Einführung Code-basierte Kryptografie

Code-basiertes Kryptosystem – McEliece

Fahrplan

Grundlagen

McEliece – Code-basierte Kryptografie

Quellen

Zusammmenfassung

- McEliece asymmetrisches Public-Key-Kryptosystem – 1978 nach Robert McEliece [McE78]
- Grundlegende Idee: Führe absichtliche Fehler in die Chiffre ein
- Verwenden eines allgemeinen fehlerkorrigierende Codes
  - Dekodierung i.A. $\mathcal{NP}$-Hart [Sch07, S. 479], [SP18, S. 353ff]
  - Unterklasse an linearen Codes auch in P lösbar $\rightarrow$ Goppa-Codes
- Angreifer ohne Goppa-Code kann nur in $\mathcal{P}$, also polynomiell viel rechnen
  - Die Entschlüsselung eines zufälligen linearen Codes ist ein $\mathcal{NP}$-Hartes Problem -> [Lju04]
  - Die Generatormatrix eines Goppa-Codes sieht zufällig aus -> [Fau+13]

Fahrplan Grundlagen

Grundlagen
Galoiskörper
Hamming Gewicht und Distanz
Generatormatrix
Parity-Check-Matrix
Lineare Codes
Zyklischer Code
Zyklischer linearer Code
Generatorpolynom
Generatorpolynom
McElice Kodierungsproblem
McElice Kodierungsproblem
McEliece – Code-basierte Kryptografie
McEliece-Kryptosystem
Parameter Definitionen

Galoiskörper/Galoisfeld

- Ein Galoiskörper $GF(p)$, wobei p prim, ist ein endlicher Körper welcher bezüglich '$+$' und '$*$' abgeschlossen ist.
- Beispiel:
  $GF(2) = \mathbb{F}_2 = \{0, 1\}$: [Kun91]

Addition :	*Multiplikation* :
$0 + 0 = 0$	$0 * 0 = 0$
$0 + 1 = 1$	$0 * 1 = 0$
$1 + 0 = 1$	$1 * 0 = 0$
$1 + 1 = 0$	$1 * 1 = 1$

Hamming Gewicht

- Das Hamming Gewicht $w(\cdot)$ eines Vektors x mit Länge n ist definiert als:

$$w(x) := \sum_{i=1}^{n} w(x_i) \qquad \textit{mit} \qquad w(x_i) = \begin{cases} w(x_i) = 1 : x_i \neq 0 \\ \\ w(x_i) = 0 : x_i = 0 \end{cases}$$

- Beispiel:

$$w(\underline{1}00\underline{1}) = 2$$
$$w(0\underline{111}) = 3$$

Hamming Distanz

- Hamming Distanz $D(\cdot, \cdot)$ zwischen zwei validen Codewörten c und c' ist definiert als:

$$D(c, c') := |\{i \in \{1, \ldots, n\} | c_i \neq c'_i\}|$$

- Beispiel:

$$D(\underline{000}, \underline{111}) = 3$$
$$D(0011\underline{0}, 0011\underline{1}) = 1$$
$$D(\vec{c}, 0000) = w(\vec{c})$$

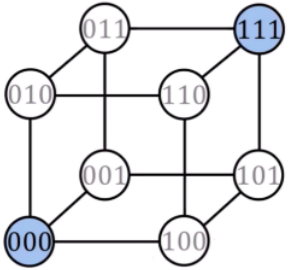
Minimale Hamming Distanz

- Die minimale Hamming Distanz d ist die kleinste Hamming Distanz zwischen zwei beliebigen gültigen Codewörten.

