Das Generalized Birthday Problem

Problem Birthday

Gegeben: Listen L_1, L_2 mit Elementen aus $\mathbb{F}_2^n$
Gesucht: $x_1 \in L_1$ und $x_2 \in L_2$ mit $x_1 + x_2 = \mathbf{0}$ in $\mathbb{F}_2^n$

Anwendungen:

- Meet-in-the-Middle Angriffe (z.B. für RSA, ElGamal)
- Kennen Lösung für $|L_1| = |L_2| = 2^{\frac{n}{2}}$ in Zeit $\tilde{\mathcal{O}}(2^{\frac{n}{2}})$.

Problem Generalized Birthday

Gegeben: Listen $L_1, \ldots, L_k$ mit Elementen aus $\mathbb{F}_2^n$, unabhängig und gleichverteilt gezogen
Gesucht: $x_1 \in L_1, \ldots, x_k \in L_k$ mit $x_1 + \ldots + x_k = \mathbf{0}$ in $\mathbb{F}_2^n$

- Listen können auf beliebige Länge erweitert werden.
- Wir erwarten die Existenz einer Lösung sobald $|L_1| \cdot \ldots \cdot |L_k| > 2^n$.

Zusammenfügen zweier Listen

Definition Join-Operator

Wir bezeichnen mit $\mathrm{low}_\ell(x)$ die ℓ niederwertigsten Bits von x. Wir definieren für zwei Listen L_1, L_2 den Join-Operator

$$L_1 \bowtie_\ell L_2 = \{(x_1, x_2, x_1 + x_2) \in L_1 \times L_2 \times \mathbb{F}_2^n \mid \mathrm{low}_\ell(x_1) = \mathrm{low}_\ell(x_2)\}.$$

Eigenschaften:

- Es gilt $\mathrm{low}_\ell(x_1 + x_2) = \mathbf{0}$ gdw $\mathrm{low}_\ell(x_1) = \mathrm{low}_\ell(x_2)$.
- Bei Eingabe L_1, L_2 kann $L_1 \bowtie L_2$ berechnet werden in Zeit
$$\tilde{\mathcal{O}}(\max\{|L_1|, |L_2|, |L_1| \bowtie_\ell |L_2|\}).$$
- Es gilt $x_1 + x_2 = x_3 + x_4$ gdw $x_1 + x_2 + x_3 + x_4 = \mathbf{0}$.
- Falls $\mathrm{low}_\ell(x_1 + x_2) = \mathbf{0}$ und $\mathrm{low}_\ell(x_3 + x_4) = \mathbf{0}$, dann gilt
$$\mathrm{low}_\ell(x_1 + x_2 + x_3 + x_4) = \mathbf{0} \text{ und}$$
$$\mathrm{Ws}[x_1 + x_2 + x_3 + x_4 = \mathbf{0} \mid \mathrm{low}_\ell(x_1 + x_2 + x_3 + x_4) = \mathbf{0}] = \frac{1}{2^{n-\ell}}.$$

Algorithmus für das 4-Listen Problem

Algorithmus 4-Listen Problem

EINGABE: L_1, L_2, L_3, L_4 der Länge $|L_i| = 2^{\frac{n}{3}}$ mit Elementen aus $\mathbb{F}_2^n$

1. Setze $\ell := \frac{n}{3}$.
2. Berechne $L_{12} = L_1 \bowtie_\ell L_2$ und $L_{34} = L_3 \bowtie_\ell L_4$.
3. Berechne $L_{1234} = L_{12} \bowtie_n L_{34}$.

