.3 legt folgende Definition nah:

Definition 1.3.4 (Optimale Normalbasis ONB). *Eine Normalbasis \mathcal{N} von \mathbb{F}_{p^n} über \mathbb{F}_p heißt optimal, wenn $c_{\mathcal{N}} = 2n-1$.*

Aus dem Beweis der unteren Schranke für $c_{\mathcal{N}}$, Satz 1.3.3, ist abzuleiten, wie die Multiplikationstabelle T zu einer von β erzeugten optimalen Normalbase aussehen muß: Die erste Spalte enthält genau einen Eintrag (die

sog. Spur $\sum_{i=0}^{n-1} \beta^{p^i}$ von β), die übrigen Spalten jeweils zwei Einträge, deren Summe Null ist. Die übrigen Werte sind Null.

Man kennt zwei konstruktive Beweise für die Existenz von optimalen Normalbasen. Aufgrund der unterschiedlichen Multiplikationstabelle werden sie als Typ I oder II klassifiziert. Optimale Normalbasen des Typs II kennt man nur von \mathbb{F}_{2^n} . Die Vermutung, dass es keine weiteren Klassen von Normalbasen für endliche Körper gibt, wurde von GAO (und später mit LNSTRA in einem allgemeineren Kontext) [GL92] bewiesen.

Satz 1.3.5 (ONB Typ I). *Sei $n + 1$ prim und $\langle p \rangle = \mathbb{Z}_{n+1}^*$. Dann sind die nicht-trivialen, $(n + 1)$ -ten Einheitswurzeln (über \mathbb{F}_p) eine optimale Normalbasis von \mathbb{F}_{p^n} über \mathbb{F}_p .*

Beweis. Weil p kein Teiler von $n + 1$ ist, sind die $(n + 1)$ -ten Einheitswurzeln eine zyklische Gruppe der Ordnung $n + 1$. Aus $p^n \equiv 1 \pmod{n + 1}$ erhalten wir $n + 1 \mid p^n - 1$, so dass die $(n + 1)$ -ten Einheitswurzeln in \mathbb{F}_{p^n} liegen. Sei $\beta \in \mathbb{F}_{p^n}$ eine primitive, $(n + 1)$ -te Einheitswurzel. Die nicht-trivialen, $(n + 1)$ -ten Einheitswurzeln $\beta, \beta^2, \dots, \beta^n$ sind linear unabhängig über \mathbb{F}_p und bilden eine Basis von \mathbb{F}_{p^n} über \mathbb{F}_p .

Die n Elemente $\beta, \beta^p, \beta^{p^2}, \dots, \beta^{p^{n-1}}$ sind paarweise verschieden, sonst wäre $p^m \equiv 1 \pmod{n + 1}$ mit $1 \leq m < n$ im Widerspruch zur Voraussetzung $\langle p \rangle = \mathbb{Z}_{n+1}^*$. $\beta, \beta^2, \dots, \beta^n$ sind linear unabhängig über \mathbb{F}_p , denn durch Erheben von $\sum_{i=0}^n a_i \beta^i = 0$ mit $a_i \in \mathbb{F}_p$ in die Potenzen p, p^2, p^4, \dots , erhalten wir

$$\begin{bmatrix} \beta & \beta^2 & \dots & \beta^n \\ \beta^p & \beta^{2p} & \dots & \beta^{np} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{p^{n-1}} & \beta^{2p^{n-1}} & \dots & \beta^{np^{n-1}} \end{bmatrix} \cdot \mathbf{a} = \mathbf{0}$$

Da $\beta, \beta^p, \beta^{p^2}, \dots, \beta^{p^{n-1}}$ paarweise verschieden sind, hat die Vandermonde-Matrix vollen Rang und aus $\sum_{i=0}^n a_i \beta^i = 0$ folgt $\mathbf{a} = \mathbf{0}$. Angesichts der Dimension n des Vektorraums bilden $\beta, \beta^p, \beta^{p^2}, \dots, \beta^{p^{n-1}}$ eine Basis.