$$d = \min_{c \neq c'} D(c, c')$$

- Beispiel: $d = 3$

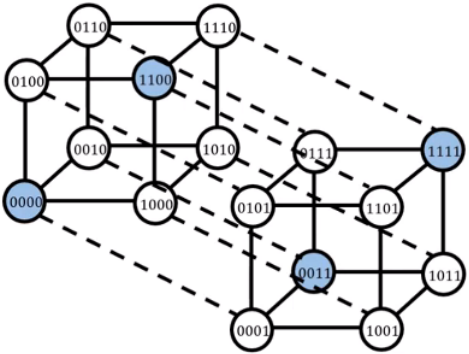
Codewort
000
111



Minimale Hamming Distanz

- Beispiel: $d = 2$

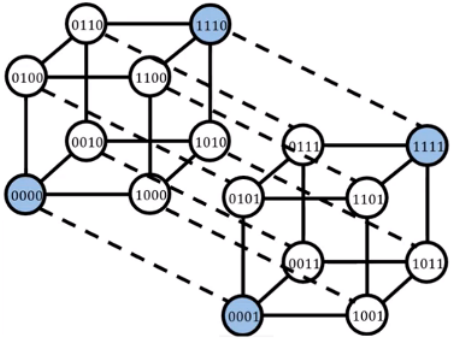
Codewort
0000
0011
1100
1111



Minimale Hamming Distanz

- Beispiel: $d = 1$

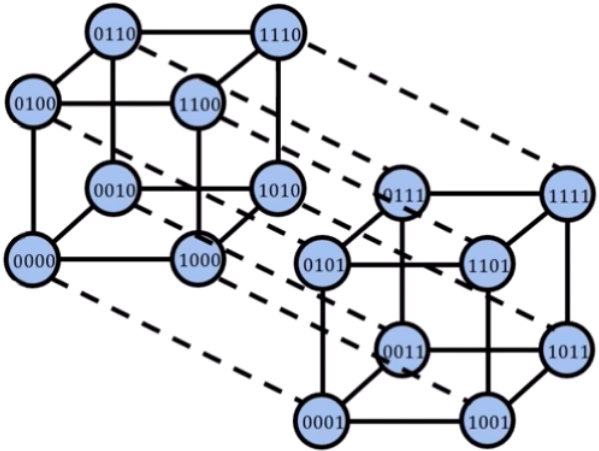
Codewort
0000
0001
1110
1111



Hamming Code Projektion

- Ein $(7, 4)$ Hamming-Code ohne mgl. Fehlerkorretur, da $d = 1$
- Projektion von $4d \to 7d$
  - D.h. 4 Bit Nachricht *abcd* auf 7 Bit Code-Wort *abcd xzy* abgebildet
- Bsp. *abcd* $\to$ *abcd xyz*

$$x = a \oplus b \oplus d$$
$$y = a \oplus c \oplus d$$
$$z = b \oplus c \oplus d$$



Hamming Code

- Beispiel:

$$abcd \to abcd\ xyz \qquad x = a \oplus b \oplus d$$
$$1001 \to 1001001 \qquad y = a \oplus c \oplus d$$
$$0010 \to 0010011 \qquad z = b \oplus c \oplus d$$

Hamming Code

- Check-Bits *xyz* können folgende Check-Bit-Status haben (Syndrome s)

x y z	**Error-State**	**a b c d**	**x y z**
✓✓✓	no error	✓✓✓✓	✓✓✓
× ✓✓	x is wrong	✓✓✓✓	×✓✓
✓× ✓	y is wrong	✓✓✓✓	✓×✓
✓✓×	z is wrong	✓✓✓✓	✓✓×
× × ✓	a is wrong	×✓✓✓	✓✓✓
× ✓×	b is wrong	×✓✓✓	✓✓✓
✓× ×	c is wrong	✓✓×✓	✓✓✓
× × ×	d is wrong	✓✓✓×	✓✓✓

Hamming Code

- Aus den Spalten können die Gleichungen wie folgt abgeleitet werden:

	x y z
a is wrong	× ×✓
b is wrong	×✓×
c is wrong	✓× ×
d is wrong	× × ×

$$x = a \oplus b \oplus d$$
$$y = a \oplus c \oplus d$$
$$z = b \oplus c \oplus d$$

