7. Woche
Codierung in der Kryptographie

Keine Details über

Hamming Code, Simplex Code, Golay Code, Reed-Reed-Muller Codes, Cyclic Codes. Das Buch von Roman ist eine gute Einführung.

Zwei Kryptographische Anwendungen von Codes.

McEliece Verfahren (1978)

- **Decodieren eines zufälligen linearen Codes ist NP-hart**.
- Verwende linearen Code C mit effizientem Decodierverfahren (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. Decodiere $\mathbf{x}$ mittels Decodieralgorithmus 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 $w(\mathbf{e}P^{-1}) = w(\mathbf{e}) = \lfloor \frac{d-1}{2} \rfloor$.
- Decodierung liefert $\mathbf{m}' = \mathbf{m}S$, d.h. $\mathbf{m} = \mathbf{m}'S^{-1}$.

Stern Identifikationsschema

- Decodieren 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: $\mathbf{e} \in \mathbb{F}_2^n$ 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 $S(\mathbf{x}) = S(\mathbf{e}) = \mathbf{e}P^t$.

Algorithmus 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: $\mathbf{e} \in \mathbb{F}_2^n$ mit $w(\mathbf{e}) = g$
2. Öffentlicher Schlüssel: $S(\mathbf{e}) = \mathbf{e}P^t$

- 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 (sogenanntes Commitment)

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

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 $\tilde{\mathbf{e}}P^t = \mathbf{e}P^t$.
  - 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 $\frac{1}{2}$.

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

Fakt (Beweis ist nicht-trivial)

Jeder Angreifer mit $\mathcal{W}s > \frac{2}{3}$ 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.