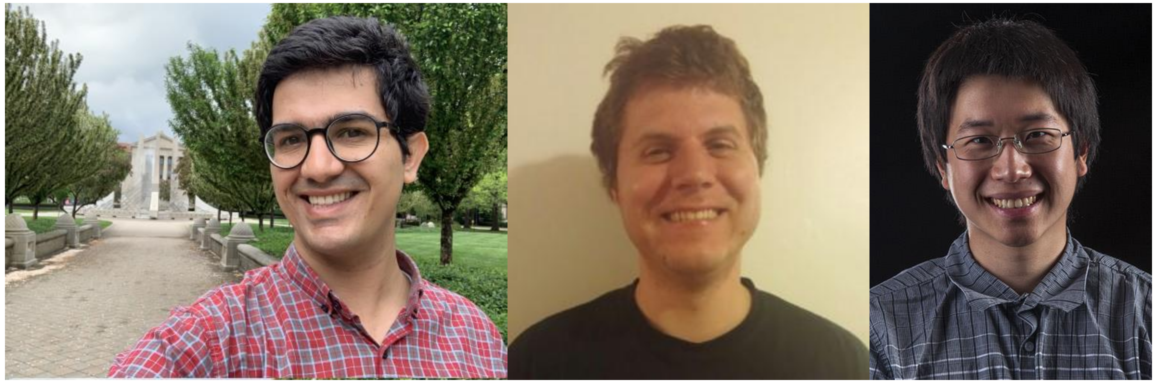
Computationally Data-Independent Memory Hard Functions

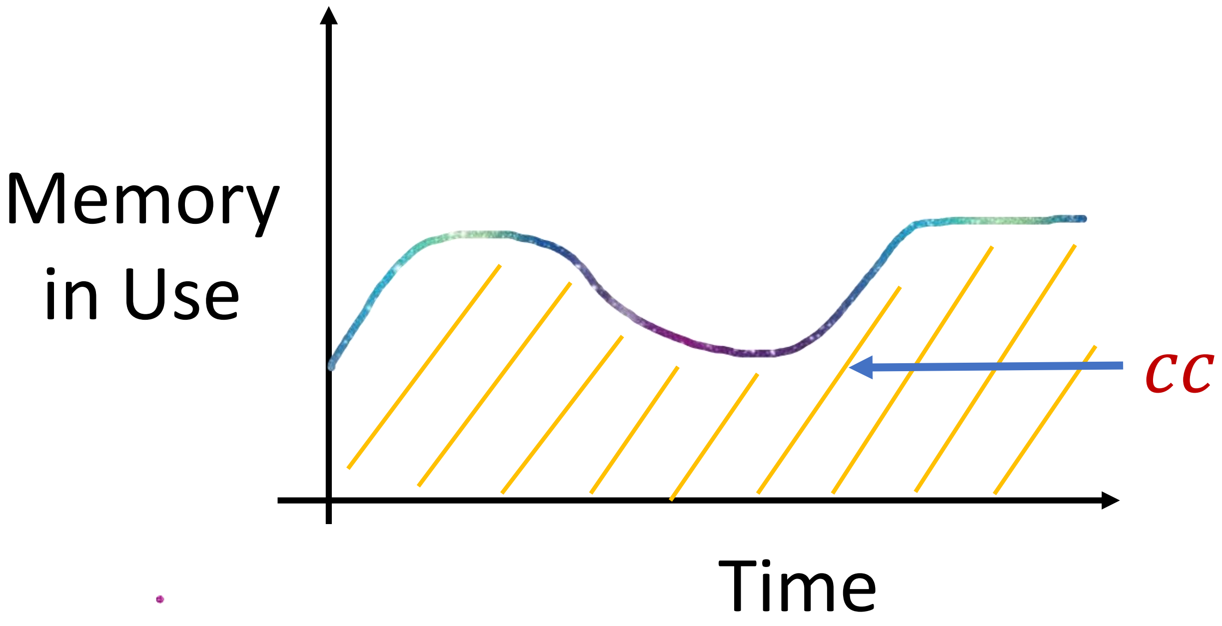


Mohammad Hassan Ameri
Jeremiah Blocki
Samson Zhou

Innovations in Theoretical Computer Science

Memory Hard Functions [ABMW05, Percival09]