Hamming Code

- Beispiel: (Fehlerkorrektur)

abcd xyz	$x = a \oplus b \oplus d$
1001 010	$y = a \oplus c \oplus d$
	$z = b \oplus c \oplus d$

Generatormatrix I

- Ein linearer binärer (n, k, d)-Code C kann durch die Generatormatrix G wie folgt charakterisiert werden:

$$C = \{c \in GF(2^n), m \in GF(2^k) : c = m \cdot G\}$$

  - n Länge des Codeworts
  - k Nachrichtenlänge
  - d minimale Distanz

- Beispiel:
  - $G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$ (Hamming$(7, 4)$-Code mit der Generatormatrix G)

Generatormatrix II

- Beispiel:
  Nachricht m wird mittels der Generatormatrix G in das zugehörige in das jeweilige Codewort c transformiert und es gilt $m \cdot G = c$:

$$\begin{pmatrix} a & b & c & d \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} a & b & c & d & x & y & z \end{pmatrix}$$

Parity-Check-Matrix / Prüfmatrix

- Ein linearer binärer (n, k, d)-Code C kann durch die Prüfmatrix H wie folgt charakterisiert werden:

$$C = \{c \in GF(2^n) : H\vec{c}^T = \vec{0}\}$$

- Bsp.: Hamming-Code $(7, 4)$ mit Parity-Check-Bit-Gleichungen *abcd xyz*

$$a \oplus b \oplus d = x$$
$$a \oplus c \oplus d = y$$
$$b \oplus c \oplus d = z$$

Parity-Check-Matrix / Prüfmatrix

- Durch umformen erhalten wir ($\oplus \equiv -$):

$$a \oplus b \oplus 0c \oplus d \oplus x \oplus 0y \oplus 0z = 0$$
$$a \oplus 0b \oplus c \oplus d \oplus 0x \oplus y \oplus 0z = 0$$
$$0a \oplus b \oplus c \oplus d \oplus 0x \oplus 0y \oplus z = 0$$

- LGS kann als Matrix geschrieben werden, alle validen Codeworte erfüllen dies

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \\ d \\ x \\ y \\ z \end{pmatrix}^T = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Parity-Check-Matrix Beispiel

- Ist 1101100 ein valides Codewort?
- Für valide Codewörter gilt: $H \cdot \vec{c}^T = \vec{0}$
- Wende H auf $\vec{c}^T$ an:

$$H \cdot \vec{c}^T = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}^T = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

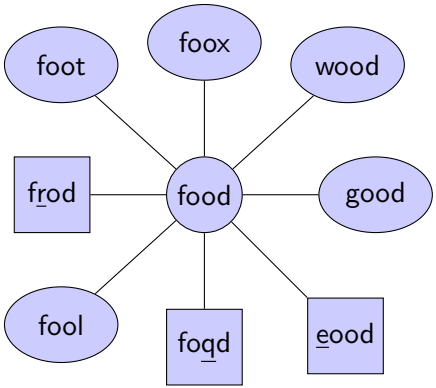
Parity-Check-Matrix Beispiel

- Durch addieren eines Fehlers $H(\vec{c}+\vec{e})^T = H \cdot \vec{c}^T + H \cdot \vec{e}^T = H \cdot \vec{e}^T$

$$H \cdot \vec{c}^T = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}^T \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}^T = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}^T + \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^T = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^T$$

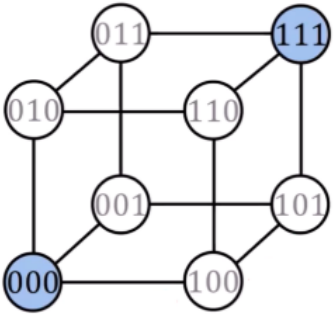
Recap / Idee der fehlerkorrigierenden Codes

- Gültige Codewörter helfen nicht bei der Fehlerkorrektur
- Ungültige helfen, da wir diese „auseinander“ ziehen können

