Aufgabe 4: Zara Zackigs Zurückkehr

Teilnahme-ID: 62454

Bearbeiter dieser Aufgabe:
Philip Gilde

12. April 2022

Inhaltsverzeichnis

1 Lösungsidee

Von den gegebenen n Karten mit jeweils m Bits werden p Karten gesucht, so dass das exklusive Oder (im Folgenden als XOR abgekürzt) von $p-1$ Karten gleich der p. Karte ist.

$$\begin{aligned} & k_1 \oplus k_2 \oplus \ldots \oplus k_{p-1} = k_p && | \oplus k_p \\ \leftrightarrow \quad & k_1 \oplus k_2 \oplus \ldots \oplus k_{p-1} \oplus k_p = 0 \end{aligned}$$

Diese Gleichung lässt sich zu jeder der p Karten umstellen. Es werden also p Karten gesucht, deren XOR gleich einer Karte mit m Nullen ist. Dieses Problem lässt sich umformulieren zu einem linearen Gleichungssystem im Galois-Feld $GF(2)$. Dieses besteht nur aus den beiden Elementen 0 und 1. Die Addition im Feld entspricht dem XOR, die Multiplikation einem UND. Die gemischten Karten entsprechen der Matrix $K \in GF(2)^{n \times m}$. K_n ist dabei die n-te Karte und $K_{n,m}$ das m-te Bit der n-ten Karte. Gesucht wird der Vektor $v \in GF(2)^n$, so dass dieses lineare Gleichungssystem gilt:

$$\begin{aligned} K_{1,1}v_1 + K_{2,1}v_2 + \ldots + K_{n,1}v_n &= 0 \\ K_{1,2}v_1 + K_{2,2}v_2 + \ldots + K_{n,2}v_n &= 0 \\ &\ldots \\ K_{1,m}v_1 + K_{2,m}v_2 + \ldots + K_{n,m}v_n &= 0 \end{aligned}$$

In Matrixform:

$$K^T \cdot v = 0$$

Dabei bestimmt v_n, ob die n-te Karte zu den gesuchten Karten gehört.
Die Menge von Vektoren, die sich oben für v einsetzen lassen, wird als Nullraum oder Kern der Matrix K^T bezeichnet. In $GF(2)$ kann ein Vektor nicht skaliert werden, weil er nur mit entweder 0 oder 1 multipliziert werden kann. Somit besteht der Nullraum aus allen möglichen Kombinationen von Summen

der Basisvektoren. Wenn r der Rang von K^T ist, dann ist $q = n - r$ die Anzahl der Basisvektoren des Nullraums. Der Rang von K^T ist die Anzahl linear unabhängiger Zeilen beziehungsweise Spalten (diese beiden Werte sind gleich). Wenn, wie in der ursprünglichen Aufgabe, $n < m$, dann ist der Rang in der Regel $n - 1$, denn nur eine Karte, die Wiederherstellungskarte, ist linear abhängig von den anderen.
Die Wahrscheinlichkeit, dass h Karten mit jeweils b Bits voneinander linear unabhängig sind, ist, wenn diese zufällig und erzeugt sind und Nullen und Einsen gleich wahrscheinlich sind, wovon der Einfachheit halber ausgegangen wird, nach [1] gegeben durch

$$p_h = \prod_{i=1}^{h} (1 - 2^{i-1-b})$$

Diese ist für $h = n - 1 = 110$ und $b = m = 128$ hoch genug, um den anderen Fall zunächst außen vor zu lassen. Somit sind alle bis auf eine der 111 Karten linear unabhängig voneinander. Der Nullraum besteht damit aus nur $q = n - (n - 1) = n - n + 1 = 1$ Vektor. Dieser muss an den Stellen der 11 echten Karten 1 und sonst überall 0 sein. Somit haben wir die echten Karten gefunden.
Um die Basisvektoren zu finden, muss man $K^T \cdot v = 0$ zunächst mit dem Gauß-Jordan-Algorithmus in Stufenform bringen. Aus dieser Stufenform kann man die freien Variablen des Gleichungssystems ablesen. Die Basisvektoren des Nullraums lassen sich finden, in dem man nacheinander jede der freien Variablen auf 1 und alle anderen auf 1 setzt und dann das daraus resultierende Gleichungssystem wiederum mit dem Gauß-Jordan-Algorithmus auf Stufenform bringt. Dann lässt sich der Basisvektor auf der rechten Seite der Gleichung ablesen. Dieses Verfahren soll an einem Beispiel erklärt werden:

$$K = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$K^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$K^T \cdot v = 0$$

