McEliece Verfahren (1978)

- **Dekodieren eines zufälligen linearen Codes ist NP-hart.**
- Verwende linearen Code C mit effizientem Dekodierverfahren (z.B. sogenannten Goppa-Code).
- Generatormatrix von C bildet den geheimen Schlüssel.
- C wird in äquivalenten linearen Code C' transformiert.

Algorithmus Schlüsselgenerierung McEliece

1. Wähle linearen $[n, k, d]$-Code C mit Generatormatrix G.
2. Wähle zufällige binäre $(k \times k)$-Matrix S mit $\det(S) = 1$.
3. Wähle zufällige binäre $(n \times n)$-Permutationsmatrix P.
4. $G' \leftarrow SGP$

öffentlicher Schlüssel: G', geheimer Schlüssel S, G, P.

McEliece Verschlüsselung

Algorithmus McEliece Verschlüsselung

EINGABE: Plaintext $\mathbf{m} \in \mathbb{F}_2^k$

1. Wähle zufälligen Fehlervektor $\mathbf{e} \in \mathbb{F}_2^n$ mit $w(\mathbf{e}) = \lfloor \frac{d-1}{2} \rfloor$.
2. $\mathbf{c} \leftarrow \mathbf{m}G' + \mathbf{e}$.

AUSGABE: Ciphertext $\mathbf{c} \in \mathbb{F}_2^n$

Vorgeschlagene Parameter:

- $[1024, 512, 101]$-Goppacode C.
- Plaintextlänge: 512 Bit, Chiffretextlänge: 1024 Bit.
- Größe des öffentlichen Schlüssels: 512×1024 Bit.

McEliece Entschlüsselung

Algorithmus McEliece Entschlüsselung

EINGABE: Ciphertext $\mathbf{c} \in \mathbb{F}_2^n$

1. $\mathbf{x} \leftarrow \mathbf{c}P^{-1}$.
2. Dekodiere $\mathbf{x}$ mittels Dekodieralgorithmus für C zu $\mathbf{m}'$.
3. $\mathbf{m} \leftarrow \mathbf{m}'S^{-1}$

AUSGABE: Plaintext $\mathbf{m} \in \mathbb{F}_2^k$

- **Korrektheit**:

  $$\mathbf{x} = \mathbf{c}P^{-1} = (\mathbf{m}G' + \mathbf{e}) \cdot P^{-1} = (\mathbf{m}SGP + \mathbf{e}) \cdot P^{-1} = (\mathbf{m}S)G + \mathbf{e} \cdot P^{-1}.$$

- $\mathbf{e} \cdot P^{-1}$ besitzt Gewicht $w(\mathbf{e}P^{-1}) = \lfloor \frac{d-1}{2} \rfloor$.
- Dekodierung liefert $\mathbf{m}' = \mathbf{m}S$, d.h. $\mathbf{m} = \mathbf{m}'S^{-1}$.

Stern Identifikationsschema

- Dekodieren eines zufälligen linearen Codes ist NP-hart.
- Definiere zufälligen Code mittels zufälliger Parity Check Matrix.

Gegeben: $[n,k]$-Code C mittels $P \in \mathbb{F}_2^{(n-k)\times n}$ und $\mathbf{x} \in \mathbb{F}_2^n$
Gesucht: $e \in \mathbb{F}_2^n$ mit $\mathbf{x} - \mathbf{e} = \mathbf{c} \in C$, so dass $w(\mathbf{e})$ minimal ist, d.h gesucht ist $\mathbf{e}$ minimalen Gewichts mit $S(\mathbf{x}) = S(\mathbf{e}) = \mathbf{e}P^t$.

Stern Schlüsselerzeugung

Globale Parameter:

1. Parity Check Matrix $P \in \mathbb{F}_2^{(n-k)\times n}$ mit linear unabhängigen Zeilen.
2. Gewicht $g \in \mathbb{N}$

Jeder Teilnehmer wählt

1. Geheimer Schlüssel: $\mathbf{e} \in F_2^n$ mit $w(\mathbf{e}) = g$
2. Öffentlicher Schlüssel: $\mathbf{e}P^t$

- Vorgeschlagen: $[n,k] = [512, 256]$ und $g = w(\mathbf{e}) = 56$.
- **Idee Identifikation:** Beweise Besitz von $\mathbf{e}$, ohne $\mathbf{e}$ preiszugeben.