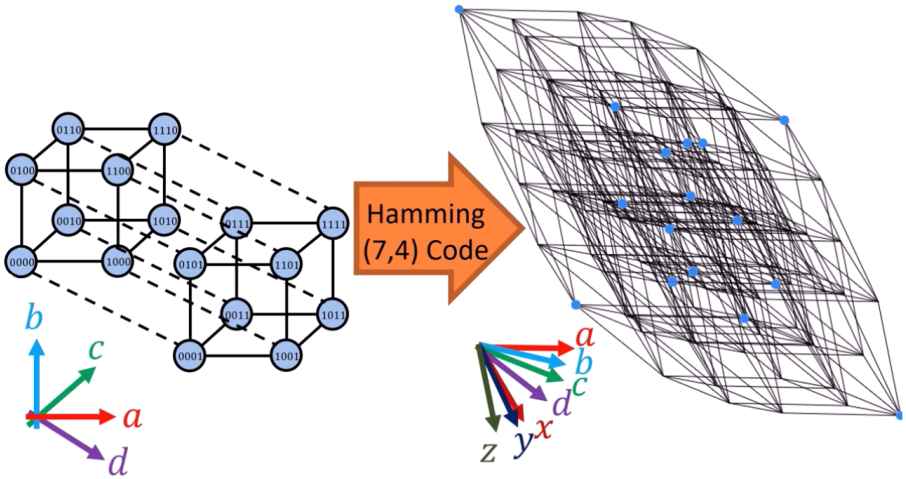


Recap / Idee der fehlerkorrigierenden Codes

- Generatormatrix $G = [1, 1, 1]$ transformiert 1-Bit Nachrichten in 3-Bit Codeworte
- Fehler können angeordnet werden, durch das Füllen der Lücken mit ungültigen Codeworten werden Fehler zuordenbar und korrigierbar
- Beispiel:
  $0 \cdot [111] \rightarrow [0, 0, 0]$ und $1 \cdot [1, 1, 1] \rightarrow [0, 0, 0]$



Beispiel Hamming(7,4)-Code



Lineare Codes

- **Defintion:** Ein binärer (n, k, d)-Code ist linear, die $\oplus$-Summe zweier beliebiger gültiger Codeworte c und c' wiederum ein gültiges Codewort ergibt:

$$\forall c, c' \in C \colon c \oplus c' \in C$$

  - Beispiel: $(0, 1, 1) + (0, 1, 1) = (0, 0, 0)$
- Bei gegebener Hamming Distanz d wird der Code C auch (n, k, d)-Code genannt

Zyklischer Code

- **Definition:** Ein binärer (n, k, d)-Code heißt zyklisch, wenn jede zyklische Verschiebung (Shift) eines gültiges Codewortes c wiederum ein gültiges Codewort ergibt:

$$\forall\ c = (c_1, c_2, \ldots, c_n) \in C \implies c' = (c_n, c_1, c_2, \ldots, c_{n-1}) \in C$$

  - Beispiel: $(10100) \in C \implies (01010) \in C$

Zyklischer linearer Code

- **Defintion:** Ein binärer (n, k, d)-Code ist linear und zyklisch wenn:
  - Die $\oplus$-Summe zweier beliebiger Codeworte c und c' wiederum ein gültiges Codewort ergibt:
    $$\forall c, c' \in C \colon c \oplus c' \in C$$
  - Jede zyklische Verschiebung eines Codeworte c wiederum ein gültiges Codewort ergibt:
    $$\forall\ c = (c_0, c_1, \ldots, c_{n_1}) \in C \implies c' = (c_{n_1}, c_0, c_1, \ldots, c_{n_2}) \in C$$

Zyklischer linearer Code

- Beispiel:
  Gegeben ein Generatorcodewort (100100):
  - Mehrfaches Verschieben bis hin zum ursprünglichen Codewort

$$(100100) \implies (010010) \implies (001001) \implies (100100)$$

Zyklischer linearer Code

- Beispiel:
  Gegeben ein Generatorcodewort c_{gen} (100100):
  - Addieren um die restlichen Codeworte zu erhalten:

$$(100100) + (100100) = (000000)$$
$$(100100) + (010010) = (110110)$$
$$(100100) + (001001) = (101101)$$
$$(010010) + (101101) = (111111)$$