Ich verwende im Folgenden der Einfachheit halber die erweiterte Koeffizientenmatrix, welche für dieses Gleichungssystem so aussieht:

$$(K^T|0) = \left[\begin{array}{ccccccc|c} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}\right]$$

Der "klassische"Gauß-Jordan-Algorithmus basiert darauf, dass die sich die Lösungsmenge nicht verändert, wenn man im Gleichungssystem eine Zeile zu einer anderen addiert, oder eine Zeile mit einem Wert multipliziert. In $GF(2)$ gilt das selbe: Man kann eine Zeile zu einer anderen addieren (beziehunsweise XOR-en), ohne die Lösungsmenge zu verändern, weil eine Gleichung ja auf beiden Seiten den gleichen Wert hat. Um das Gleichungssystem auf Stufenform zu bringen, muss in möglichst vielen Reihen der erste Wert in seiner Spalte allein sein. Als erstes soll in der ersten Spalte nur eine eins stehen, und zwar in der ersten Reihe. Dafür wird die erste Reihe zu allen Reihen addiert, deren erster Wert nicht Null ist, denn

dadurch wird aus diesen eine Null:

$$\left[\begin{array}{ccccccc|c} 1&0&1&0&1&1&0&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0\\ 1&1&1&1&0&0&0&0\\ 1&0&0&1&0&1&1&0\\ 0&1&0&0&1&1&0&0\\ 1&0&1&1&0&0&0&0 \end{array}\right] \rightarrow \left[\begin{array}{ccccccc|c} 1&0&1&0&1&1&0&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0\\ 0&1&0&1&1&1&0&0\\ 0&0&1&1&1&0&1&0\\ 0&1&0&0&1&1&0&0\\ 0&0&0&1&1&1&0&0 \end{array}\right]$$

In der zweiten Zeile soll die eins in der zweiten Spalte allein stehen. Dafür wird genauso verfahren:

$$\left[\begin{array}{ccccccc|c} 1&0&1&0&1&1&0&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0\\ 0&1&0&1&1&1&0&0\\ 0&0&1&1&1&0&1&0\\ 0&1&0&0&1&1&0&0\\ 0&0&0&1&1&1&0&0 \end{array}\right] \rightarrow \left[\begin{array}{ccccccc|c} 1&0&1&0&1&1&0&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0\\ 0&0&0&1&0&1&1&0\\ 0&0&1&1&1&0&1&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&1&1&1&0&0 \end{array}\right]$$

Für die restlichen Reihen wird genauso verfahren:

$$\left[\begin{array}{ccccccc|c} 1&0&1&0&1&1&0&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0\\ 0&0&0&1&0&1&1&0\\ 0&0&1&1&1&0&1&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&1&1&1&0&0 \end{array}\right] \rightarrow \left[\begin{array}{ccccccc|c} 1&0&0&0&0&1&1&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0\\ 0&0&0&1&0&1&1&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&1&1&1&0&0 \end{array}\right] \rightarrow \left[\begin{array}{ccccccc|c} 1&0&0&0&0&1&1&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0\\ 0&0&0&1&0&1&1&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&0&1&0&1&0 \end{array}\right]$$

In der fünften Reihe ist das Element in der fünften Spalte Null. Um daraus eine 1 zu machen, wird eine beliebige darunterliegende Reihe, die eine Eins an an fünfter Stelle hat, zur fünften addiert.

$$\left[\begin{array}{ccccccc|c} 1&0&0&0&0&1&1&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0\\ 0&0&0&1&0&1&1&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&0&1&0&1&0 \end{array}\right] \rightarrow \left[\begin{array}{ccccccc|c} 1&0&0&0&0&1&1&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0\\ 0&0&0&1&0&1&1&0\\ 0&0&0&0&1&1&0&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&0&1&0&1&0 \end{array}\right]$$