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)

memory access pattern for $f(x)$

3		
	2	
9		8
	1	
4	5,7	
6		10

RAM

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)

same memory access pattern for $f(y)$

3, 3		
	2, 2	
9, 9		8, 8
	1, 1	
4, 4	5, 5, 7, 7	
6, 6		10, 10

RAM

dMHFs

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

memory access pattern for $f(x)$

3		
	2	
9		8
	1	
4	5,7	
6		10

RAM

dMHFs

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

different memory access pattern for $f(y)$

3	4	6
2	2	1
9		8, 5
9	1	7
4, 8	5,7	
6	3	10, 10

RAM

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)

optimal

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

iMHFs vs. dMHFs

iMHFs: P

$O\left(\frac{N^2 \log}{}\right.$

algo

dMHFs: E $^2)$

[Percival09, AC to side channel

attacks [Bernstein0

Can we design functions f with $cc(f) = \Omega(N^2)$ 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

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

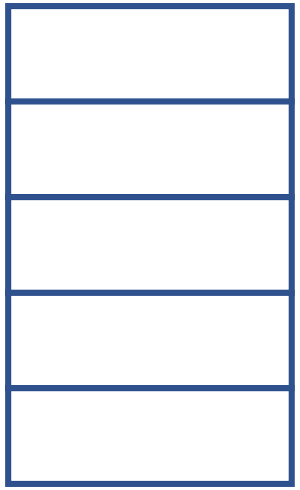
Thus for $k = o\left(N^{1/\log\log N}\right)$, we have $cc(f_{G,H}) = o(N^2)$

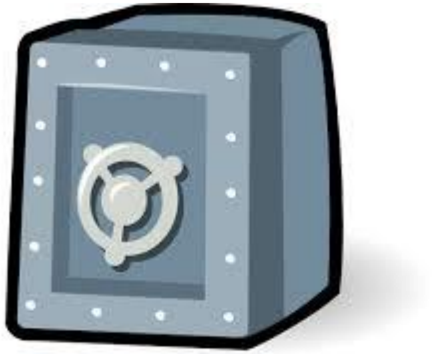
Results essentially characterize the spectrum of k-restricted dynamic graphs, i.e., $k = o(N^{\epsilon})$ and $k = o\left(N^{1/\log\log N}\right)$

Assumption

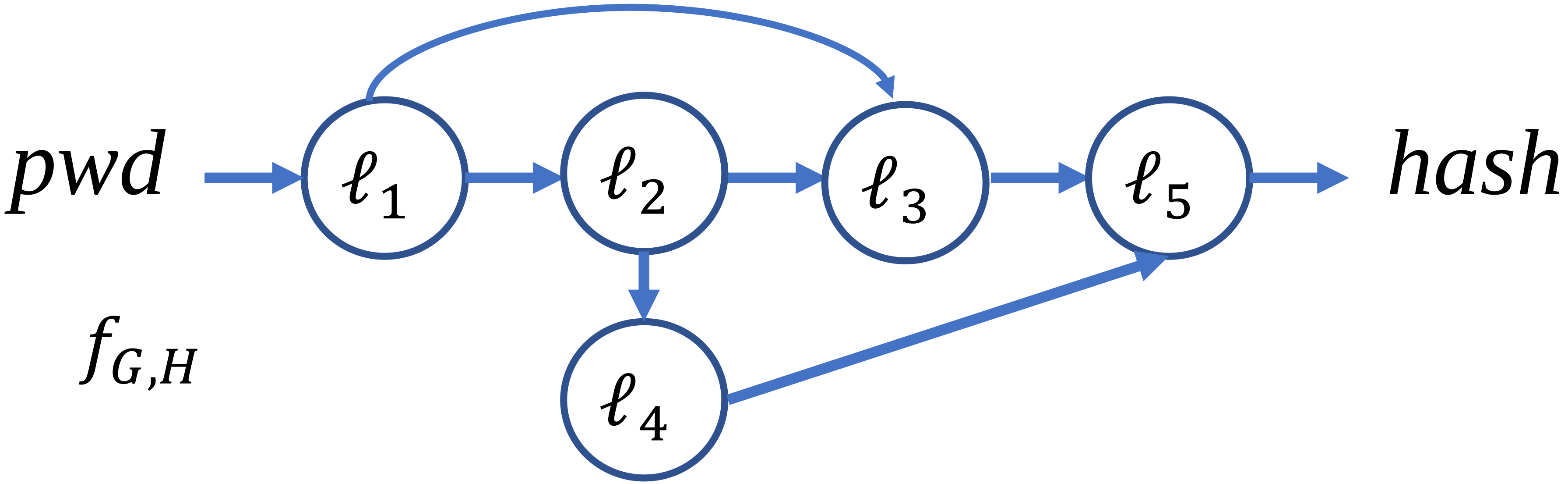
Evaluation algorithm has (small) data structure that attacker cannot see