Identifikationsverfahren

Algorithmus Stern Identifikation

Prover: Wähle zufällige $\mathbf{y} \in \mathbb{F}_2^n$ und Permutation $\sigma : \mathbb{F}_2^n \to \mathbb{F}_2^n$.
Hinterlege beim Verifier

$$c_0 = \sigma(\mathbf{y} + \mathbf{e}), c_1 = \sigma(\mathbf{y}) \text{ und } c_2 = (\sigma, \mathbf{y}P^t).$$

Verifier: Wähle zufälliges $b \in \{0, 1, 2\}$ für Prüfung von $c_i, i \neq b$.

Prover: Falls $b = 0$: Sende $\mathbf{y}$,σ und öffne c_1, c_2.
Falls $b = 1$: Sende $\mathbf{y} + \mathbf{e}, \sigma$ und öffne c_0, c_2.
Falls $b = 2$: Sende $\sigma(\mathbf{y}), \sigma(\mathbf{e})$ und öffne c_0, c_1.

Verifier: Falls $b = 0$: Prüfe Korrektheit von c_1 und c_2.
Falls $b = 1$: Prüfe c_0 und $c_2 = (\sigma, (\mathbf{y} + \mathbf{e})P^t - \mathbf{e}P^t)$.
Falls $b = 2$: Prüfe $c_0 = \sigma(\mathbf{y}) + \sigma(\mathbf{e}), c_1, w(\sigma(\mathbf{e})) = w(\mathbf{e})$.

Korrektheit: Nur Besitzer von **e** bestehen Protokoll.

- **Completeness:** Falls Prover **e** besitzt, akzeptiert Verifier.
- **Soundness:** Falls Prover **e** nicht besitzt, besteht er das Protokoll mit Ws höchstens $\frac{2}{3}$.
- **Strategie 1**: Prover wählt σ, **y** und $\tilde{\mathbf{e}}$ mit Gewicht $w(\tilde{\mathbf{e}})$.
  - Prover besteht nur $b = 1$ nicht, da hier $\tilde{\mathbf{e}}P^t \neq \mathbf{e}P^t$.
- **Strategie 2**: Prover wählt σ, **y** und $\mathbf{y} + \tilde{\mathbf{e}}$ mit $\tilde{\mathbf{e}}P^t = \mathbf{0}$.
  - Prover besteht nur $b = 2$ nicht, da hier $w(\tilde{\mathbf{e}}) \neq w(\mathbf{e})$.
- Prover wählt **Strategie 1** und **Strategie 2** jeweils mit Ws $\frac{1}{2}$.

$$\begin{aligned}&\mathrm{Ws}(\text{P besteht Protokoll})\\ &= \mathrm{Ws}(b \neq 1) \cdot \mathrm{Ws}(\textbf{Strategie 1}) + \mathrm{Ws}(b \neq 2) \cdot \mathrm{Ws}(\textbf{Strategie 2})\\ &= \tfrac{2}{3}\left(\tfrac{1}{2} + \tfrac{1}{2}\right) = \tfrac{2}{3}.\end{aligned}$$

Fakt (Beweis ist nicht-trivial)

Jeder Angreifer mit $\mathrm{Ws} > \frac{2}{3}$ liefert Berechnung von **e**.

- Intuitiv: Prover kann nur $b = 1$ und 2 bestehen, falls er **e** kennt.