Zyklischer linearer Code in polynomialer Darstellung

- Die Zeichen eines Codewortes c lassen sich als Koeffizienten eines Polynomes $c(x)$ betrachten:

$$\forall c = (c_0, c_1, \ldots, c_{n_1}) \iff c(x) = c_0 + c_1 x + \cdots + c_{n_1}$$

- Das Polynom des Generatorcodewortes ist das Generatorpolynom
  - Beispiel:

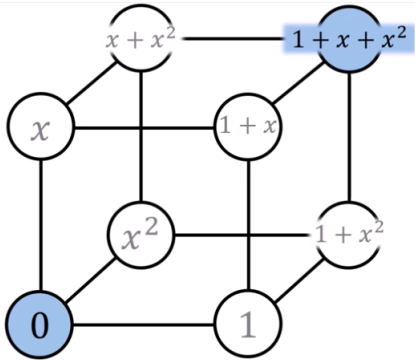
$$(1011) = \underline{1} \cdot X^0 + = \underline{0} \cdot X^1 + = \underline{1} \cdot X^1 + = \underline{1} \cdot X^3$$

Generatorpolynom

- **Defintion:** Ein von null verschiedenes Polynom g minimalen Grades eines zyklischen Codes heißt Generatorpolynom.
- Ein zyklischer Code der Länge n mit einem Generatorpolynom g des Grades r hat einen Coderaum der Dimension $r = n - k$ und codiert somit r -viele Bits pro Codewort.

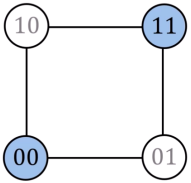
Generatorpolynom

Nachricht	Codewortpolynomial
0	0
1	$x^3 + 1$
x	$x^4 + x$
$x + 1$	$x^4 + x^3 + x + 1$
x^2	$x^5 + x^2$
$x^2 + 1$	$x^5 + x^3 + x^2 + 1$
$x^2 + x$	$x^5 + x^4 + x^2 + x$
$x^2 + x + 1$	$x^5 + x^4 + x^3 + x^2 + x + 1$

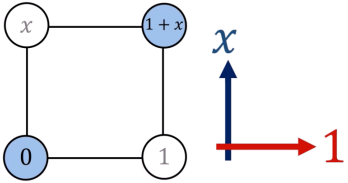


Recap Generatormatrix / Generatorpolynom

Nachricht	Codewort
0	00
1	11

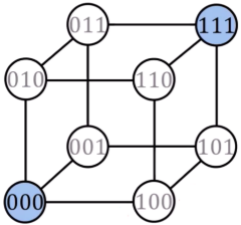


Nachricht	Codewort
0	0
1	$1+x$

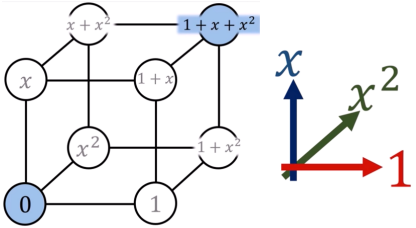


Recap Generatormatrix / Generatorpolynom

Nachricht	Codewort
0	000
1	111



Nachricht	Codewort
0	0
1	$1 + x + x^2$



Goppa-Code

- Ein binärer Goppa-Code ist ein linearer zyklischer (n, k, d)-Code, der durch ein Generatorpolynom g definiert ist und $\lfloor \frac{d-1}{2} \rfloor$ Fehler korrigieren kann.

