SUSTAINED MEMORY COMPLEXITY FOR MEMORY HARD FUNCTIONS

ANNIE RAICHEV AND JOSHUA GERSTEIN

ADVISOR: MIKE ROSULEK
OREGON STATE UNIVERSITY

ABSTRACT. We define a property of memory hard functions, "sustained complexity," which we believe is desirable. We attempt to find graphs which give data-indepent memory hard functions with this property, and which also are simple to implement. Ideally, memory hard functions with high sustained complexity would give attackers less of an advantage from performing many computations in parallel.

1. INTRODUCTION

Memory Hard Function (MHF) A function where the cost of computation is dominated by memory costs.
Purpose: Adversaries have access to specialized hardware that allow them to do computations in parallel and computations must faster than the honest user.
Goal is to find functions that take up more memory since memory is costly even for an attacker with specialized hardware, but still run generally fast for the honest user. These functions make it harder for adversaries who want to run dictionary attacks in order to access a password file.

We will focus on *data-independent memory hard functions* (iMHFs) whose memory access pattern is not dependent on user input. Data dependent memory hard functions, are usually easier to construct, and candidates like scrypt, have been proven to be maximally memory hard.[Al3] However, they allow for side channel attacks which make independent memory hard functions more desirable in this case.[Al2] The class of data-independent memory hard functions can also be modeled by directed acylic graphs and the pebbling game.

1.1. Properties of memory hard functions.

Definition 1.1. *The memory hardness of our function can be shown by playing the* ***pebbling game*** [Sa]*: Given a graph $G = (V,E)$, source node s, a sink node t, and a set of target nodes T with $t \in T$, a valid pebbling P is a sequence of configurations of pebbled nodes $P_i \subseteq V$ such that*

(1) *There is some $P_i \in P$ such that $T \subseteq P_i$*
(2) *$v \in P_i$ only if for all nodes $u \in V$, $(u,v) \in E$ implies $u \in P_{i-1}$, that is, pebbles are added to a node only when their predecessors have a pebble at the end of the previous step*
(3) *(sequential version only) $|P_i \setminus P_{i-1}| \leq 1$, or only one new pebble can be added at a time*

Date: 07.19.2017.

This work was done during the 2017 REU program in mathematics at Oregon State University, with support by National Science Foundation Grant DMS-1359173.

(4) *Pebbles can be removed at any point in time.*

Definition 1.2. *The **Cumulative complexity of a pebbling** P is $\Sigma_{P_i \in P}|P_i|$. This is a measure of the memory cost of computing an iMHF using the strategy given by pebbling of its graph.*

Definition 1.3. *The **Cumulative complexity of a graph** G is* $\min(\Sigma_{P_i \in P}|P_i|)$ *over all valid pebblings P of G. This is a measure of the memory cost of computing an iMHF using the best possible pebbling strategy.*

Definition 1.4. *A DAG $G = (V,E)$ is* (e,d)-reducible *if there exists a subset $S \subseteq V$, such that $|S| \leq e$ and the length of the longest directed path in the graph $G' = (V \setminus S, E)$ is less than or equal to d. If a graph is not (e,d)-reducible, we call it **(e,d)-depth robust**.*

Definition 1.5. ***Sustained Complexity(SC):** Given a pebbling Sequence P and $p \in \mathbb{N}$, $SC(P,p) = |\{P_i \in P \mid |P_i| \geq p\}|$ i.e the number of times there are at least p pebbles on the graph.*

2. Basic Structures

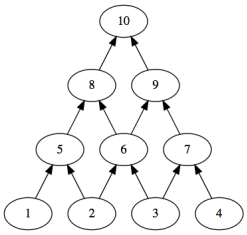


FIGURE 1. Pyramid graph with 4 source vertices

The *pyramid graph* has m source vertices with edges directed diagonally up, with each layer having one less vertex then the previous until it reaches one sink vertex. We know that in a pyramid graph with m source vertices, any pebbling strategy will use at least m pebbles.
A graph G is *covered* when all paths from source to sink contain at least one pebble. The last pebble placed prior to covering the graph happens by placing a pebble at one of the sources, and since we have m sources there must be m pebbles on the graph.