Zeroknowledge Eigenschaft

- **Zeroknowledge:** Verifier lernt nichts über **e**.
- Verifier lernt für
  - $b = 0$: Zufälliges $\mathbf{y} \in \mathbb{F}_2^n$, unabhängig von **e**.
  - $b = 1$: Zufälliges $\mathbf{y} + \mathbf{e} \in \mathbb{F}_2^n$, da $\mathbf{y} \in \mathbb{F}_2^n$ zufällig ist.
    (D.h. **y** ist One-Time Pad für **e**.)
  - $b = 2$: Zufälliges $\sigma(\mathbf{e}) \in \mathbb{F}_2^n$ mit Gewicht $w(\mathbf{e})$.
- Formaler Zeroknowledge Beweis verwendet Simulator für Prover, ohne dabei **e** zu kennen.

Einführung in die NP-Vollständigkeitstheorie

Notationen

- Alphabet $A = \{a_1, \ldots, a_m\}$ aus Buchstaben a_i
- Worte der Länge n sind Elemente aus $A^n = \{a_{i_1} \ldots a_{i_n} \mid a_{i_j} \in A\}$.
- $A^0 = \epsilon$, ϵ ist das leere Wort.
- $A^* = \bigcup_{n=0}^{\infty} A^n$, $A^+ = A^* \setminus \epsilon$, $A^{\leq m} = \bigcup_{n=0}^{m} A^n$
- Länge $|a_1 \ldots a_n| = n$. $\mathrm{bin}(a_1)$ ist Binärkodierung von a_1.

Definition Sprache L

Sei A ein Alphabet. Eine Menge $L \subseteq A^*$ heißt *Sprache* über dem Alphabet A. Das *Komplement* von L über A ist definiert als $\bar{L} = A^* \setminus L$.

Turingmaschine (informal)

Turingmaschine besteht aus:

- Einseitig unendlichem Band mit Zellen (Speicher),
- Kontrolle und einem Lesekopf, der auf einer Zelle steht.

Arbeitsweise einer Turingmaschine

- Bandsymbol $\triangleright$ steht in der Zelle am linken Bandende.
- Kontrolle besitzt Zustände einer endlichen Zustandsmenge.
- Abhängig vom Zelleninhalt und Zustand schreibt die Kontrolle ein Zeichen und bewegt den Lesekopf nach links oder rechts.
- Zu Beginn der Berechnung gilt:
  - Lesekopf befindet sich auf dem linken Bandende $\triangleright$.
  - Band enthält $\triangleright a_1 \ldots a_n \sqcup \sqcup \ldots$, wobei $a_1 \ldots a_n$ die Eingabe ist.
- Turingmaschine M hält nur, falls Kontrolle in Zuständen q_a oder q_r.
  - Falls M in q_a hält: M akzeptiert die Eingabe $a_1 \ldots a_n$.
  - Falls M in q_r hält: M verwirft die Eingabe $a_1 \ldots a_n$.
  - Falls M nie in die Zustände q_a, q_r kommt: M läuft unendlich.

Turingmaschine (formal)

Definition Deterministische Turingmaschine (Turing 1936)

Eine deterministische Turingmaschine DTM ist ein 4-Tupel $(Q, \Sigma, \Gamma, \delta)$ bestehend aus

1. Zustandmenge Q: Enthält Zustände q_a, q_r, s.
2. Bandalphabet Γ mit $\sqcup, \triangleright \in \Gamma$
3. Eingabealphabet $\Sigma \subset \Gamma \setminus \{\sqcup, \triangleright\}$.
4. Übergangsfunktion $\delta : Q \setminus \{q_a, q_r\} \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
   - Es gilt stets am linken Bandende $\delta(q, \triangleright) = (q', \triangleright, R)$.
   - Es gilt nie $\delta(q, a) = (q', \triangleright, L/R)$ (nicht am linken Bandende).

Beispiel DTM M_1

Bsp: $a^n, n \geq 1$

- $Q = \{q_0, q_1, q_a, q_r\}$ mit $s = q_0$
- $\Sigma = \{a\}$ und $\Gamma = \{\sqcup, \triangleright, a\}$
- Übergangsfunktion

δ	a	$\sqcup$	$\triangleright$
q_0	(q_1, a, R)	$(q_r, \sqcup, R)$	$(q_0, \triangleright, R)$
q_1	(q_1, a, R)	$(q_a, \sqcup, R)$	$(q_1, \triangleright, R)$

