Addition von Punkten

Lemma Addition von Punkten auf E

Seien P, Q auf E mit $P \neq -Q$. Dann schneidet die Gerade durch P, Q die Kurve E in einem dritten Punkt R mit $R := -(P + Q)$.

Beweis:

- Wir zeigen nur $P \neq Q$. Der Beweis für $P = Q$ folgt analog.
- Wie zuvor setzen wir $P = (x_1, y_1)$, $Q = (x_2, y_2)$ und $R = (x_3, y_3)$.
- Sei G die Gerade $y = \alpha x + \beta$ durch P, Q. Dann gilt für $i = 1, 2$

$$(\alpha x_i + \beta)^2 = x_i^3 + a x_i + b.$$

- x_1, x_2 sind damit Nullstellen des Polynoms $g(x) = x^3 - \alpha^2 x^2 + \ldots$.
- Dann muss $g(x)$ 3 Nullstellen besitzen

$$g(x) = (x - x_1)(x - x_2)(x - x_3) = x^3 - (x_1 + x_2 + x_3)x^2 + \ldots$$

- Durch Koeffizientenvergleich folgt $x_1 + x_2 + x_3 = \alpha^2$.
- Wir erhalten $y_3 = \alpha x_3 + \beta$ und damit $-R = (x_3, -y_3)$.

Eigenschaften der Addition auf E

Korollar Effizienz der Addition

Sei E eine elliptische Kurve mit Punkten P, Q. Dann kann $P + Q$ in Laufzeit $\mathcal{O}(\log^2 p)$ berechnet werden.

- Wir benötigen nur Addition, Multiplikation und Division in $\mathbb{Z}_p$.

Satz von Mordell

Jede elliptische Kurve E bildet mit der definierten Addition eine abelsche Gruppe.

Beweis:

- Abgeschlossenheit: $P + Q$ liefert wieder einen Punkt auf E.
- Neutrales Element ist der Punkt $\mathbf{O}$.
- Inverses von $P \neq \mathbf{O}$ ist $-P$ und $-\mathbf{O} = \mathbf{O}$.
- Abelsch: Berechnung von G unabhängig von Reihenfolge P, Q.
- Assoziativität kann durch Nachrechnen gezeigt werden.

Gruppenordnung einer elliptischen Kurve

Satz von Hasse

Sei E eine elliptische Kurve über $\mathbb{F}_p$. Dann gilt

$$|E| = p + 1 + t \text{ mit } |t| \leq 2\sqrt{p}.$$

Anmerkungen: (ohne Beweis)

- Sei $x \in \mathbb{Z}_p$ und $f(x) = x^3 + ax + b$.
- Falls $f(x)$ ein quadratischer Rest modulo p ist, dann existieren genau zwei Lösungen $\pm y$ der Gleichung $y^2 = f(x) \bmod p$, d.h.

  $$(x, y) \text{ und } (x, -y) \text{ liegen auf } E.$$
- Falls $f(x)$ ein Nichtrest ist, besitzt E keinen Punkt der Form $(x, \cdot)$.
- Genau die Hälfte aller Elemente in $\mathbb{Z}_p^*$ ist ein quadratischer Rest.
- Falls $x \mapsto f(x)$ sich zufällig verhält auf $\mathbb{Z}_p$, erwarten wir $\frac{p}{2} \cdot 2 = p$ Punkte. Hinzu kommt der Punkt **O**, d.h. $|E| \approx p + 1$.
- Satz von Hasse: $x \mapsto f(x)$ ist fast zufällig mit Fehler $|t| \leq 2\sqrt{p}$.

Unser Modell

- Shannon 1948: Informationstheorie und Mathematik der Kommunikation
- Hamming 1950: Erste Arbeit über fehlerkorrigierende Codes

Modell:
Sender $\rightarrow$ Kodierer $\rightarrow$ Kanal $\rightarrow$ Dekodierer $\rightarrow$ Empfänger

- Kanal ist bandbreitenbeschränkt (Kompression)
- Kanal ist fehleranfällig (Fehlerkorrektur)
  - Bits können ausfallen: $0 \rightarrow \epsilon$, $1 \rightarrow \epsilon$ (Ausfallkanal)
  - Bits können kippen: $0 \rightarrow 1$, $1 \rightarrow 0$ (Symmetrischer Kanal)

Motivierendes Bsp: Datenkompression

Szenario:

- Kanal ist **fehlerfrei**.
- Übertragen gescannte Nachricht:
  Wahrscheinlichkeiten: 99% weißer, 1% schwarzer Punkt.
- Weiße Punkte erhalten Wert 0, schwarze Wert 1.

Kodierer:

- Splitten Nachricht in Blocks der Größe 10.
- Wenn Block x=0000000000, kodiere mit 0, sonst mit 1x.
- 1 dient als Trennzeichen beim Dekodieren.

Dekodierer:

- Lese den Code von links nach rechts.
- Falls 0, dekodiere 0000000000.
- Falls 1, übernehme die folgenden 10 Symbole.