E.g., tiered memory architecture, where attacker can see access patterns to RAM but not cache





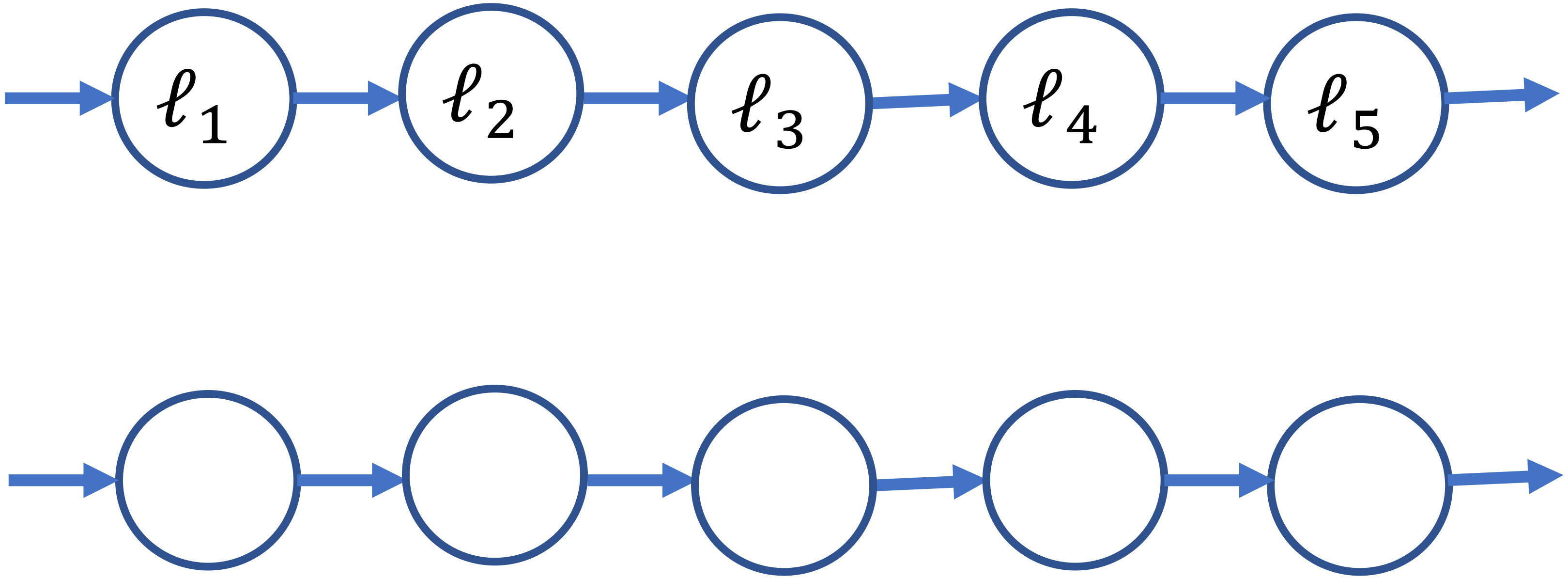
Memory Hard Functions [ABMW05, Percival09]