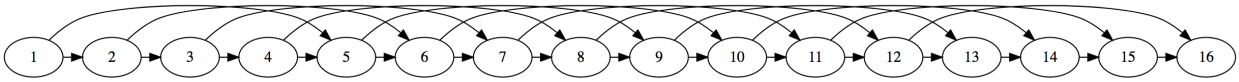


FIGURE 2. Grid graph represented in a path structure with n =16

We define a *grid graph* to be a graph $G = (V,E)$ with $V = \{1,\ldots,n\}$ and $E = \{(v, v+s) \mid v \in V \text{ and } s \in S\}$ $S = \{1, \sqrt{n}\}$

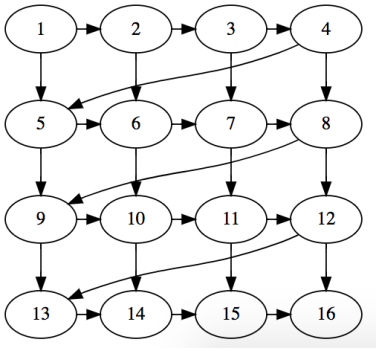


FIGURE 3. Grid graph drawn in layers

There are two natural pebbling strategies for this graph:

(1) Advance sequentially through the grid, only removing pebbles when they won't be needed later.
(2) Pebble in parallel, keeping pebbles on the negative slope diagonals, and jumping from each diagonal to the next. Use sequential pebbling where necessary.

3. CATENA DOUBLE BUTTERFLY GRAPH

We started analyzing a graph with a recursive structure, based off the Waksman network. This seemed promising because there are few "types" of vertices, and the building of the graph follows a recursive structure.

Unfortunately, this graph was identical to the *Catena Double Butterfly Graph*. There have been many memory hard functions developed for password hashing that have performed well in practice, and the Catena Double Butterfly Graph is one of them. It has a known CC of $O(n^{1.66})$[Fo].

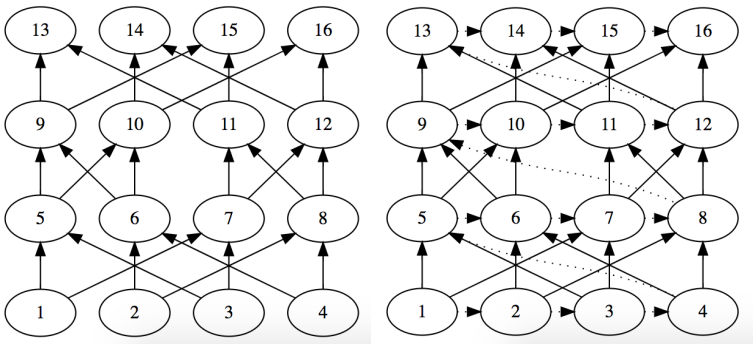


FIGURE 4. Catena DBG with 4 input vertices without / with the sequential connecting layer(dotted, winding path), which disturbs potential parallel pebbling of the graph.

3.1. **Analyzing Catena.** Our first approach to pebbling the Catena graph was through experimentation by hand. There were patterns to pebbling the graph, but it wasn't obvious just by looking at the graph what the "optimal" pebbling was. Therefore we decided to model the graphs as trees, since we could prove clear bounds on this.

Theorem 3.1. *Given a Complete binary tree of height k, with a single root being of height 0, and* 2^k *children. The minimum number of pebbles needed is* $k+1$ *in* $2^{k+1}-1$ *time steps* [Sa].

Proof: *See Lemma 10.2.1 in* [Sa] Using tree structures, gives an optimal pebbling strategy by doing a post-order traversal of the tree. And gives us clear lower bounds for pebbling, which may not be clear from trying to pebble the original graph.

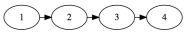


FIGURE 5. Layer 0 of the Catena DBG is just a simple path, so this could be pebbled sequentially with just one pebble.

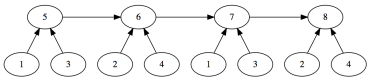


FIGURE 6. Layer 1 has binary trees of height 1, connected by a path. Therefore we need 2 pebbles for the trees, and 1 to sequentially pebble the path in between the pebbling of the trees, giving us 3 for layer 1.