Erwartete Codelänge

Sei $q := \text{Ws}[\text{Block ist } 0000000000] = (0.99)^{10} \geq 0.9$.
Sei Y Zufallsvariable für die Codewortlänge eines 10-Bit Blocks:

$$E[Y] = \sum_{y \in \{0,1x\}} |y| \cdot \text{Ws}(Y = |y|) = 1 \cdot q + 11 \cdot (1 - q) = 11 - 10q.$$

- D.h. die erwartete Bitlänge der Kodierung eines 10-Bit Blocks ist
$$11 - 10q \leq 2.$$
- Datenkompression der Nachricht auf 20%.
- Können wir noch stärker komprimieren?
- Entropie wird uns Schranke für Komprimierbarkeit liefern.

Ausblick: fehlerkorrigierende Codes

Szenario: Binärer symmetrischer Kanal

- Bits 0,1 kippen mit Ws $p, p < \frac{1}{2}$ zu 1,0. (Warum $< \frac{1}{2}$?)
- Korrekte Übertragung $0 \mapsto 0$, $1 \mapsto 1$ mit Ws $1 - p$.
- In unserem Beispiel $p = 0.1$.

Kodierer:

- Verdreifache jedes Symbol, d.h. $0 \mapsto 000$, $1 \mapsto 111$
- Repetitionscode der Länge 3.

Dekodierer:

- Lese den Code in 3er-Blöcken.
- Falls mindestens zwei Symbole 0 sind, dekodiere zu 0.
- Sonst dekodiere zu 1.

Ws Dekodierfehler

Symbol wird falsch dekodiert, falls mind. zwei der drei Bits kippen.

$$\begin{aligned} & \text{Ws}(\text{Bit wird falsch dekodiert}) \\ = \; & \text{Ws}(\text{genau 2 Bits kippen}) + \text{Ws}(\text{genau 3 Bits kippen}) \\ = \; & 3 * p^2 * (1-p) + p^3 = 3 * 10^{-2} * (1 - 10^{-1}) + 10^{-3} \end{aligned}$$

- Ohne Kodierung Fehlerws von 0.1.
- Mit Repetitionscode Fehlerws von ≈ 0.03.
- Nachteil: Kodierung ist dreimal so lang wie Nachricht.
- **Ziel:**
  Finde guten Tradeoff zwischen Fehlerws und Codewortlänge.

Ausblick: fehlertolerante Codes

Szenario: Binärer Ausfallkanal

- Bits 0,1 gehen mit Ws $p, p < \frac{1}{2}$ verloren, d.h. $0 \mapsto \epsilon$ bzw. $1 \mapsto \epsilon$.
- Korrekte Übertragung $0 \mapsto 0$, $1 \mapsto 1$ mit Ws $1 - p$.
- In unserem Beispiel $p = 0.1$.

Kodierer: Repetitionscode der Länge 3.

Dekodierer:

- Lese den Code in 3er-Blöcken.
- Falls 3er-Block Zeichen $x \in \{0, 1\}$ enthält, Ausgabe x.

Fehler beim Dekodieren: Alle drei Symbole gehen verloren.

- $\text{Ws}(\text{Bit kann nicht dekodiert werden}) = p^3 = 0.001$.
- Fehlerws kleiner beim Ausfallkanal als beim sym. Kanal.

Definition Code

Bezeichnungen:

- Alphabet $A = \{a_1, \ldots, a_n\}$, Menge von Symbolen a_i
- Nachricht sind Elemente $m \in A^*$.

Definition Code

Sei A ein Alphabet. Eine (binäre) *Codierung C des Alphabets A* ist eine injektive Abbildung

$$\begin{aligned} C: \quad & A \to \{0,1\}^* \\ & a_i \mapsto C(a_i). \end{aligned}$$

Die *Codierung einer Nachricht* $m = a_{i_1} \ldots a_{i_\ell} \in A^*$ definieren wir als

$$C(m) = C(a_{i_1}) \ldots C(a_{i_\ell}) \quad \text{(Erweiterung von } C \text{ auf } A^*).$$

Die Abbildung C heißt *Code*.

Bezeichnungen Code

- Die Elemente $c_i := C(a_i)$ bezeichnen wir als *Codeworte*.
- Wir bezeichnen sowohl die Abbildung von Nachrichten auf Codeworte als auch die *Menge der Codeworte* mit dem Buchstaben C.
- Falls $C \subseteq \{0,1\}^n$ spricht man von einem *Blockcode* der Länge n. In einem Blockcode haben alle Codeworte die gleiche Länge.

Entschlüsselbarkeit von Codes

Szenario: Datenkompression in fehlerfreiem Kanal

Definition eindeutig entschlüsselbar

Ein Code heißt eindeutig entschlüsselbar, falls jedes Element aus $\{0,1\}^*$ Bild höchstens einer Nachricht ist. D.h. die Erweiterung der Abbildung C auf A^* muss injektiv sein.

