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
Outline

- Definition of the Approximate $k$-List Problem
- Algorithms to solve this problem
- Application to the Parity Check 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:** $\tilde{O}(2^{\log(k)+1})$, e.g. $\tilde{O}(2^{\frac{n}{3}})$ 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

$H(\cdot) \leq w$

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

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

$\Rightarrow H(1101 + 0101) = 1$ is a solution
Our Approximate 4-List Algorithm

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

$L_1 = \{\textbf{1101}, 0011, 1000\}, \quad L_2 = \{0010, 1110, \textbf{0101}\},$

$L_3 = \{1100, \textbf{0111}, 0100\}, \quad L_4 = \{0110, 0010, \textbf{1011}\}$

$\Rightarrow L_{12} = \{\textbf{1000}\}, \quad L_{34} = \{\textbf{1100}\}$

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

Can be generalized easily for arbitrary powers of 2
Our Approximate 4-List Algorithm

- Runtime: Maximum of listsizes and Nearest Neighbor Search Runtime

\[ \frac{\log(T)}{n} \quad \text{over} \quad \frac{w}{n} \quad \text{for} \quad k = 4 \]
Our Approximate 3-List Algorithm

\[
L_1 = \{1101, 0011, 1000\}, \quad L_2 = \{0010, 1110, 0101\},
\]

\[
L_3 = \{0011, 1110, 1111\}
\]

\[
L_{12} = \{1000\}, \quad L'_3 = \{1110\}
\]

\[
H(1101 + 0101 + 1110) = H(10 + 11) + H(00 + 10) = 2
\]

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

- Can be generalized for \(k = 6, k = 12, \ldots\)
Our Approximate 3-List Algorithm

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

![Graph showing the comparison of runtime for different $k$ values](image-url)

The Approximate $k$-List Problem | FSE 2017 | 06.-08.03.2017
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
Idea: Identify a polynomial $F_2[X]/P[X]$ with its coefficient vector $\in F_2^n$

► We fill $k$ lists with polynomials of the form

$$X^a \mod P(X) \in F_2[X]/P[X], \ a \leq d$$

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

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

► $X^{a_1} + \ldots + X^{a_k} + Q'(X)$ solves the Parity Check Problem

$$|\cdot| \leq k \quad |\cdot| \leq w - k$$
Comparison To Previous Results

Results for $k = 4$:

<table>
<thead>
<tr>
<th>$w$</th>
<th>$\leq 4$</th>
<th>$\leq 5$</th>
<th>$\leq 6$</th>
<th>$\leq 7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\leq k$</td>
<td>Our Algorithm min. T/M T/M deg</td>
<td>Minder &amp; Sinclair (SODA ’09) T/M T/M deg</td>
<td>Wagner (Crypto ’02) T/M T/M deg</td>
<td></td>
</tr>
<tr>
<td>$\leq 4$</td>
<td>42 40</td>
<td>42 40</td>
<td>42 40</td>
<td>42 40</td>
</tr>
<tr>
<td>$\leq 5$</td>
<td>41 39</td>
<td>43 36</td>
<td>43 39</td>
<td>42 40</td>
</tr>
<tr>
<td>$\leq 6$</td>
<td>39 37</td>
<td>47 32</td>
<td>47 37</td>
<td>42 40</td>
</tr>
<tr>
<td>$\leq 7$</td>
<td>38 36</td>
<td>49 28</td>
<td>49 36</td>
<td>42 40</td>
</tr>
</tbody>
</table>

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?