RUHR-UNIVERSITÄT BOCHUM

RUB

The Approximate k-List Problem

FSE 2017, 06.-08.03.2017

Leif Both, Alexander May
Horst Görtz Institute for IT-Security
Ruhr-University Bochum, Germany
Faculty of Mathematics

hgi
Horst Görtz Institut
für IT-Sicherheit

Outline

- Definition of the Approximate k-List Problem
- Algorithms to solve this problem
- Application to the Parity Check Problem

D. Wagner '02: Generalized Birthday Problem

Definition 1 (The k-list Problem)

Given: k lists $L_1, \ldots, L_k \subset \mathbb{F}_2^n$
Find: $(x_1, \ldots, x_k) \in L_1 \times \ldots \times L_k$:

$$x_1 + \ldots + x_k = 0.$$

- Runtime: $\tilde{\mathcal{O}}(2^{\frac{n}{\log(k)+1}})$, e.g. $\tilde{\mathcal{O}}(2^{\frac{n}{3}})$ for 4 lists

Our Problem

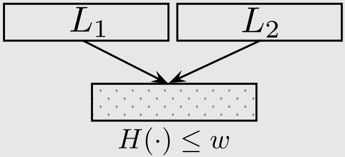
Definition 2 (The Approximate k-list Problem)

Given: k lists $L_1, \ldots, L_k \subset \mathbb{F}_2^n$, target weight $w \in [0, \frac{n}{2}]$
Find: $(x_1, \ldots, x_k) \in L_1 \times \ldots \times L_k$:

$$H(x_1 + \ldots + x_k) \leq w.$$

- Less restrictive $\Rightarrow$ more solutions $\Rightarrow$ smaller listsizes, less time/memory consumption
- Sufficient for many applications (e.g. Parity Check Problem, LPN, Decoding)

Our Approximate 2-List Algorithm

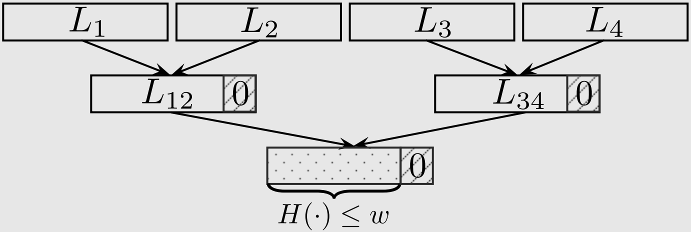


- Nearest Neighbor Search over the two lists
- Example ($n = 4, w = 1$):

$$L_1 = \{1101, 0011, 1000\}, \quad L_2 = \{0000, 0011, 0101\}$$

$$\Rightarrow H(1101 + 0101) = 1 \text{ is a solution}$$

Our Approximate 4-List Algorithm



- Example ($n = 4, w = 1$, matching on 2 bits):

$$L_1 = \{\mathbf{11\underline{01}}, 0011, 1000\}, \quad L_2 = \{0010, 1110, \mathbf{01\underline{01}}\},$$

$$L_3 = \{1100, \mathbf{01\underline{11}}, 0100\}, \quad L_4 = \{0110, 0010, \mathbf{10\underline{11}}\}$$

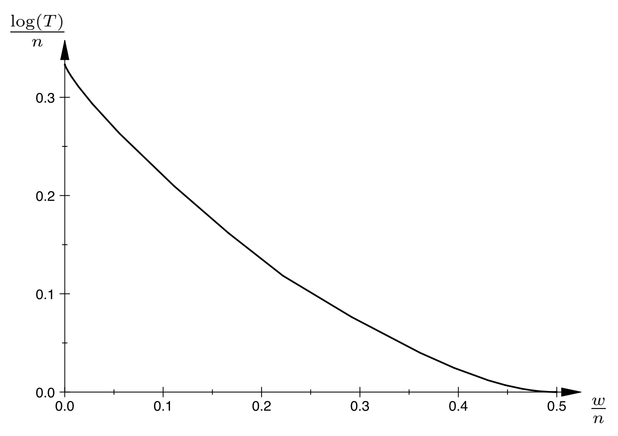
$$\Rightarrow L_{12} = \{\mathbf{\underline{10}00}\}, \quad L_{34} = \{\mathbf{\underline{11}00}\}$$

