Institut für Informatik
Prof. Dr. J. Rothe

Sommersemester 2024
Übungsblatt 8

Lösungsvorschläge

Kryptokomplexität II

Bearbeitungszeit: 28. Mai bis 7. Juni
Verantwortlich: Roman Zorn

Aufgabe 1: NP-Vollständigkeitsbeweis von SOS

Gegeben sei die X-3-Cover-Instanz $\langle U, \mathcal{S} \rangle$ mit

$$U = \{1, ..., 9\},$$
$$\mathcal{S} = \{S_1, ..., S_5\},$$
$$S_1 = \{1, 4, 5\}, S_2 = \{5, 6, 7\}, S_3 = \{7, 8, 9\}, S_4 = \{1, 2, 3\}, S_5 = \{1, 3, 8\}.$$

In der Vorlesung wurde gezeigt, dass SOS NP-vollständig ist. Vollziehen Sie den Beweis nach.

- Zeigen Sie zunächst, dass SOS in NP ist.
- Führen Sie anschließend die Reduktion von X-3-Cover auf SOS durch.
- Argumentieren Sie abschließend über die SOS-Instanz, ob die gegebene X-3-Cover-Instanz eine Ja- oder Nein-Instanz ist.

Lösungsvorschlag:
Zu zeigen: SOS $\in$ NP.
Angenommen wir kennen eine Lösung der SOS-Instanz, d.h. wir wissen einen Vektor $\vec{x} \in \{0, 1\}^n$. Dann lässt sich in Polynomialzeit verifizieren, dass $\sum_{i=1}^{n} x_i s_i = T$.

Zu zeigen: SOS ist NP-schwer.
Wir beginnen nun mit der Konstruktion der SOS-Instanz aus der X-3-Cover-Instanz.
Für jedes $S_i \in \mathcal{S}$ konstruieren wir einen Vektor $\vec{s}_i$.

$$S_1 = \{1, 4, 5\} \Rightarrow \vec{s}_1 = (1, 0, 0, 1, 1, 0, 0, 0, 0)$$
$$S_2 = \{5, 6, 7\} \Rightarrow \vec{s}_2 = (0, 0, 0, 0, 1, 1, 1, 0, 0)$$
$$S_3 = \{7, 8, 9\} \Rightarrow \vec{s}_3 = (0, 0, 0, 0, 0, 0, 1, 1, 1)$$
$$S_4 = \{1, 2, 3\} \Rightarrow \vec{s}_4 = (1, 1, 1, 0, 0, 0, 0, 0, 0)$$
$$S_5 = \{1, 3, 8\} \Rightarrow \vec{s}_5 = (1, 0, 1, 0, 0, 0, 0, 1, 0)$$

Da $|\mathcal{S}| = 5$ wählen wir als Basis $b = 6$ und bestimmen für jedes $\vec{s}_i$ die Zahl

$$s_i = \sum_{j=1}^{3m} b^{3m-j} \cdot \vec{s}_i[j] = \sum_{j=1}^{9} 6^{9-j} \cdot \vec{s}_i[j]$$

wobei $\vec{s}_i[j]$ die j'te Stelle im Vektor $\vec{s}$ ist.

$$\begin{aligned}
\vec{s}_1 &= (1,0,0,1,1,0,0,0,0) \Rightarrow s_1 = 6^8 + 6^5 + 6^4 = 1688688 \\
\vec{s}_2 &= (0,0,0,0,1,1,1,0,0) \Rightarrow s_2 = 6^4 + 6^3 + 6^2 = 1548 \\
\vec{s}_3 &= (0,0,0,0,0,0,1,1,1) \Rightarrow s_3 = 6^2 + 6^1 + 6^0 = 43 \\
\vec{s}_4 &= (1,1,1,0,0,0,0,0,0) \Rightarrow s_4 = 6^8 + 6^7 + 6^6 = 2006208 \\
\vec{s}_5 &= (1,0,1,0,0,0,0,1,0) \Rightarrow s_5 = 6^8 + 6^6 + 6^1 = 1726278
\end{aligned}$$

Es ist $T = \sum_{j=0}^{8} 6^j = 2015539$. Damit ergibt sich die SOS-Instanz $\langle s_1, s_2, s_3, s_4, s_5, s_6, T\rangle$. Nun gilt $\langle U, \mathcal{S}\rangle$ ist eine JA-Instanz für X3C, genau dann, wenn $\langle s_1, s_2, s_3, s_4, s_5, s_6, T\rangle$ eine JA-Instanz für SOS ist.

Es ist leicht zu sehen, dass wir nur genau eine der Zahlen s_1, s_4 und s_5 wählen dürfen. Wählen wir zwei, so ist die Summe größer als T. Die Summen $s_1 + s_2 + s_3$, $s_4 + s_2 + s_3$ und $s_5 + s_2 + s_3$ sind aber zu klein. Also ist es nicht möglich genau T als Summe zu erreichen. Damit ist es eine NEIN-Instanz für SOS und somit muss die gegebene X3C-Instanz auch eine NEIN-Instanz sein.

Aufgabe 2: Superwachsende Folgen in SOS

Zeigen Sie den Fakt aus der Vorlesung: Für Instanzen $\langle \vec{s}, T\rangle$ mit einer superwachsenden (superincreasing) Folge $\vec{s}$ kann das Problem SOS in deterministischer Polynomialzeit gelöst werden.

▶ Geben Sie dazu einen deterministischen Algorithmus an, der SOS für diesen speziellen Fall in Polynomialzeit löst.

▶ Beweisen Sie, dass Ihr Algorithmus korrekt ist.

