Computationally Data-Independent Memory Hard Functions

Mohammad Hassan Ameri
Jeremiah Blocki
Samson Zhou
MHF is a function that has high cumulative memory complexity ($cc$) when computed by any (parallel) algorithm.
Memory Hard Functions [ABMW05, Percival09]

MHF is a function that has high cumulative memory complexity ($cc$) when computed by any (parallel) algorithm.

**Amortization**: Metric that scales to the memory cost of computing function on $m$ inputs [AS15]

**Application**: Password hashing (want to ensure cost of checking millions/billions of password guesses is prohibitively high for attacker)
iMHFs

In a data-independent memory hard functions (iMHFs) $f$, the memory access pattern for the evaluation algorithm of the MHF is static (information theoretically independent of the input)

<table>
<thead>
<tr>
<th>memory access pattern for $f(x)$</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5,7</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>
iMHFs

In a data-independent memory hard functions (iMHFs) $f$, the memory access pattern for the evaluation algorithm of the MHF is static (information theoretically independent of the input).

<table>
<thead>
<tr>
<th></th>
<th>3, 3</th>
<th>2, 2</th>
<th>9, 9</th>
<th>8, 8</th>
<th>1, 1</th>
<th>5, 5, 7, 7</th>
<th>10, 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>same memory access pattern for $f(y)$</td>
<td>4, 4</td>
<td>6, 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RAM
dMHFs

In a data-dependent memory hard functions (dMHFs) $f$, the memory access pattern is dynamic and depends on the input.

<table>
<thead>
<tr>
<th>Memory Access Pattern for $f(x)$</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>5,7</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>
In a data-dependent memory hard functions (dMHFs) \( f \), the memory access pattern is dynamic and depends on the input.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>8,5</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>4,8</td>
<td>5,7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>10,10</td>
</tr>
</tbody>
</table>
iMHFs vs. dMHFs

iMHFs: Resists side channel attacks, but $cc(f) = O\left(\frac{N^2 \log \log N}{\log N}\right)$ [AB16] ($N$ is the sequential evaluation algorithm runtime)

dMHFs: Exist constructions with $cc(f) = \Omega(N^2)$ [Percival09, ACP+17], but vulnerable to side channel attacks [Bernstein05]
iMHFs vs. dMHFs

iMHFs: Resists side channel attacks, but $c_c(f) = O(\frac{N^2}{\log^2 N})$ [AB16] ($N$ is the sequential evaluation algorithm runtime).

dMHFs: Exist constructions with $c_c(f) = \Omega(N^2)$ [Percival09, ACP+17], but vulnerable to side channel attacks [Bernstein05].

Can we design functions $f$ with $c_c(f) = \Omega(N^2)$ AND resist side channel attacks?
ciMHFs

**Intuition:** Attacker cannot distinguish between memory access patterns for inputs $x$ and $y$
- Resists side channel attacks
- Can we get maximally hard ciMHF constructions?

**Naïve approach:** ORAM hides memory access patterns but incurs $\Omega(\log N)$ overhead time, so the new runtime is $\Omega(N \log N)$ and does not achieve effective $cc(f) = \Omega(N^2)$
Our Contributions (I)

We construct a family of “$k$-restricted dynamic graphs” where $k = o(N^\epsilon)$ for any constant $0 < \epsilon < 1$ with $cc(f_{G,H}) = \Omega(N^2)$

For each $G$ that is “amenable to shuffling”, there exists a computationally data-independent sequential evaluation algorithm computing an MHF based on the graph $G$ that runs in time $O(N)$

There exists a family of ciMHFs $G$ with $cc(f_{G,H}) = \Omega(N^2)$
Our Contributions (II)

Let $G$ be any family of $k$-restricted dynamic graphs with constant indegree. Then

\[
cc(f_{G,H}) = O\left(\frac{N^2}{\log \log N} + N^2 \frac{1}{2 \log \log N} \sqrt{k \frac{1}{\log \log N}}\right)
\]

Thus for $k = o\left(N^{1/\log \log N}\right)$, we have $cc(f_{G,H}) = o(N^2)$

Results essentially characterize the spectrum of $k$-restricted dynamic graphs, i.e., $k = o(N^\epsilon)$ and $k = o\left(N^{1/\log \log N}\right)$
Assumption

Evaluation algorithm has (small) data structure that attacker cannot see

E.g., tiered memory architecture, where attacker can see access patterns to RAM but not cache
Memory Hard Functions [ABMW05, Percival09]

$$\ell_1 = H(pwd), \ell_2 = H(\ell_1), \ell_3 = H(\ell_1, \ell_2),$$

$$\ell_4 = H(\ell_2), \ell_5 = H(\ell_3, \ell_4)$$
Dynamic and Static Graphs

dMHF $\ell_1 \rightarrow \ell_2 \rightarrow \ell_3 \rightarrow \ell_4 \rightarrow \ell_5$

$\ell_1 \rightarrow \ell_2 \rightarrow \ell_3 \rightarrow \ell_4 \rightarrow \ell_5$
Dynamic and Static Graphs

dMHF → $\ell_1$ → $\ell_2$ → $\ell_3$ → $\ell_4$ → $\ell_5$

$\ell_6$ → $\ell_7$ → $\ell_8$ → $\ell_9$ → $\ell_{10}$
Dynamic and Static Graphs

\[ \text{dMHF} \rightarrow \ell_1 \rightarrow \ell_2 \rightarrow \ell_3 \rightarrow \ell_4 \rightarrow \ell_5 \]

\[ \ell_6 \rightarrow \ell_7 \rightarrow \ell_3 \rightarrow \ell_4 \rightarrow \ell_5 \]
Dynamic and Static Graphs

dMHF

\( \ell_1 \rightarrow \ell_2 \rightarrow \ell_3 \rightarrow \ell_4 \rightarrow \ell_5 \)