McElice Kodierungsproblem

- McElice basiert auf klassischen Dekodierungsproblemen [**berlekamp1978inherent**]
- Sei C ein linearer (n, k) Code über $\mathbb{F}$ und $y \in C$ das empfangene Codewort
- Somit ist $s = y \cdot H$ Syndrom des empfangenen Worts, wobei H die Parity-Check-Matrix ist
- Die beste Abschätzung der empfangen Nachricht des Codeworts ist $x = y + z_0$ (z Fehlervektor), wobei z_0 minimal bzgl. der Gleichung $s = z \cdot H$ ist
- Problem: finde $x \in C$ mit minimaler Distanz zwischen y, x
- Im Mittel muss jedoch die gesamte Lösungsmenge für $s = y \cdot H$ durchsucht werden, um die minimale Distanz zu finden
  - Größe des Lösungsraums ist jedoch 2^k
- Ein Algorithmus $\mathcal{A}$ mit gegebener Matrix H und Vektor s soll die Lösung der minimale Gewichtung für $y \cdot H = s$ finden, kann für die Eingabe nur eine Exponentialfunktion sein.

McElice Kodierungsproblem

- Beweis der $\mathcal{NP}$-Härte in [**berlekamp1978inherent**] via Reduktionsbeweis der Entschediugngsprobleme
  - Coset-Weight
  - Subspace-Weight

Fahrplan Code-basierte Kryptografie

Grundlagen
Galoiskörper
Hamming Gewicht und Distanz
Generatormatrix
Parity-Check-Matrix
Lineare Codes
Zyklischer Code
Zyklischer linearer Code
Generatorpolynom
Generatorpolynom
McElice Kodierungsproblem
McElice Kodierungsproblem
McEliece – Code-basierte Kryptografie
McEliece-Kryptosystem
Parameter Definitionen

Grundlegende Idee McEliece Kryptosystem

- Transformiere Klartext m (Message) mithilfe einer Generator-Matrix in allgemeinen Goppa-Code
- Multiplikation mit randomisierten Matrizen führt zu allgemeinem linearen Code
  - Gist: Reihe von Matrix-Multiplikationen ist Verschlüsselung
- Retransformation ohne Matrizen in Goppa-Code ist problemtisch: $\mathcal{NP}$-Hart [SP18]
- Öffentlicher Schlüssel:
  - Beinhaltet Generator-Matrix zur Umwandlung in allg. linearen Code
  - Zusätzlich: Anzahl der maximal einbaubaren Fehler in der Chiffre c
  - Fehler sind also die Anzahl der Bits, die invertiert werden sollen
- Privater Schlüssel: Umwandlung des allgemeinen, linearen Codes in Goppa-Code
  - Für performante Retransformation
  - Und Fehlerkorrektur

Parameter Definitionen

- Systemparameter m gibt die Blockgröße an, für zu verschlüsselnde Nachricht
- C sei ein binärer (n, k) Goppa-Code mit t effizient korrigierbaren Fehlern
- t gibt die maximale Anz. eff. korrigierbarer Fehler durch Goppa-Code C [1]
- Daraus ergeben sich:
  - Blocklänge Chiffretext: $n = 2^m$
  - Nachricht Blocklänge $k = n - m \cdot t$
  - Minimale Hamming-Distanz d des Codes C: $d = 2 \cdot t + 1$

[1] McEliece fixiert $t = 50$, als Maximalwert [McE78]

McEliece als CPA-Sicheres kryptografisches Shema

- Das *McEliece*-Kryptosystem $\Pi := (Gen, Enc, Dec)$
- Wobei:
  - *Gen* Schlüsselerzeugung
  - *Enc* Verschlüsselung
  - *Dec* Entschlüsselung
- Korrekheit: Es muss gelten

$$m = Dec_{priv}(c) = Dec_{priv}(Enc_{pub}(m))$$

Schlüsselerzeugung *Gen*

- Erzeuge Generator-Matrix $G^{k \times n}$ für Goppa-Code C
  - Matrix aus der binärer Klartext mit Länge k die Chiffre der Länge n berechnet werden kann
- Erzeuge zufällige, binäre, nicht singuläre[2] *Scramble-Matrix* $S^{k \times k}$
  - S muss in $GF(2)$ invertierbar sein
- Permutationsmatrix $P^{n \times n}$
  - Binärmatrix, je Zeile genau ein 1 Element enthalten ist
- Berechne: $G'^{k \times n} = S \cdot G \cdot P$
- Schlüssel: $K := (G, S, P, G', t)$
  - Öffentlicher Schlüssel: $K_{pub} := (G', t)$
  - Privater Schlüssel: $K_{priv} := (G, S, P)$