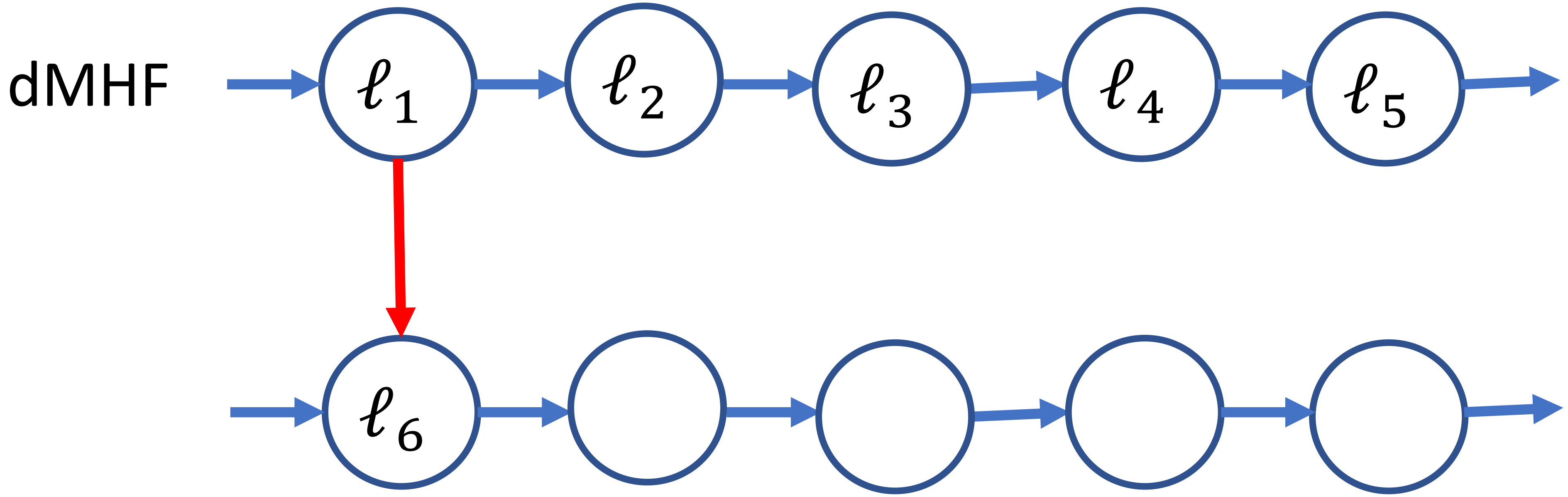
$$\ell_1 = H(pwd), \ell_2 = H(\ell_1), \ell_3 = H(\ell_1, \ell_2),$$
$$\ell_4 = H(\ell_2), \ell_5 = H(\ell_3, \ell_4)$$