Dann wird wie vorher weitergemacht:

$$\left[\begin{array}{ccccccc|c} 1&0&0&0&0&1&1&0\\ 0&1&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0\\ 0&0&0&1&0&1&1&0\\ 0&0&0&0&1&1&0&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&0&1&0&1&0 \end{array}\right] \rightarrow \left[\begin{array}{ccccccc|c} 1&0&0&0&0&1&1&0\\ 0&1&0&0&0&1&1&0\\ 0&0&1&0&0&1&1&0\\ 0&0&0&1&0&1&1&0\\ 0&0&0&0&1&1&0&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&0&0&1&1&0 \end{array}\right] \rightarrow \left[\begin{array}{ccccccc|c} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&1&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&0&0&0&0&0 \end{array}\right]$$

Die Koeffizientenmatrix ist jetzt also in Stufenform. Die einzige freie Variable ist v_7. Diese wird auf 1 gesetzt und die Matrix auf Stufenform gebracht:

$$\left[\begin{array}{ccccccc|c} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&1&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&0&0&0&0&0 \end{array}\right] \rightarrow \left[\begin{array}{ccccccc|c} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&1&0\\ 0&0&0&0&0&1&1&0\\ 0&0&0&0&0&0&1&1 \end{array}\right] \rightarrow \left[\begin{array}{ccccccc|c} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&1\\ 0&0&0&0&0&1&0&1\\ 0&0&0&0&0&0&1&1 \end{array}\right]$$

Wie man ablesen kann, sind v_5, v_6, v_7 alle 1. Somit sind die drei Öffnungskarten die letzten drei:

$$\begin{array}{ccccccc} 1&1&1&0&0&1&0\\ 1&0&0&0&1&1&0\\ 0&1&1&0&1&0&0 \end{array}$$

Wie man sieht, ist diese Lösung korrekt, denn jede Karte ist das XOR der jeweils anderen beiden.

Wenn $q > 1$ ist, kann es sein, dass keiner der Basisvektoren des Nullraums 11 Einsen beinhaltet, sondern eine Linearkombination dieser. In diesem Fall wird jede der 2^q Kombinationen der Basisvektoren durchprobiert, bis eine davon 11 Eisen enthält.

Die richtige Karte für das s-te Haus kann gefunden werden, in dem man die Karten aufsteigend sortiert und zuerst die s-te und dann die $s+1$-te Karte ausprobiert. Wenn die Sicherungskarte kleiner als die Öffnungskarte des s-ten Hauses ist, liegt sie davor im Stapel und die Öffnungskarte an der Stelle $s+1$. Wenn sie größer ist, dann liegt sie dahinter im Stapel und die Öffnungskarte an Stelle s.

Das Verfahren stößt an seine Grenzen, wenn $n > m$ ist. Der Rang von K^T ist dann nämlich höchstens m, wodurch der Nullraum $q = n - m$ Basisvektoren hat. Diese Basisvektoren müssen nun nicht zwangsweise einen mit 11 Einsen beinhalten, dieser könnte auch eine Linearkombination der anderen Basisvektoren sein. Wenn das der Fall ist, müsste man alle 2^q Kombinationen von Basisvektoren durchprobieren, bis eine davon 11 Einsen beinhaltet, was für die Beispieleingabe auf der BwInf-Webseite mit 161 Karten zu je 128 Bit $2^q = 2^{n-m} = 2^{161-128} = 2^{33} = 8.589.934.592$ Kombinationen sind. Diese lassen sich nicht wirklich in überschaubarer Zeit durchprobieren.

2 Umsetzung

Die Lösung wurde in Python umgesetzt. Dabei wurde die Bibliothek `NumPy` verwendet, um die Koeffizientenmatrizen darzustellen.

3 Laufzeit

Ein Durchlauf des Gauß-Jordan-Algorithmus hat eine Laufzeit von $\mathcal{O}(n^3)$. Der Algorithmus wird einmal ursprünglich und dann für jeden der q Basisvektoren des Nullraums durchgeführt, somit ist die Laufzeit $\mathcal{O}((q+1)n^3)$

4 Beispiele

5 Quellcode

Literatur

[1] A. F. (https://math.stackexchange.com/users/54227/alfonso fernandez), "Probability that a random binary matrix will have full column rank?." Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/564699 (version: 2013-11-12).