[2] M.a.W. S ist regulär, $\det S \neq 0$; wichtig für Invertierbarkeit

Verschlüsselung *Enc*

- Nachricht in Blöcke, sodass $m \in \mathbb{Z}_2^k$
- Sei $z \in \mathbb{Z}_2^n$ ein belieber Vektor der Länge n, mit maximaler Gewichtung t
  - Gewichtung t: maximale Anzahl Einsen in z
  - Fehlervektor erlaubt es Chiffre an maximal t Stellen zu invertieren
- $Enc_{pub}(m, G', z) = m \cdot G' + z = c$

Entschlüsselung *Dec*

- Berechne $c' = cP^{-1}$
  - $c' = c \cdot P^{-1} = (mG' + z) \cdot P^{-1} = (mG' \cdot P^{-1} + z \cdot P^{-1}) = m(SGP \cdot P^{-1}) + z \cdot P^{-1}$
- Anwenden $decode(c')$ des Goppa-Codes auf c', sodass m' gefunden werden kann
  - Rausrechnen des Fehlervektors z
  - D.h. wir erhalten: $m' = m(SGP \cdot P^{-1}) = m \cdot SG$
  - Hamming-Distanz: $d_H(m'G, c') \leq t$
    - Invertiere mit Generatormatrix G
- Multiplikation mit S^{-1}: $m = m'S^{-1}$
- Kompakt: $dec_{priv}(c) = decode(cP^{-1}) \cdot S^{-1}$

Beispiel McElicece-Kryptosystem

- Kryptosystem (n, k, d) mit Systmeparameter: $n = 7, k = 4, d = 3$
  - 4 Bit Klartext auf 7 Bit Chiffretext
  - Hamming-Distanz $d = 3$
  - Somit lassen sich $t = \frac{d-1}{2} = 1$ Bitfehler korrigieren

Beispiel McElicece-Kryptosystem, cont'

- Schlüsselerzeugung *Gen*: Generator-Matrix erzeugt Hamming-Code statt Goppa-Code
- $G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$

  Da $d = 3$ unterscheidet sich jede Zeile in mindestens drei Werten
- Zufällige Matrizen S und P

$$S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \qquad P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Beispiel McElicece-Kryptosystem, cont'

Berechnung des öffentlichen Schlüssels $G' = S \cdot G \cdot P$:

$$G' = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} =$$

Beispiel McElicece-Kryptosystem, cont'

Berechnung des öffentlichen Schlüssels $G' = S \cdot G \cdot P$:

$$= \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Beispiel McElicece-Kryptosystem, cont'

Der öffentlichen Schlüssels $K_{pub} = (G', t)$:

$$K_{pub} = (G', t) = \left(\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}, 1 \right)$$

Beispiel McElicece-Kryptosystem, cont'

Nachricht $m = (1101)$, Fehlervektor z mit maximalem Gewicht $t = 1$ und Länge $n = 7$: Wähle $z = (0000100)$

$$Enc_{pub}(m, G', z) = c = m \cdot G' + z$$

$$\begin{aligned} m &= \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} = c \end{aligned}$$

Beispiel McElicece-Kryptosystem, cont'

Entschlüsselung der Chiffre:
Invertierung der Permuation $c' = cP^{-1}$

$$c' = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Beispiel McElicece-Kryptosystem, cont'