Dynamic and Static Graphs

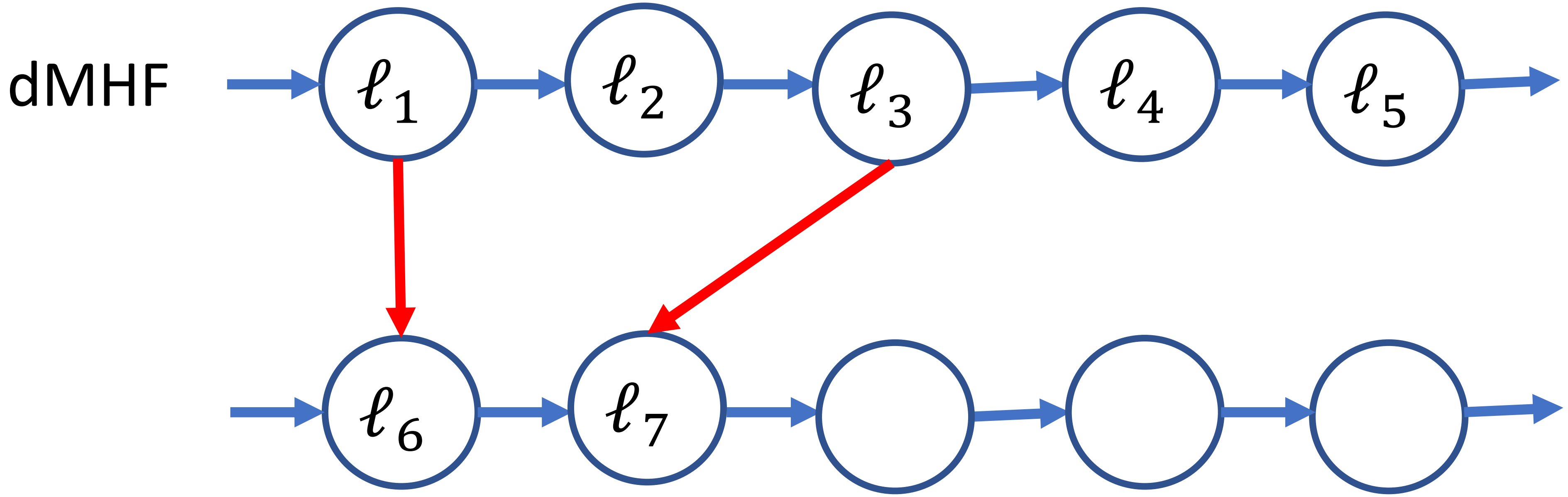
dMHF



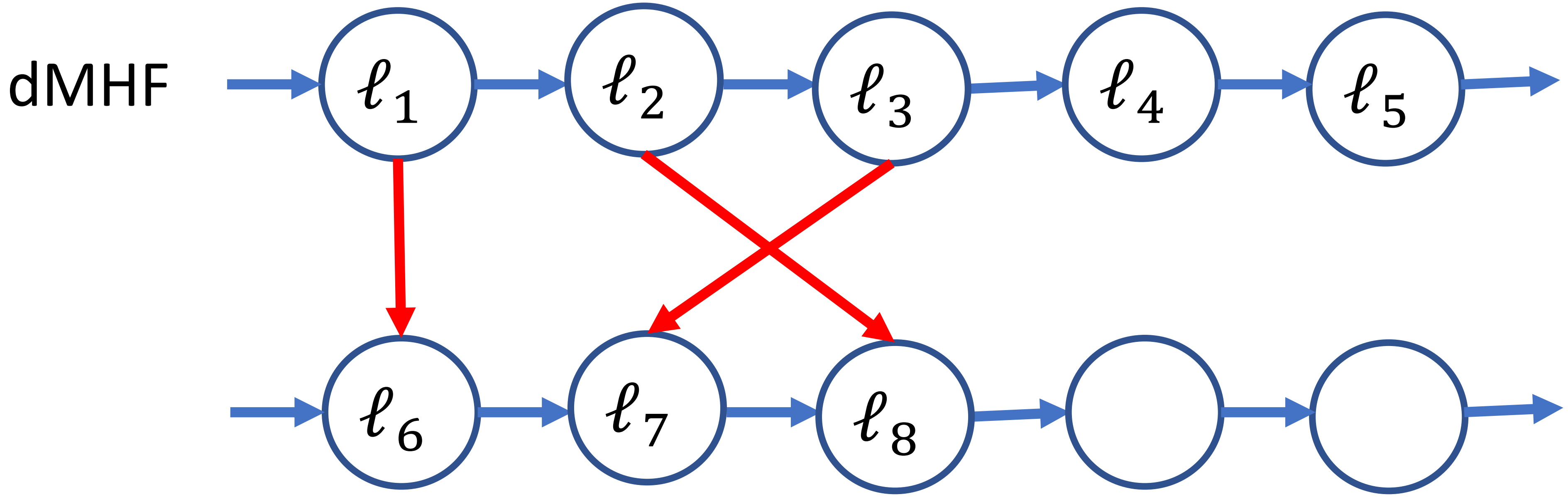
Dynamic and Static Graphs



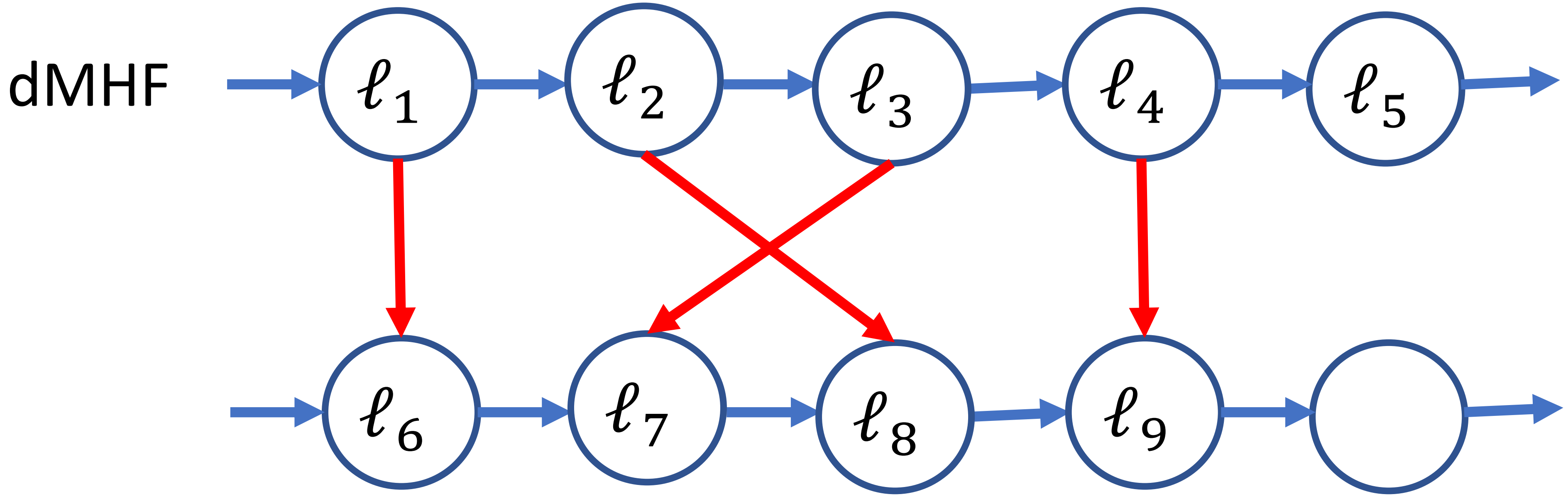