Lösungsvorschlag:
Folgender Algorithmus löst SOS für superwachsende Funktionen in Polynomialzeit.

```
solveSOS (⟨s_1, ..., s_n, T⟩){
    t := T;
    x⃗ := (x_1, ..., x_n) = (0, ..., 0);
    for (i = n, i ≥ 1, i − −){
        if (s_i ≤ t){
            x_i := 1;
            t := t − s_i;
```

```
        }
    }
    if (t ≠ 0){
        return NEIN-Instanz;
    }
    return JA-Instanz mit Lösung x⃗;
}
```

Beweis für Korrektheit:
Zunächst zeigen wir, dass alle ausgewählten s_i (also jene mit $x_i = 1$) auch tatsächlich benötigt werden. Wegen der superwachsenden Folge gilt $\sum_{i=1}^{k-1} s_i < s_k$. Wenn also zu einem Zeitpunkt k in der Schleife $s_k < t$, so *muss* s_k in die Summe aufgenommen werden, weil man sogar mit allen folgenden $s_1, ..., s_{k-1}$ zusammen t sonst nicht erreichen könnte. Danach wird t entsprechend um s_k erniedrigt und man kann das selbe Argument induktiv auf s_{k-1} anwenden.

Nun zeigen wir noch, dass alle s_i, die wir nicht auswählen auch tatsächlich nicht benötigt werden. Wir wählen nur s_i *nicht* aus, wenn $s_i > t$ gilt. Da alle s_i positiv sind, würde das auswählen eines solchen s_i dazu führen, dass wir am Ende eine Summe größer als T erhalten. Also dürfen wir diese s_i nicht auswählen.

Aufgabe 3: Rivest-Sherman-Schlüsselaustausch und -Signatur I

Definition. Eine zweistellige, totale Funktion $f : A \times A \to A$ heißt *assoziativ*, wenn für alle $x, y, z \in A$ gilt, dass $f(x, f(y, z)) = f(f(x, y), z)$ (bzw. in Infixnotation: $xf(yfz) = (xfy)fz$).

Betrachten Sie das Rivest-Sherman-Protokoll zum Schlüsselaustausch. Gegeben sei die stark nichtinvertierbare, assoziative, totale Einwegfunktion $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$.
► Modifizieren Sie das Schlüsselaustauschprotokoll so, dass ein Protokoll für digitale Signaturen entsteht. Geben Sie das Protokoll explizit an.
► Zeigen Sie, dass die Verifikationsbedingung von Ihrem Protokoll erfüllt wird.

Lösungsvorschlag: [a]

Alice		Bob
		wählt zwei unterschiedliche, zufällige $x_B, y_B \in \Sigma^*$, hält x_B geheim und berechnet $\beta = \sigma(x_B, y_B)$
	$\xleftarrow{\langle y_B, \beta \rangle}$	
		signiert eine Nachricht m via $\text{SIG}_B(m) = \sigma(m, x_B)$
	$\xleftarrow{\langle m, \text{SIG}_B(m) \rangle}$	
Verifiziert Bobs Signatur, indem sie überprüft, ob $\sigma(\text{SIG}_B(m), y_B) = \sigma(m, \beta)$ gilt.		

Da σ stark nichtinvertierbar ist, kann man aus y_B und β nicht ohne weiteres x_B berechnen, d.h. nur Bob kennt x_B und kann Nachrichten signieren.

Beim Verifizieren gilt $\sigma(\text{SIG}_B(m), y_B) = \sigma(\sigma(m, x_B), y_B) = \sigma(m, \sigma(x_B, y_B)) = \sigma(m, \beta)$ wegen der Assoziativität von σ.

[a] Rabi, M; Sherman, A. *An Observation on Associative One-Way Functions in Complexity Theory.*

Aufgabe 4: Rivest-Sherman-Schlüsselaustausch und -Signatur II

Definition. Eine zweistellige, totale Funktion $f : A \times A \to A$ heißt *kommutativ*, wenn für alle $x, y \in A$ gilt, dass $f(x, y) = f(y, x)$ (bzw. in Infixnotation: $xfy = yfx$).

Betrachten Sie erneut das Rivest-Sherman-Protokoll zum Schlüsselaustausch. Sei $\sigma : \Sigma^* \times \Sigma^* \to \Sigma^*$ nun eine totale, stark nichtinvertierbare, assoziative und kommutative Einwegfunktion.

▶ Modifizieren Sie mittels der Funktion σ das Schlüsselaustauschprotokoll von Rivest-Sherman zu einem Protokoll mit beliebig vielen Parteien.

Lösungsvorschlag: Seien $P = \{1, ..., k\}$ die Parteien die am Protokoll teilnehmen.

- Eine beliebige Partei wählt ein zufälliges $y \in \Sigma^*$ das dann allen Parteien bekannt gemacht wird.
- Jede Partei $i \in P$, wählt dann zufällig ein $x_i \in \Sigma^*$, hält es geheim und berechnet $\beta_i = x_i \sigma y$.
- Alle β_i werden veröffentlicht.

- Jede Partei i kann jetzt den gemeinsamen Schlüssel

$$\begin{aligned} & x_i \sigma \beta_1 \sigma \ldots \sigma \beta_{i-1} \sigma \beta_{i+1} \sigma \ldots \sigma \beta_k \\ = \; & x_i \sigma x_1 \sigma y \sigma \ldots \sigma x_{i-1} \sigma y \sigma x_{i+1} \sigma y \sigma \ldots \sigma x_k \sigma y \end{aligned}$$

  berechnen.

Da σ kommutativ und assoziativ ist, ist der Schlüssel für alle Parteien gleich.