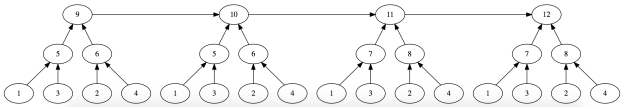


FIGURE 7. Similarly Layer 2 has binary trees of height 2, connected by a path. Therefore we need 4 pebbles for this layer.

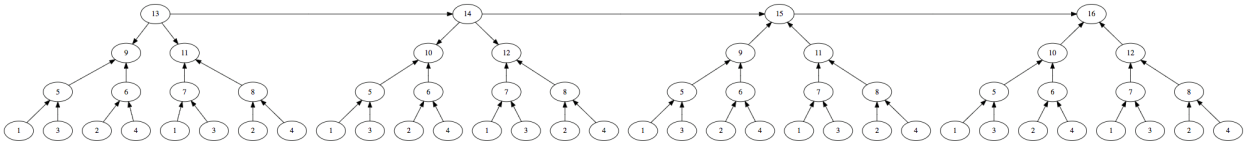


FIGURE 8. Layer 3 has binary trees of height 3, which gives us 4 pebbles for the trees, and one more for the path. Therefore any pebbling of the Catena graph with 4 input vertices, will need at least 5 pebbles.

From this we then know that for the Catena graph with 4 input vertices and the sequential connecting layer, there is no possible way to do it in less than 5 pebbles. Since the Catena Graph

has a recursive structure the number of input/output vertices (seen in Figure 3.1), of the graph increases by a factor of 2 through each recursive step. Therefore, the number of input vertices increases from 2^k to 2^{k+1}. Thus the trees corresponding to the top layer also increase by a height of 2. From this we can confirm that from every enlargement of the Catena DBG graph we get an increase of 2 pebbles needed.

3.2. **Attacking the Catena DBG.** The Catena Graph is an example of a *Layered DAG*, since all edges $(v, v+1)$ are directed in layers, and the additional edges go from a lower level to a higher level.

Theorem 3.2. *Layered DAGS are not depth robust. Given a* λ*- layered DAG G, then G is* $(n^{2/3}, n^{1/3}(\lambda+1))$*-reducible.* [Al1]

By breaking up each layer into segments of length $n^{1/3}$. We then take the last node of each one of these segments, and add it to set S. Then $S = \{i \cdot n^{1/3} \mid i \leq n^{2/3}\}$. Therefore any path can spend at most $n^{1/3}$ steps on layer i with at most $\lambda+1$ levels, giving us depth $n^{1/3}(\lambda+1)$.

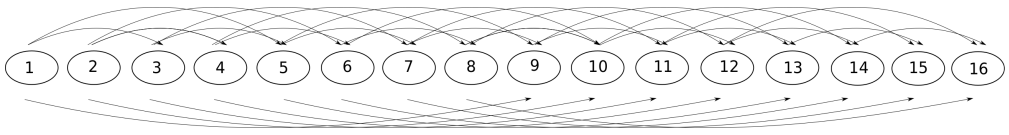
4. Graphs with Fixed Sets of Edge Lengths

Graphs that tend to have few "types" of edges or some uniform distribution of edges proved to be useful in terms of sustained memory complexity. So we looked at graphs with fixed sets of edges lengths.

Definition 4.1. *Let S be a set, and define a graph G as follows:* $G = (V,E)$*, where* $V = \{1,\ldots,N\}$ *and* $(u,v) \in E$ *if and only if* $v - u \in S$*.*

The grid graph described earlier in figure 2 falls in this category, with $S = \{1, \sqrt{|V|}\}$. These graphs are promising for sustained complexity because they have the same structure at every node, and are relatively easy to analyze. We believe that this uniformity prevents pebbling strategies that use many pebbles at only a specific point.

4.1. S **Independent of Graph Size.** We studied several families of graphs whose sets of edges S were independent of graph size. The graph with edges $S = \{2^k \mid k \in \mathbb{Z}^+\}$ seemed promising, but is not sufficiently depth robust. An attack was found involving binary Hamming weights of



the nodes.[1] We looked into similar sets of edge lengths, such as $\{2^k - 1 \mid k \in \mathbb{N}\}$. These graphs have indegree $O(\log|V|)$, but this can be dealt with by indegree reduction [Al2]. We checked the resultant graphs for depth robustness by brute force, and these graphs performed worse than those of equal size and with edges from $\{2^k \mid k \in \mathbb{Z}^+\}$. We concluded that these graphs were not worth pursuing further.