AUSGABE: Elemente von L_{1234}

Korrektheit des 4-Listen Problem Algorithmus

Korrektheit:

- Elemente von L_{12}, L_{34} erfüllen $\mathrm{low}_{\frac{n}{3}}(x_1 + x_2) = \mathrm{low}_{\frac{n}{3}}(x_3 + x_4) = \mathbf{0}$.
- Wir erwarten Listenlänge $E[|L_{12}|] = \sum_{(x_1,x_2)\in L_1 \times L_2} \mathrm{Ws}[\mathrm{low}_{\frac{n}{3}}(x_1 + x_2) = \mathbf{0}] = \frac{|L_1|\cdot|L_2|}{2^{\frac{n}{3}}} = 2^{\frac{n}{3}}$.
- Analog gilt $E[|L_{34}|] = 2^{\frac{n}{3}}$.
- Elemente von L_{1234} erfüllen $x_1 + x_2 + x_3 + x_4 = \mathbf{0}$.
- Die erwartete Listenlänge $E[|L_{1234}|]$ von L_{1234} ist

$$\sum_{(x_1,\ldots,x_4)\in L_{12} \times L_{34}} \mathrm{Ws}[x_1 + \ldots + x_4 = \mathbf{0} \mid \mathrm{low}_{\frac{n}{3}}(x_1 + x_2) = \mathrm{low}_{\frac{n}{3}}(x_3 + x_4)]$$
$$= \frac{E(|L_{12}|)\cdot E(|L_{34}|)}{2^{\frac{2n}{3}}} = 1.$$

- D.h. wir erwarten, dass L_{1234} eine Lösung enthält.

Laufzeitanalyse des 4-Listen Problem Algorithmus

Laufzeit und Speicherplatz:

- Die Listen $L_1, \ldots, L_4, L_{12}, L_{34}$ benötigen jeweils Platz $\tilde{\mathcal{O}}(2^{\frac{n}{3}})$.
- Die Konstruktion von L_{12}, L_{34} geht in Laufzeit $\tilde{\mathcal{O}}(2^{\frac{n}{3}})$.
- Konstruktion von L_{1234} benötigt ebenfalls Laufzeit $\tilde{\mathcal{O}}(2^{\frac{n}{3}})$.
- **Gesamt:** Zeit und Platz $\tilde{\mathcal{O}}(2^{\frac{n}{3}})$

Übungen: Modifizieren Sie den Algorithmus, so dass

- $\mathrm{low}_\ell(x_1 + x_2) = \mathrm{low}_\ell(x_3 + x_4) = c$ für ein $c \in \mathbb{F}_2^\ell$.
- wir $x_1 + x_2 + x_3 + x_4 = c'$ für ein $c' \in \mathbb{F}_2^n$ lösen können.
- wir jede Instanz mit $k \geq 4$ in Zeit und Platz $\tilde{\mathcal{O}}(2^{\frac{n}{3}})$ lösen können.

4-Listen Problem in $\mathbb{Z}_{2^n}$

Ziel: Verwende Gruppe $(\mathbb{Z}_{2^n}, +)$ statt $(\mathbb{F}_{2^n}, +)$.

Sei $-L = \{-x \in \mathbb{Z}_{2^n} \mid x \in L\}$.

Algorithmus 4-Listen Problem

EINGABE: L_1, L_2, L_3, L_4 mit Elementen aus $\mathbb{Z}_{2^n}$ der Länge $|L_i| = 2^{\frac{n}{3}}$

1. Setze $\ell := \frac{n}{3}$.
2. Berechne $L_{12} = L_1 \bowtie_\ell -L_2$ und $L_{34} = L_3 \bowtie_\ell -L_4$.
3. Berechne $L_{1234} = L_{12} \bowtie_n -L_{34}$.

AUSGABE: Elemente von L_{1234}

Korrektheit:

- Wir erhalten $(x_1, x_2, x_1 + x_2) \in L_{12}$ mit $x_1 + x_2 = 0 \bmod 2^\ell$.
- Man beachte: Für $x_1 + x_2 = 0 \bmod 2^\ell$ und $x_3 + x_4 = 0 \bmod 2^\ell$ gilt

$$x_1 + x_2 + x_3 + x_4 = 0 \bmod 2^\ell.$$

Algorithmus k-Listen Problem, $k = 2^m$

Algorithmus k-Listen Problem

EINGABE: $L_1, \ldots, L_{2^m}$ mit Elementen aus $\mathbb{F}_2^n$, Länge $|L_i| = 2^{\frac{n}{m+1}}$

