RUHR-UNIVERSITÄT BOCHUM
LEHRSTUHL FÜR KRYPTOLOGIE UND IT-SICHERHEIT
Prof. Dr. Alexander May
Elena Kirshanova, Ilya Ozerov

Hausübungen zur Vorlesung

Kryptanalyse

SS 2014

Blatt 4 / 14. Mai 2014
Abgabe: 22. Mai 2014, 14.00 Uhr, Kasten NA/02

AUFGABE 1 (5 Punkte)**:**

Wir betrachten ein ± 1 Subset Sum Problem. Sei $n \in 8\mathbb{N}$. Gegeben sind $a_1, \ldots, a_n \in \mathbb{N}$ und $S \in \mathbb{N}$. Gesucht sind $I, J \subseteq \{1, \ldots, n\}$ mit $I \cap J = \emptyset$, $|I| = |J| = \frac{n}{4}$ und $\sum_{i \in I} a_i - \sum_{i \in J} a_i = S$. Sei $m \in 8\mathbb{N}$. Wie in der Präsenzübung betrachten wir die Gleichung

$$\sum_{i \in I_1} a_i - \sum_{i \in J_1} a_i = S - \left(\sum_{i \in I_2} a_i - \sum_{i \in J_2} a_i \right)$$

mit $I_1, I_2, J_1, J_2 \subseteq \{1, \ldots, n\}$, $I_1 \cap J_1 = \emptyset$, $I_2 \cap J_2 = \emptyset$, $|I_1| = |I_2| = |J_1| = |J_2| = \frac{n+m}{8}$. Sei nun $S = \sum_{i \in I} a_i - \sum_{i \in J} a_i$ mit $I, J \subseteq \{1, \ldots, n\}, I \cap J \neq \emptyset, |I| = |J| = \frac{n}{4}$. Wie viele Repräsentationen $S = S_1 + S_2$ mit $S_1 = \sum_{i \in I_1} a_i - \sum_{i \in J_1} a_i$ und $S_2 = \sum_{i \in I_2} a_i - \sum_{i \in J_2} a_i$ existieren? Begründen Sie!

AUFGABE 2 (5 Punkte)**:**

Wir betrachten, wie in der Präsenzübung, die Anwendung der Repräsentationstechnik auf das Syndrom-Dekodierproblem. Seien $t \in \mathbb{N}$, $m, n \in 8\mathbb{N}$. Gegeben sind $a_1, \ldots, a_n, s \in \mathbb{F}_2^t$. Gesucht ist ein $I \subseteq \{1, \ldots, n\}$ mit $|I| = \frac{n}{4}$ und $\sum_{i \in I} a_i = s$. Seien $I_1, I_2 \subseteq \{1, \ldots, n\}$, $|I_1| = |I_2| = \frac{n+m}{8}$. Wie viele Repräsentationen $s = s_1 + s_2$ mit $s_1 = \sum_{i \in I_1} a_i$ und $s_2 = \sum_{i \in I_2} a_i$ existieren? Begründen Sie!

Hinweis: In $\mathbb{F}_2$ gilt $1 + 1 = 0$.

Bitte wenden!

AUFGABE 3 (5 Punkte)**:**

Implementieren Sie eine Variante des Meet in the Middle Angriffs auf Subset Sum (Folie 52) in sage. Für ein $n \in 20\mathbb{N}$ sind $a_1, \ldots, a_n, s \in \mathbb{N}$ gegeben. Gesucht ist ein $I \subseteq \{1, \ldots, n\}$ mit (anders als in den Folien) $|I| = \frac{n}{10}$. Sie müssen in diesem Fall folglich in $I_1 \subseteq \{1, \ldots, \frac{n}{2}\}$ und $I_2 \subseteq \{\frac{n}{2}+1, \ldots, n\}$ mit $|I_1| = |I_2| = \frac{n}{20}$ aufsplitten. In der Datei subsetsum.txt finden Sie eine Instanz mit $n = 80$. Nehmen Sie an, dass es I_1, I_2 mit der angegebenen Anzahl von Einsen bereits gibt. Eine initiale Permutation (Schritt 1 des Algorithmus) ist deshalb nicht mehr nötig. Was sind die gesuchten Mengen I_1 und I_2 (beachten Sie, dass sage ab 0 zählt)? Geben Sie den Quelltext mit ab.

Hinweis: Eine Menge aller $I_1 \subseteq \{1, \ldots, \frac{n}{2}\}$ mit $|I| = \frac{n}{20}$ kann mit Subsets(range(0, n/2), n/20) angelegt werden. Sage zählt hierbei wie gewohnt ab 0.

AUFGABE 4 (5 Punkte)**:**

Wir betrachten eine Variante des Information Set Decoding Algorithmus zum Dekodieren von zufälligen, linearen Codes (Folie 58). Wir wählen dabei $p = 0$, d.h. es werden so lange zufällige Permutationen gewählt, bis *alle* Einsen im hinteren Teil des gesuchten Vektors liegen. Bei Eingabe einer Matrix $P \in \mathbb{F}_2^{n-k \times n}$, eines Spaltenvektors $s \in \mathbb{F}_2^{n-k}$ und einer Anzahl von Einsen $\omega \in \mathbb{N}$ soll der Algorithmus einen Spaltenvektor $e \in \mathbb{F}_2^n$ finden mit $Pe = s$ und $\mathrm{wt}(e) = \omega$. Der Algorithmus arbeitet dabei wie folgt:

(1) Wähle zufällige Permutationsmatrix $U_P \in \mathbb{F}_2^{n \times n}$, berechne $P' = PU_P$.

(2) Spalte auf in $P' = (P''|U_G)$ mit $U_G \in \mathbb{F}_2^{n-k \times n-k}$. Berechne $s_2 = U_G^{-1} s$.

(3) Teste, ob $\mathrm{wt}(s_2) = \omega$. Falls ja, gib $e = U_P \binom{0}{s_2}$ zurück. Sonst gehe zurück zu (1).

(a) Zeigen Sie, dass die Ausgabe e das Problem löst, d.h. $\mathrm{wt}(e) = \omega$ und $Pe = s$ erfüllt.

(b) Implementieren Sie den Algorithmus in sage und lösen Sie die Instanz in codes.txt, d.h. finden Sie e. Geben Sie den Quelltext mit ab.

Hinweis: In Schritt 2 des Algorithmus muss U_G invertiert werden. Es ist jedoch so, dass U_G nur mit konstanter Wahrscheinlichkeit invertierbar ist. Fangen Sie eine mögliche Exception mit try : ... except : ... ab. Sie können eine zufällige $n \times n$ Permutationsmatrix beispielsweise mit UP = M(Permutations(n).random_element().to_matrix()) für M = MatrixSpace(GF(2), n, n) erzeugen. Die Matrix UG kann mit UG.inverse() invertiert werden.