Infolge $\langle p \rangle = \mathbb{Z}_{n+1}^*$ durchlaufen Modulo $n + 1$ die p -Potenzen p, p^2, p^3, \dots die Werte $1, 2, \dots, n$:

$$\mathcal{N} := \{\beta, \beta^p, \beta^{p^2}, \dots, \beta^{p^{n-1}}\} = \{\beta, \beta^2, \dots, \beta^n\}, \quad (1.8)$$

Wir stellen die Multiplikationstabelle indiziert durch $\beta, \beta^2, \dots, \beta^n$ statt durch $\beta, \beta^p, \beta^{p^2}, \dots, \beta^{p^{n-1}}$ auf. Diese stimmt nach (1.8) bis auf Permutation der Zeilen und Spalten mit der Multiplikationstabelle T überein. Für die Multiplikationstabelle sind von $\beta^i \beta^j$ die Koeffizienten der Darstellung bezüglich der Basis $\beta, \beta^2, \dots, \beta^n$ zu bestimmen. Aufgrund $\beta^{n+1} = \beta^0 = 1$

und $\sum_{i=0}^n \beta^i = 0$ gilt:⁵

$$\beta\beta^j = \beta^{j+1} = \begin{cases} -\sum_{i=1}^n \beta^i & \text{falls } j = n \\ \beta^{j+1} & \text{sonst} \end{cases}$$

In der Matrixform stehen in Zeile β^j die Koeffizienten der Darstellung von $\beta\beta^j$:

$$\begin{array}{c} \beta \\ \beta^2 \\ \beta^3 \\ \vdots \\ \beta^{n-1} \\ \beta^n \end{array} \begin{pmatrix} \beta & \beta^2 & \beta^3 & \dots & \beta^{n-1} & \beta^n \\ 0 & +1 & 0 & \dots & 0 & 0 \\ 0 & 0 & +1 & & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & +1 \\ -1 & -1 & -1 & \dots & -1 & -1 \end{pmatrix} \quad (1.9)$$

Permutation der Spalten und Zeilen ändert nicht die Anzahl der Einträge ungleich Null, so dass infolge $c_{\mathcal{N}} = n + (n-1) \cdot 1 = 2n-1$ die Normalbasis \mathcal{N} optimal ist. \square

Als Typ I werden optimale Normalbasen bezeichnet, deren Multiplikationstabellen folgende Struktur aufweisen (die übrigen Einträge sind Null):

- Eine Zeile besteht aus (-1) -Einträgen und
- die übrigen Zeilen enthalten genau eine 1.

Einen anderen Aufbau haben die Multiplikationstabellen optimaler Normalbasen des Typs II:

Satz 1.3.6 (ONB Typ II). *Sei $2n+1$ prim und*

- a) $\langle 2 \rangle = \mathbb{Z}_{2n+1}^*$ oder
- b) $2n+1 \equiv 3 \pmod{4}$ und $\langle 2 \rangle = \text{QR}_{2n+1}$.

Dann erzeugt $\beta := \omega + \omega^{-1}$ eine optimale Normalbasis von \mathbb{F}_{2n} über \mathbb{F}_2 , für eine $(2n+1)$ -te, primitive Einheitswurzel ω (über \mathbb{F}_2).

Beweis. Setze $\mathcal{N} := \{\beta, \beta^2, \dots, \beta^{2n-1}\}$. Unter beiden Voraussetzungen ist $\mathcal{N} = \{\beta, \beta^2, \dots, \beta^n\}$ und eine Basis von \mathbb{F}_{2n} über \mathbb{F}_2 .