1. Setze $\ell := \frac{n}{m+1}$.
2. For $i := 1$ to $m - 1$
   1. FOR $j := 1$ to 2^m step 2^i
      /* Join aller benachbarten Listen auf Level i des Baumes */
   2. Berechne $L_{j \ldots j+2^i-1} = L_{j \ldots j+2^{i-1}-1} \bowtie_{i\ell} L_{j+2^{i-1} \ldots j+2^i-1}$.
3. Berechne $L_{1 \ldots 2^m} = L_{1 \ldots 2^{m-1}} \bowtie_n L_{2^{m-1}+1 \ldots 2^m}$.

AUSGABE: Elemente von $L_{1 \ldots 2^m}$

Beispiel für $k = 2^3$:

- Join für $i = 1$: $L_{12} = L_1 \bowtie_\ell L_2$, $L_{34} = L_3 \bowtie_\ell L_4, \ldots, L_{78} = L_7 \bowtie_\ell L_8$.
- Join für $i = 2$: $L_{1234} = L_{12} \bowtie_\ell L_{34}$, $L_{5678} = L_{56} \bowtie_\ell L_{78}$.
- Join in Schritt 3: $L_{1 \ldots 8} = L_{1 \ldots 4} \bowtie_n L_{5 \ldots 8}$.

Analyse des k-Listen Algorithmus

Korrektheit:

- Alle Startlisten besitzen Länge 2^ℓ.
- D.h. durch das Join auf unterster Ebene entstehen Listen mit erwarteter Länge $\frac{2^\ell \cdot 2^\ell}{2^\ell} = 2^\ell$.
- Damit entstehen in Schritt 2 stets Listen mit erwarteter Länge 2^ℓ.
- In Schritt 3 entsteht eine Liste $L_{1\ldots k}$ mit erwarteter Länge

$$\sum_{(x_1,\ldots,x_k)} \mathrm{Ws}[x_1 + \ldots + x_k = \mathbf{0} \mid \mathrm{low}_{(m-1)\ell}(x_1 + \ldots + x_{\frac{k}{2}}) = \mathrm{low}_{(m-1)\ell}(x_{\frac{k}{2}+1} + \ldots + x_k)] = \frac{2^{2\ell}}{2^{n-(m-1)\ell}} = 1.$$

Analyse des k-Listen Algorithmus

Laufzeit und Platz:

- Die Listen $L_1, \ldots, L_k$ benötigen jeweils Platz $\tilde{\mathcal{O}}(2^\ell)$.
- In Schritt 2 berechnen wir $k-2$ Listen mit erwarteter Länge $\tilde{\mathcal{O}}(2^\ell)$.
- Damit erhalten wir Speicherplatzbedarf $\tilde{\mathcal{O}}(k2^\ell) = \tilde{\mathcal{O}}(k2^{\frac{n}{\log k+1}})$.
- Die Laufzeit für alle $k-1$ Join-Operationen beträgt $\tilde{\mathcal{O}}(2^\ell)$.
- Damit ist die Gesamtlaufzeit ebenfalls $\tilde{\mathcal{O}}(k2^\ell) = \tilde{\mathcal{O}}(k2^{\frac{n}{\log k+1}})$
- Für $k = 2^{\sqrt{n}}$ erhalten wir Zeit und Speicherplatz Komplexität

$$\tilde{\mathcal{O}}(2^{\sqrt{n}} \cdot 2^{\frac{n}{\sqrt{n}+1}}) = \tilde{\mathcal{O}}(2^{2\sqrt{n}}).$$

- Dies ist eine subexponentielle Funktion in n.

Übung: Konstruieren Sie einen Algorithmus für $k = 2^m + j, 0 < j < 2^m$ mit Komplexität $\tilde{\mathcal{O}}(k2^{\frac{n}{\log k+1}})$.

Offenes Problem:
Geht es für $k = 2^m + j$ besser? Für $k = 3$ besser als $\tilde{\mathcal{O}}(2^{\frac{n}{2}})$?