Dynamic and Static Graphs



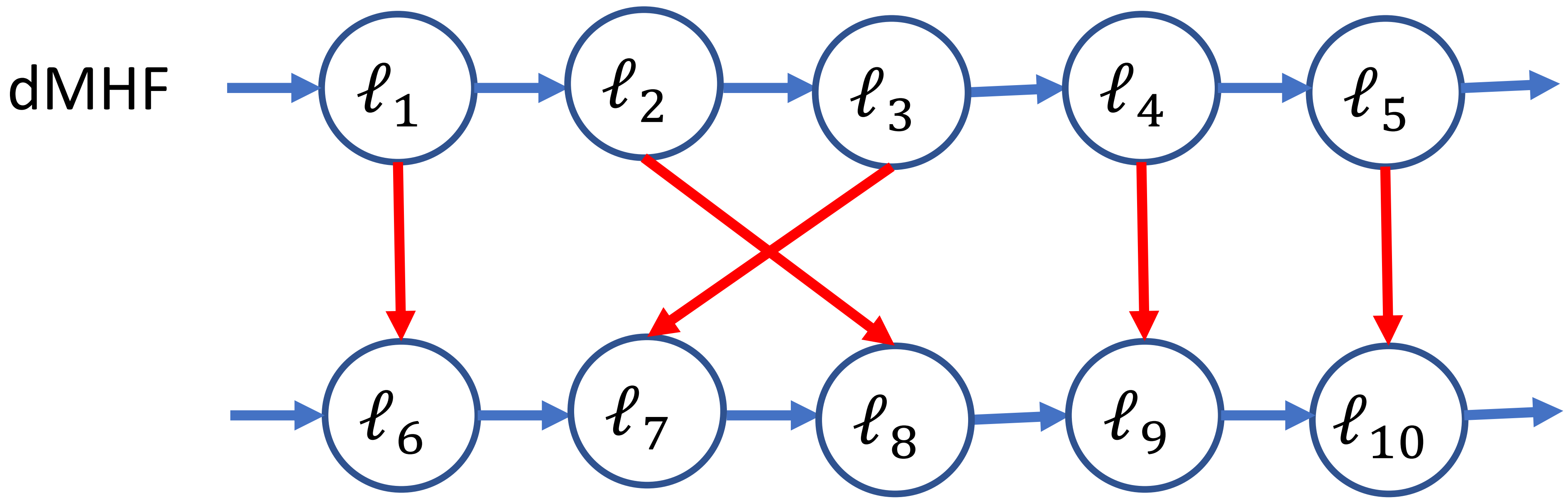
Dynamic and Static Graphs



Dynamic and Static Graphs

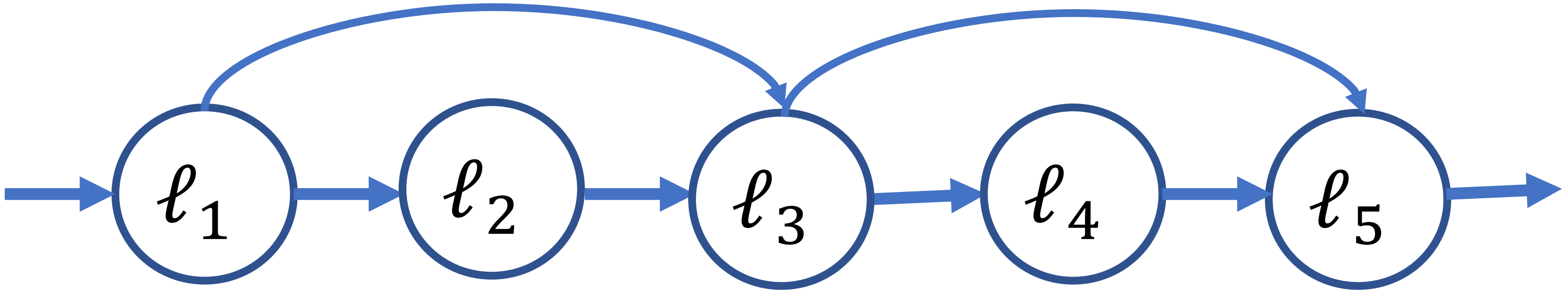