⁵Die $(n+1)$ -ten Einheitswurzeln über \mathbb{F}_p sind die Nullstellen des Polynoms $f(X) = X^{n+1} - 1 \in \mathbb{F}_p[X]$. Infolge $X^{n+1} - 1 = (X-1)\sum_{i=0}^n X^i$ ist $f(\beta) = (\beta-1)\sum_{i=0}^n \beta^i = 0$. Da Körper nullteilerfrei sind, gilt $\sum_{i=0}^n \beta^i = 0$ für $\beta \neq 1$.

Als Typ II werden optimale Normalbasen bezeichnet, deren Multiplikationstabellen folgende Struktur aufweisen (die übrigen Einträge sind Null):

- Eine Zeile enthält genau eine Eins und
- die restlichen Zeilen je zwei Einsen.

Zum Vergleich: Bei Typ I besteht eine Zeile aus (-1) -Einträgen und die übrigen Zeilen enthalten genau eine $+1$.

1.3.3 Optimale Normalbasen von \mathbb{F}_{2^n}

Betrachten wir zum Abschluß den wichtigen Fall $p = 2$. Die Voraussetzungen für die Konstruktion von optimalen Normalbasen nach Satz 1.3.5 und Satz 1.3.6 lassen sich anhand der folgenden, hinreichenden Kriterien überprüfen:

- Für prime $r = 4s + 1$ mit $s > 2$ prim gilt $\langle 2 \rangle = \mathbb{Z}_r^*$.
- Für prime $r = 2s + 1$ mit $s \equiv 1 \pmod{4}$ prim gilt $\langle 2 \rangle = \mathbb{Z}_r^*$.
- Für prime $r = 2s + 1$ mit $s \equiv 3 \pmod{4}$ prim gilt $\langle 2 \rangle = \text{QR}_r$.

Eine Tabelle der n , für zu \mathbb{F}_{2^n} optimale Normalbasen des Typs I, des Typs II oder beider Typen existieren, findet sich in [BG⁺93, Chapter 5].

Wir konstruieren als Beispiel die Multiplikationstabelle T einer optimalen Normalbasis von \mathbb{F}_{2^4} . Die Konstruktion der optimalen Normalbasis vom Typ I leiten wir aus dem Beweis zu Satz 1.3.5 her. Sei $\beta \in \mathbb{F}_{2^4}$ eine primitive 5-te Einheitswurzel. Die Multiplikationstabelle, geordnet nach den Potenzen von β , sieht wie folgt aus (Vergleiche (1.9)):

$$\begin{array}{c} \beta \quad \beta^2 \quad \beta^3 \quad \beta^4 \\ \beta \left(\begin{array}{cccc} 0 & +1 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \\ -1 & -1 & -1 & -1 \end{array} \right), \\ \beta^2 \\ \beta^3 \\ \beta^4 \end{array}$$

Für die von β erzeugte, optimale Normalbasis $\mathcal{N} = \{\beta_0, \beta_1, \beta_2, \beta_3\}$ gilt

$$\begin{aligned} \beta_0 &= \beta = \beta \\ \beta_1 &= \beta^2 = \beta^2 \\ \beta_2 &= \beta^4 = \beta^4 \\ \beta_3 &= \beta^8 = \beta^3, \end{aligned}$$

die Multiplikationstabelle T ist gegeben durch:

$$\begin{matrix} & \beta_0 & \beta_1 & \beta_2 & \beta_3 \\ \beta_0 & \left(\begin{array}{cccc} 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & +1 \\ -1 & -1 & -1 & -1 \\ 0 & 0 & +1 & 0 \end{array} \right) \\ \beta_1 & & & & \\ \beta_2 & & & & \\ \beta_3 & & & & \end{matrix}.$$