[1] David Cash, Mike Rosulek

4.2. *S* **Dependent on Graph Size.**

Definition 4.2. *The **geometric graphs** are the family of graphs of the form* $G = (V,E)$, $V = \{1,\ldots,b^d\}$, $E = \{(v,v+s) \mid v \in V, s \in S\}$, *with* $S = \{1,|V|^{\frac{1}{d}},|V|^{\frac{2}{d}},\ldots,|V|^{\frac{d-1}{d}}\}$, *and* $b,d \in \mathbb{N}$.

Note: geometric graphs have maximum indegree d, which is constant. Also, when $d = 2$, this definition describes the grid graph, from figure 2. These graphs are not vulnerable to the same hamming weight attack that worked on graphs with edge lengths powers of two, even though these edge lengths also form a geometric series.

4.3. **1/3 2/3 Graph.** After deciding the aforementioned family of graphs could prove as good candidates for sustained memory complexity we began to look at specific cases.

Definition 4.3. $G = (E,V)$ *with n vertices and edges* $\{(v,v+s) \mid v \in V$ *and* $s \in S\}$ *s.t* $S = \{1,n^{1/3},n^{2/3}\}$.

Note this is an example of the generalization above when d=3.

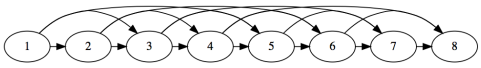


FIGURE 9. $1/3$ $2/3$ Graph with n = 8 vertices.

This graph seemed promising because it has a uniform structure like the grid graph, and also is resistant to hamming weight attacks.
The general structure of this graph, follows an overlapping pyramid structure, which alludes to having good sustained memory complexity like the grid.

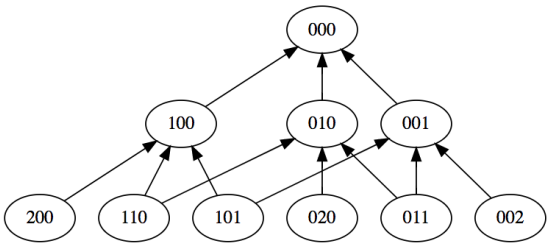


FIGURE 10. Generalized structure of $1/3$ $2/3$ showing overlapping pyramid structure.

We can again look at this graph as a tree, but in this case it is a ternary tree instead of a binary tree, which means for height k, with a single vertex being of height 0, and 3^k children; the minimum number of pebbles needed is $2k+1$. So for the previous example, we get the tree below, which means we need a minimum of 5 pebbles for this graph.

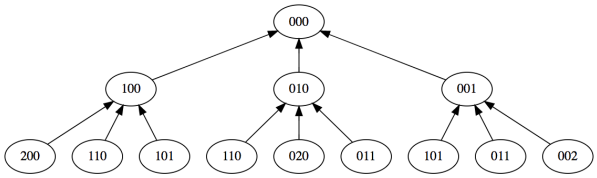


FIGURE 11. Tree structure of 1/3 2/3 graph.

5. CONCLUSION

While we did not find any iMHFs with both good sustained complexity and cumulative complexity comparable to current widespread password hash MHFs, we the furthered understanding of properties of graphs with good sustained complexity. We found families of graphs with many interesting properties. We believe that the geometric graphs in particular warrant further study.

REFERENCES

[Fo] Forler, Christian; Lucks, Stefan and Wenzel, Jacob. *Catena: A Memory-Consuming Password-Scrambling Framework*

[Al1] Alwen, Joël; Blocki, Jeremiah. *Efficiently Computing Data-Independent Memory-Hard Functions*

[Al2] Alwen, Joël; Blocki, Jeremiah; Pietrzak, Krzysztof. *Depth-Robust Graphs and Their Cumulative Memory Complexity*

[Sa] Savage, John. *Models of Computation: Exploring the Power of Computing*

[Al3] Alwen, Joël; Chen, Binyi; Pietrzak, Krzysztof; Reyzin, Leonid; Tessaro, Stefano. *Scrypt is Maximally Memory-Hard*

UNIVERSITY OF CALIFORNIA, SAN DIEGO
E-mail address: araichev@ucsd.edu

CARLETON COLLEGE
E-mail address: gersteinj@carleton.edu