Dynamic and Static Graphs

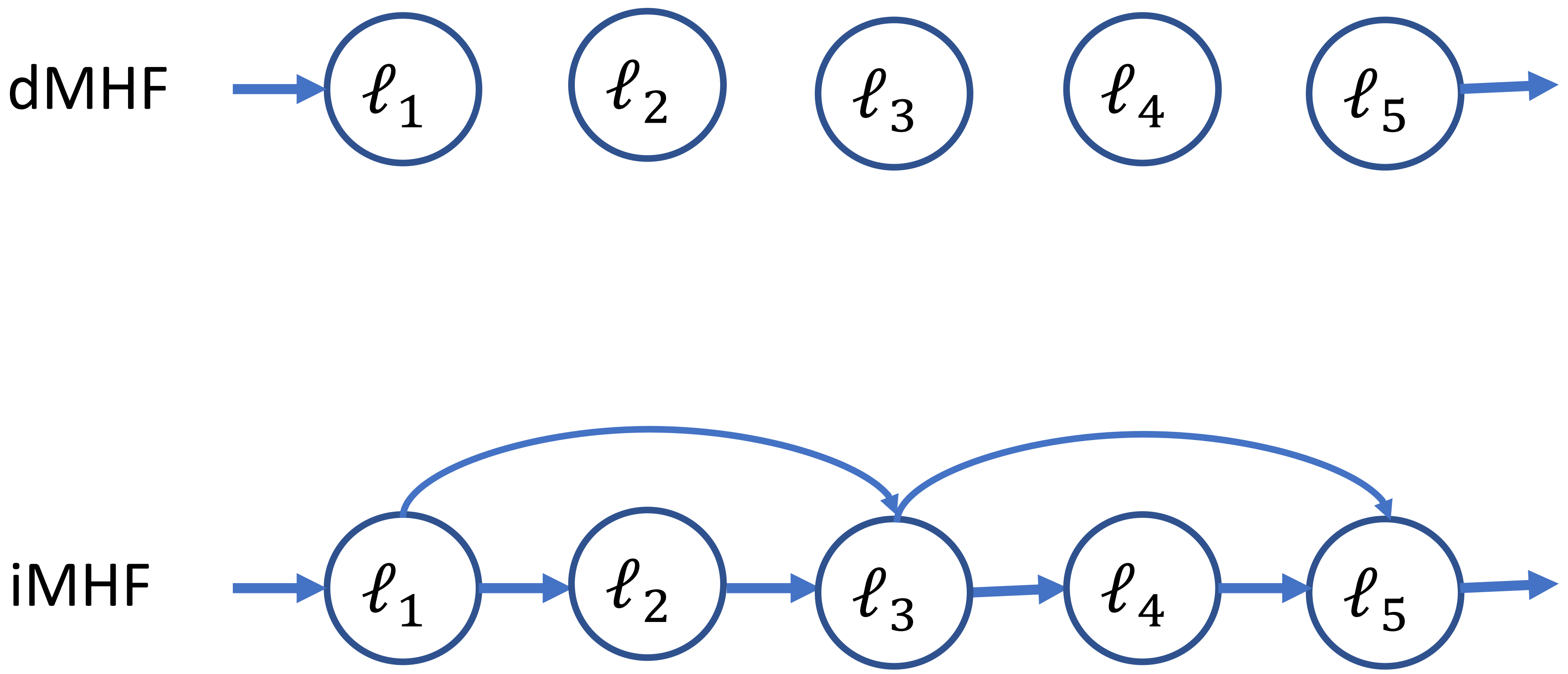


Dynamic and Static Graphs

iMHF

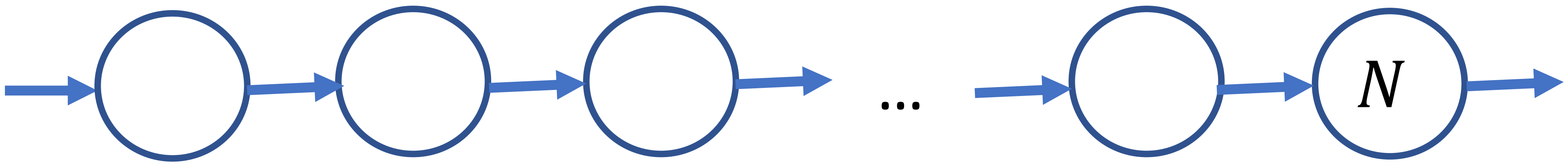