Die Wahl einer optimalen Normalbase ermöglicht (neben schneller Addition) auch eine schnelle Multiplikation in \mathbb{F}_{2^n} . Quadrieren ist dank der Normalbasis besonders einfach, die Einträge des Koeffizientenvektors werden um eine Position zyklisch verschoben. Zur Berechnung des Inversen von $A \in \mathbb{F}_{2^n}^*$ existiert ein Algorithmus von ITOH und TSUJII [IT88], der auf Multiplikation und hauptsächlich Quadrierungen beruht. Es gilt:

$$A^{-1} = A^{(2^n-1)-1} = A^{2^n-2} = (A^{2^{n-1}-1})^2$$

Wir berechnen $A^{2^{n-1}-1}$ anhand folgender Identitäten:

$$A^{2^{n-1}-1} = \begin{cases} A & n=1 \\ A \cdot A^2 & n=2 \\ A^{(2^{\frac{n-1}{2}-1})(2^{\frac{n-1}{2}+1})} = (A^{2^{\frac{n-1}{2}-1})}^{2^{\frac{n-1}{2}}} \cdot A^{2^{\frac{n-1}{2}-1}} & n > 1, \text{ ungerade} \\ AA^{2^{n-1}-2} = A(A^{2^{n-2}-1})^2 & n > 2 \text{ gerade} \end{cases}$$

Ein einfacher Induktionsbeweis nach n zeigt:

Satz 1.3.7 (Itoh, Tsujii 1988). *Das Inverse zu $A \in \mathbb{F}_{2^n}^*$ ist mit $n-1$ Quadrierungen und*

$$[\log_2(n-1)] + \#_1(n-1) - 1$$

Multiplikationen zu berechnen, wobei $\#_1(n-1)$ die Anzahl der Einsen in der Dualdarstellung von $n-1$ bezeichnet.

Literaturverzeichnis

- [BG⁺93] I. BLAKE, X.H. GAO, R.C. MULLIN, S.A. VANSTONE und T. YAGHOUBAIN: **Application of Finite Fields**, A.J. Menezes (ed.), Kluwer Academic Publishers, 1993,
- [BSS99] I. BLAKE, G. SEROUSSI und N. SMART: **Elliptic Curves in Cryptography**, London Mathematical Society Lecture Notes, Band 265, Cambridge University Press, 1999.
- [DK90] S.R. DUSSÉ und B.S. KALISKI: *A Cryptographic Library for the Motorola DSP56000*, Proceedings Eurocrypt '90, Springer Lecture Notes in Computer Science, Band 473, Seiten 230–244, 1991.
- [GL92] S. GAO und H.W. LENSTRA: *Optimal Normal Bases*, Design, Codes and Cryptography, Band 2, Seiten 315–323, 1992.
- [gmp] GNU-Software: *gmp*, C-Library für Langzahl-Arithmetik. Erhältlich im WWW u.a. auf den Sites <http://www.swox.com/gmp/> oder www.gnu.org.
- [IT88] T. ITOH und S. TSUJII: *A fast Algorithm for Computing Multiplicative Inverse in $GF(2^m)$ Using Normal Bases*, Information and Computation, Band 78, Seiten 171–177, 1988.
- [K95] B.S. KALISKI: *The Montgomery Inverse and Its Application*, IEEE Transaction on Computers, Band 44, Nr. 8, Seiten 1064–1065, 1995.
- [K98] D.E. KNUTH: **Seminumerical Algorithms**, The Art of Computer Programming, Band II, 3.te Auflage, Addison Wesley, 1998.
- [LN86] R. LIDL und H. NIEDERREITER: **Introduction to Finite Fields and Their Applications**, Cambridge University Press, 1986.
- [MOV97] A.J. MENEZES, P.C. VAN OORSCHOT und S.A. VANSTONE: **Handbook of Applied Cryptography**, CRC Press, 1997.

- [M85] P. MONTGOMERY: *Modular Multiplication Without Trial Division*, Mathematics of Computation, Band 44, Nr. 170, Seiten 519–521, 1985.
- [sun] Sun Microsystems: *Java*, Erhältlich im WWW u.a. auf der Site <http://java.sun.com>.