$$\Rightarrow H(1101 + 0101 + 0111 + 1011) = H(10 + 11) = 1$$

- Can be generalized easily for arbitrary powers of 2

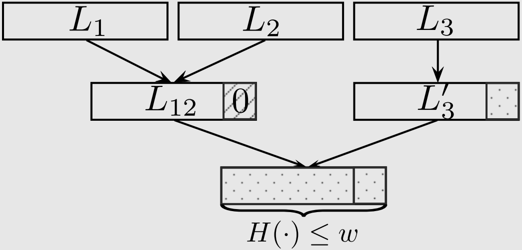
Our Approximate 4-List Algorithm

- Runtime: Maximum of listsizes and Nearest Neighbor Search Runtime



$\frac{\log T}{n}$ over $\frac{w}{n}$ for $k = 4$

Our Approximate 3-List Algorithm



- Example ($n = 4$, $w = 2$, filtering for weight 1 on 2 bits):

$$L_1 = \{\mathbf{11\underline{01}}, 0011, 1000\}, \quad L_2 = \{0010, 1110, \mathbf{01\underline{01}}\},$$

$$L_3 = \{0011, \mathbf{11\underline{10}}, 1111\}$$

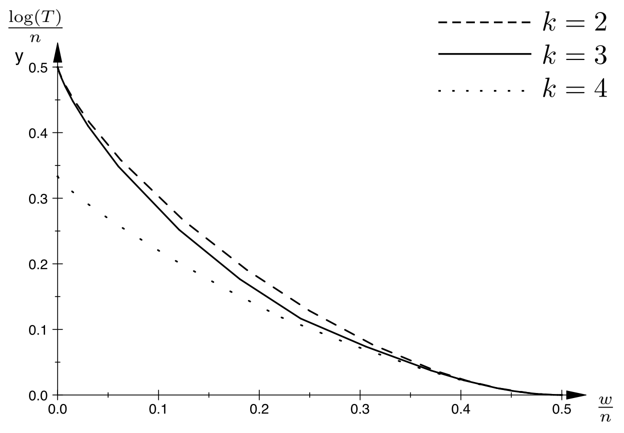
$$\Rightarrow L_{12} = \{\mathbf{\underline{10}00}\}, \quad L_3' = \{\mathbf{\underline{11}10}\}$$

$$\Rightarrow H(1101 + 0101 + 1110) = H(10 + 11) + H(00 + 10) = 2$$

- Can be generalized for $k = 6, k = 12, \ldots$

Our Approximate 3-List Algorithm

- Exponentially improved runtime for $k = 3$ in comparison to $k = 2$



$\frac{\log T}{n}$ over $\frac{w}{n}$ for different k

Application: The Parity Check Problem

Definition 3 (The Parity Check Problem)

Given: Irreducible polynomial $P(X)$ over $\mathbb{F}_2$ of degree n and upper bounds w, d.
Find: Multiple $Q(X)$ of $P(X)$ with $|Q(X)| \leq w$ and degree $\leq d$.

- Essential for Fast Correlation Attacks on Stream Ciphers

Application: The Parity Check Problem

Idea: Identify a polynomial $\mathbb{F}_2[X]/P[X]$ with its coefficient vector $\in \mathbb{F}_2^n$

- We fill k lists with polynomials of the form

$$X^a \bmod P(X) \in \mathbb{F}_2[X]/P[X],\ a \leq d$$

- Our Approximate k-list algorithm finds polynomials $X^{a_1}, \ldots, X^{a_k}$ s.t.

$$X^{a_1} + \ldots + X^{a_k} = Q'(X) \bmod P(X) \text{ with } |Q'(X)| \leq w - k$$

- $\underbrace{X^{a_1} + \ldots + X^{a_k}}_{|\cdot| \leq k} + \underbrace{Q'(X)}_{|\cdot| \leq w-k}$ solves the Parity Check Problem

Comparison To Previous Results

- Results for $k = 4$:

	Our Algorithm				Minder & Sinclair (SODA '09)		Wagner (Crypto '02)	
	min. T/M		fixed T/M					
w	T/M	deg	T/M	deg	T/M	deg	T/M	deg
≤ 4	42	40	42	40	42	40	42	40
≤ 5	41	39	43	36	43	39	42	40
≤ 6	39	37	47	32	47	37	42	40
≤ 7	38	36	49	28	49	36	42	40

Comparison of the logarithmic time/memory consumption and degree for different weights w and $n = 120$.

Summary

- Definition of the Approximate k-List Problem
- Algorithms for powers of two and in between
- Application to the Parity Check Problem

Many thanks for your attention!

Questions?