Definition Präfixcode

Ein Code $C = \{c_1, \ldots, c_n\}$ heißt Präfixcode, falls es keine zwei Codeworte $c_i \neq c_j$ gibt mit

$$c_i \text{ ist Präfix (Wortanfang) von } c_j.$$

Beispiel

	a_1	a_2	a_3
C_1	0	0	1
C_2	0	1	00
C_3	0	01	011
C_4	0	10	11

- C_1 ist kein Code, da $C_1 : A \to \{0,1\}^*$ nicht injektiv.
- C_2 ist nicht eindeutig entschlüsselbar, da $C_2 : A^* \to \{0,1\}^*$ nicht injektiv.
- C_3 ist eindeutig entschlüsselbar, aber kein Präfixcode.
- C_4 ist ein Präfixcode.

Präfixcodes sind eindeutig entschlüsselbar.

Satz Präfixcode eindeutig entschlüsselbar

Sei $C = \{c_1, \ldots, c_n\}$ ein Präfixcode. Dann kann jede kodierte Nachricht $C(m)$ in Zeit $\mathcal{O}(|C(m)|)$ eindeutig zu m decodiert werden.

Beweis:

- Zeichne binären Baum
  - Kanten erhalten Label 0 für linkes Kind, 1 für rechtes Kind.
  - Codewort $c_i = c_{i_1} \ldots c_{i_k}$ ist Label des Endknoten eines Pfads von der Wurzel mit den Kantenlabeln $c_{i_1}, \ldots, c_{i_n}$
- **Präfixeigenschaft:** Kein einfacher Pfad von der Wurzel enthält zwei Knoten, die mit Codeworten gelabelt sind.
- Codewort c_i ist Blatt in Tiefe $|c_i|$

Algorithmus Dekodierung Präfix

Algorithmus Dekodierung Präfix

1. Lese $C(m)$ von links nach rechts.
2. Starte bei der Wurzel. Falls 0, gehe nach links. Falls 1, gehe nach rechts.
3. Falls Blatt mit Codewort $c_i = C(a_i)$ erreicht, gib a_i aus und iteriere.

Laufzeit: $\mathcal{O}(|C(m)|)$

Woher kommen die Nachrichtensymbole?

Modell

- *Quelle Q* liefert Strom von Symbolen aus A.
- Quellwahrscheinlichkeit: $\mathrm{Ws}[\text{ Quelle liefert } a_i] = p_i$
- Ws p_i ist unabhängig von der Zeit und vom bisher produzierten Strom (erinnerungslose Quelle)
- X_i: Zufallsvariable für das Quellsymbol an der i-ten Position im Strom, d.h.

$$\mathrm{Ws}[X_i = a_j] = p_j \quad \text{für } j = 1, \ldots, n \text{ und alle } i.$$

Ziel: Kodiere a_j mit großer Ws p_j mittels kleiner Codewortlänge.

Kompakte Codes

Definition Erwartete Codewortlänge

Sei Q eine Quelle mit Alphabet $A = \{a_1, \ldots, a_n\}$ und Quellwahrscheinlichkeiten $p_1, \ldots, p_n$. Die Größe

$$E(C) := \sum_{i=1}^{n} p_i |C(a_i)|$$

bezeichne die erwartete Codewortlänge.

Definition Kompakter Code

Ein Code C heißt kompakt bezüglich einer Quelle Q, falls er *minimale erwartete Codewortlänge* besitzt.

Wann sind Codes eindeutig entschlüsselbar?

Definition Suffix

Sei C ein Code. Ein String $s \in \{0,1\}^*$ heißt Suffix in C falls

1. $\exists c_i, c_j \in C : c_i = c_j s$ oder
2. $\exists c \in C$ und einen Suffix s' in C: $s' = cs$ oder
3. $\exists c \in C$ und einen Suffix s' in C: $c = s's$.

- Bedingung 1: Codewort c_j lässt sich zu Codewort c_i erweitern.
- Bedingung 2: Codewort c lässt sich zu Suffix s' erweitern.
- Bedingung 3: Suffix s' lässt sich zu Codewort c erweitern.

Effiziente Berechnung von Suffixen

Algorithmus Berechnung Suffix

EINGABE: $C = \{c_1, \ldots, c_n\}$

1. Setze $S := \emptyset, T := \emptyset$.
2. Für alle $c_i, c_j \in C \times C$: Falls es ein $s \in \{0,1\}^*$ gibt mit $c_i = c_j s$, füge s in S und T ein.
3. Solange $T \neq \emptyset$
   1. Entferne ein beliebiges s' aus T.
   2. Für alle $c \in C$: Falls es ein $s \in \{0,1\}^* \setminus S$ gibt mit $s' = cs$ oder $c = s's$, füge s zu S und T hinzu.

AUSGABE: Menge S der Suffixe von C

Laufzeit Suffixberechnung

Laufzeit:

- Schritt 2: $\mathcal{O}(n^2)$ Codewortpaare
- Suffixlänge ist durch $\max_i\{|c_i|\}$ beschränkt.
- Es kann höchstens $n \cdot \max_i\{|c_i|\}$ Suffixe geben. (Warum?)
- Schritt 3: $\mathcal{O}(n^2 \cdot \max_i\{|c_i|\})$
- **Polynomiell in der Eingabelänge**: n, $max_i\{|c_i|\}$