Notation der Konfigurationen bei Eingabe a^2:

$$
\begin{aligned}
& q_0 \triangleright aa \\
\vdash\ & \triangleright q_0 aa \\
\vdash\ & \triangleright a q_1 a \\
\vdash\ & \triangleright aa q_1 \sqcup \\
\vdash\ & \triangleright aa \sqcup q_a \sqcup
\end{aligned}
$$

Nachfolgekonfigurationen

Notation Nachfolgekonfiguration

- Direkte Nachfolgekonfiguration: $aqb \vdash a'q'b'$
- i-te Nachfolgekofiguration: $aqb \vdash^i a'q'b'$
- Indirekte Nachfolgekonfiguration $aqb \vdash^* a'b'q'$, d.h. $\exists i \in \mathbb{N} : aqb \vdash^i a'q'b'$.

Akzeptanz und Ablehnen von Eingaben

- DTM M erhalte Eingabe $w \in \Sigma^*$.
  - M akzeptiert $w \Leftrightarrow \exists a, b \in \Gamma^* \text{ mit } s \triangleright w \vdash^* aq_a b$
  - M lehnt w ab $\Leftrightarrow \exists a, b \in \Gamma^* \text{ mit } s \triangleright w \vdash^* aq_r b$

Akzeptierte Sprache, L rekursiv aufzählbar

Definition Akzeptierte Sprache, Rekursive Aufzählbarkeit

Sei M eine DTM. Dann bezeichne

$$L(M) = \{w \in \Sigma^* \mid M \text{ akzeptiert Eingabe } w\}$$

die von M *akzeptierte Sprache*.
Eine Sprache L heißt *rekursiv aufzählbar* gdw eine DTM M existiert mit $L = L(M)$.

- Unsere Beispiel DTM M_1 akzeptiert die Sprache $L(M_1) = \{a\}^+$.
- D.h. $L = \{a\}^+$ ist rekursiv aufzählbar, da für M_1 gilt $L = L(M_1)$.
- Aus der obigen Definition folgt:
  L ist nicht rekursiv aufzählbar $\Leftrightarrow \nexists$ DTM M mit $L = L(M)$.
- Es gibt Sprachen, die nicht rekursiv aufzählbar sind, z.B.
  $\bar{H} = \{\langle M, x\rangle \mid M \text{ hält bei Eingabe } w \text{ nicht.}\}$. (ohne Beweis)

Definition Entscheidbarkeit und rekursive Sprachen

Definition Entscheidbarkeit

Sei M eine DTM, die die Sprache $L(M)$ akzeptiert. Wir sagen, dass M die Sprache $L(M)$ *entscheidet* gdw M alle Eingaben $w \in \bar{L}$ ablehnt.
D.h. insbesondere M hält auf allen Eingaben.
Eine Sprache L heißt *entscheidbar* gdw eine DTM M existiert, die L entscheidet.

- Unsere Beispiel-DTM M_1 entscheidet die Sprache $L(M_1) = \{a\}^+$.
- $L = \{a\}^+$ ist entscheidbar, da M_1 die Sprache L entscheidet.

Korollar Entscheidbarkeit impliziert rekursive Aufzählbarkeit

Sei L eine entscheidbare Sprache. Dann ist L rekursiv aufzählbar.

- Die Rückrichtung stimmt nicht:
  Es gibt rekursiv aufzählbare L, die nicht entscheidbar sind, z.B.
  $H = \{\langle M, x\rangle \mid M \text{ hält auf Eingabe } w.\}$. (ohne Beweis)

Entscheiden versus Berechnen

Definition Berechnung von Funktionen

Eine DTM M berechnet die Funktion $f : \mathbb{N}^n \to \mathbb{N}$, falls M für jedes $(a_1, \ldots, a_n)$ bei Eingabe $\mathrm{bin}(a_1)\# \ldots \#\mathrm{bin}(a_n)$ den Bandinhalt $\mathrm{bin}(f(a_1, \ldots, a_n))$ berechnet und in q_a hält.

- Werden der Einfachheit halber Sprachen entscheiden, nicht Funktionen berechnen.