- Dekodierung des Hamming-Codes:
- Berechne Hamming-Distanz d der Generator-Matrix G: $\begin{pmatrix}1 & 3 & 3 & 2\end{pmatrix}$
- Somit ist $m' = \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}$
- Berechne Klartext m

$$
\begin{aligned}
m = m'S^{-1} &= \\
&= \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix} \cdot \begin{pmatrix}1 & 1 & 0 & 1\\ 1 & 1 & 0 & 0\\ 0 & 1 & 1 & 1\\ 1 & 0 & 0 & 1\end{pmatrix} \\
&= \begin{pmatrix}1 & 1 & 0 & 1\end{pmatrix}
\end{aligned}
$$

Ausblick

- The good news: Es gab keine erfolgreichen Angriffe gegen das eigentliche McEliece-Verfahren
- Verhfahren gilt als *IND-CCA2* [Dot+12] sicher, somit ist es auch *IND-CPA* sicher [Noj+08]
- Angriffe McEliece mit originalen Parametern von 1978 in 1400 Tagen (Einzelne Machine) oder in 7 Tagen mithilfe von 200 CPUs [Bal+16]
- Statisches Angriff auf McEliece [CS98]
- Jedoch:
  - Bruce Schneier: McEliece-Kryptosystem etwa 2 bis 3 mal langsamer als RSA [Sch07, S. 479ff]
    - Leider keine Angabe, wie die Werte zustande kommen
  - Extrem große öffentliche Schlüssel: G' ist Matrix $k \times n$
    - Bei Parameter $(1024, 524, 101)$ ist $k \cdot n = 1024 \cdot 524 = 536576$ Bit also etwa 67$kBytes$
  - Chiffretext ist fast doppelt so groß wie Klartext, aus $524 Bit$ klartext werden zu 1024 Bit Chiffre

Quellen I

[Bal+16] Marco Baldi u. a. „Enhanced public key security for the McEliece cryptosystem“. In: *Journal of Cryptology* 29.1 (2016), S. 1–27.

[CS98] Anne Canteaut und Nicolas Sendrier. „Cryptanalysis of the original McEliece cryptosystem“. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 1998, S. 187–199.

[Dot+12] Nico Dottling u. a. „A CCA2 secure variant of the McEliece cryptosystem“. In: *IEEE Transactions on Information Theory* 58.10 (2012), S. 6672–6680.

[Fau+13] J. Faugère u. a. „A Distinguisher for High-Rate McEliece Cryptosystems“. In: *IEEE Transactions on Information Theory* 59.10 (2013).

[Kun91] Ernst Kunz. *Endliche Körper (Galois-Felder)*. Vieweg+Teubner Verlag, 1991, S. 185–190.

Quellen II

[Lju04] Ivana Ljubic. „Exact and memetic algorithms for two network design problems“. In: *PhD, Technische Universitat Wien, Vienna Austria* (2004).

[McE78] Robert J McEliece. „A public-key cryptosystem based on algebraic“. In: *Coding Thv* 4244 (1978), S. 114–116.

[Noj+08] Ryo Nojima u. a. „Semantic security for the McEliece cryptosystem without random oracles“. In: *Designs, Codes and Cryptography* 49.1-3 (2008), S. 289–305.

[Sch07] Bruce Schneier. *Applied cryptography: protocols, algorithms, and source code in C*. John Wiley & Sons, 2007.

[SP18] Douglas Robert Stinson und Maura Paterson. *Cryptography: theory and practice*. CRC press, 2018.

Genereller Hamming-Code

- **Defintion:** Ein genereller Hamming-Code kann als Parity-Check-Matrix H mit allen möglichen Kombinationen aus Einsen und Nullen dargestellt werden (ausgenommen eine Null-Spalte)
- Matrix hat r Zeilen und $2^r - 1$ Spalten
- r dieser Spalten sind für die Paritäts-Bits, da es $2^r - 1 - r = k$

(n,k)-Code

(n,k,d)-Code

- Längere Codes werden Effizienter hinsichtlich der Fehlerkorrektur