\( \ell_6 \rightarrow \ell_7 \rightarrow \ell_8 \)

Red arrows indicate dynamic connections.
Dynamic and Static Graphs

dMHF → $\ell_1$ → $\ell_2$ → $\ell_3$ → $\ell_4$ → $\ell_5$

$\ell_6$ → $\ell_7$ → $\ell_8$ → $\ell_9$
Dynamic and Static Graphs

dMHF

\[ \ell_1 \rightarrow \ell_2 \rightarrow \ell_3 \rightarrow \ell_4 \rightarrow \ell_5 \]

\[ \ell_6 \rightarrow \ell_7 \rightarrow \ell_8 \rightarrow \ell_9 \rightarrow \ell_{10} \]
Dynamic and Static Graphs

iMHF → $\ell_1$ → $\ell_2$ → $\ell_3$ → $\ell_4$ → $\ell_5$
$k$-Restricted Dynamic Graphs

dMHF

\[ \ell_1 \rightarrow \ell_2 \rightarrow \ell_3 \rightarrow \ell_4 \rightarrow \ell_5 \]

iMHF

\[ \ell_1 \rightarrow \ell_2 \rightarrow \ell_3 \rightarrow \ell_4 \rightarrow \ell_5 \]
Maximally Hard $k$-Restricted Dynamic Graphs

Family of $k$-restricted dynamic graphs where $k = o(N^\epsilon)$ for any constant $0 < \epsilon < 1$ with $cc(f_{G,H}) = \Omega(N^2)$
Maximally Hard $k$-Restricted Dynamic Graphs

Family of $k$-restricted dynamic graphs where $k = o(N^{\epsilon})$ for any constant $0 < \epsilon < 1$ with $cc(f_{G,H}) = \Omega(N^2)$
Maximally Hard $k$-Restricted Dynamic Graphs

Family of $k$-restricted dynamic graphs where $k = o(N^\epsilon)$ for any constant $0 < \epsilon < 1$ with $cc(f_{G,H}) = \Omega(N^2)$

Block 1  Block 2  Block $N/k$
Maximally Hard $k$-Restricted Dynamic Graphs

Family of $k$-restricted dynamic graphs where $k = o(N^\epsilon)$ for any constant $0 < \epsilon < 1$ with $cc(f_{G,H}) = \Omega(N^2)$
Maximally Hard $k$-Restricted Dynamic Graphs

To compute labels $N + 1$ to $2N$, attacker should either (1) keep labels on nodes 1 to $N$ throughout or (2) recompute labels of nodes 1 to $N$ when necessary.
Maximally Hard $k$-Restricted Dynamic Graphs

To compute labels $N + 1$ to $2N$, attacker should either (1) keep labels on nodes 1 to $N$ throughout or (2) recompute labels of nodes 1 to $N$ when necessary.

Cost is $N \times N = \Omega(N^2)$
Maximally Hard $k$-Restricted Dynamic Graphs

To compute labels $N + 1$ to $2N$, attacker should either (1) keep labels on nodes 1 to $N$ throughout or (2) recompute labels of nodes 1 to $N$ when necessary.

Design the graph on nodes 1 to $N$ to be very expensive to recompute!
Maximally Hard $k$-Restricted Dynamic Graphs

Grates [Sch83]

Superconcentrators [Pip77,LT82]

Amenable to shuffling

To compute labels $N + 1$ to $2N$, attacker should either (1) keep labels on nodes 1 to $N$ throughout or (2) recompute labels of nodes 1 to $N$ when necessary.

Design the graph on nodes 1 to $N$ to be very expensive to recompute!
Maximally Hard $k$-Restricted Dynamic Graphs

Grates [Sch83]  
Superconcentrators [Pip77,LT82]

Amenable to shuffling

ciMHF with optimal $cc$

To compute labels $N + 1$ to $2N$, attacker should either (1) keep labels on nodes 1 to $N$ throughout or (2) recompute labels of nodes 1 to $N$ when necessary

Design the graph on nodes 1 to $N$ to be very expensive to recompute!
Attack on $k$-Restricted Dynamic Graphs

For $k = o\left(N^{1/\log \log N}\right)$, we have $cc(f_{G,H}) = o(N^2)$

To compute labels $N + 1$ to $2N$, attacker should either (1) keep labels on nodes 1 to $N$ throughout or (2) recompute labels of nodes 1 to $N$ when necessary

Previously: Design the graph on nodes 1 to $N$ to be very expensive to recompute labels??

**Attack**: always recompute labels because no longer possible to be very expensive for small $k$, similar strategy to [AB16]
Attack on $k$-Restricted Dynamic Graphs

For $k = o\left(N^{1/\log\log N}\right)$, we have $cc(f_{G,H}) = o(N^2)$.

To compute labels on nodes $1$ to $N$ when necessary:

1. Keep labels of a small number of key locations.

Previously: Design the graph on nodes $1$ to $N$ to be very expensive to recompute labels??

Attack: always recompute labels because no longer possible to be very expensive for small $k$, similar strategy to [AB16]
We construct a family of $k$-restricted dynamic graphs where $k = o(N^\epsilon)$ for any constant $0 < \epsilon < 1$ with $cc(f_{G,H}) = \Omega(N^2)$ and give a ciMHF implementation of $f_{G,H}$

We show that $cc(f_{G,H}) = o(N^2)$ for $k = o(N^{1/\log \log N})$
Future Directions

Fully characterize and tighten bounds for the spectrum of $k$-restricted dynamic graphs

Optimal ciMHFs without cache hierarchy assumptions

Show pebbling reduction for dMHFs