k-Restricted Dynamic Graphs



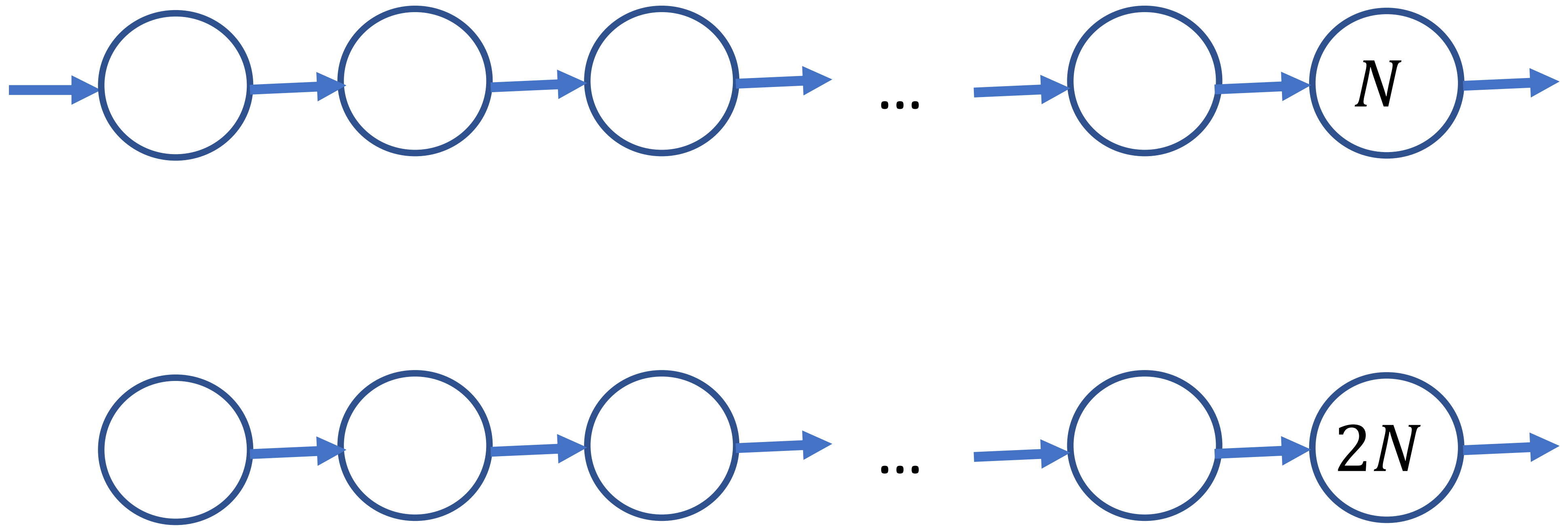
Maximally Hard k-Restricted Dynamic Graphs

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



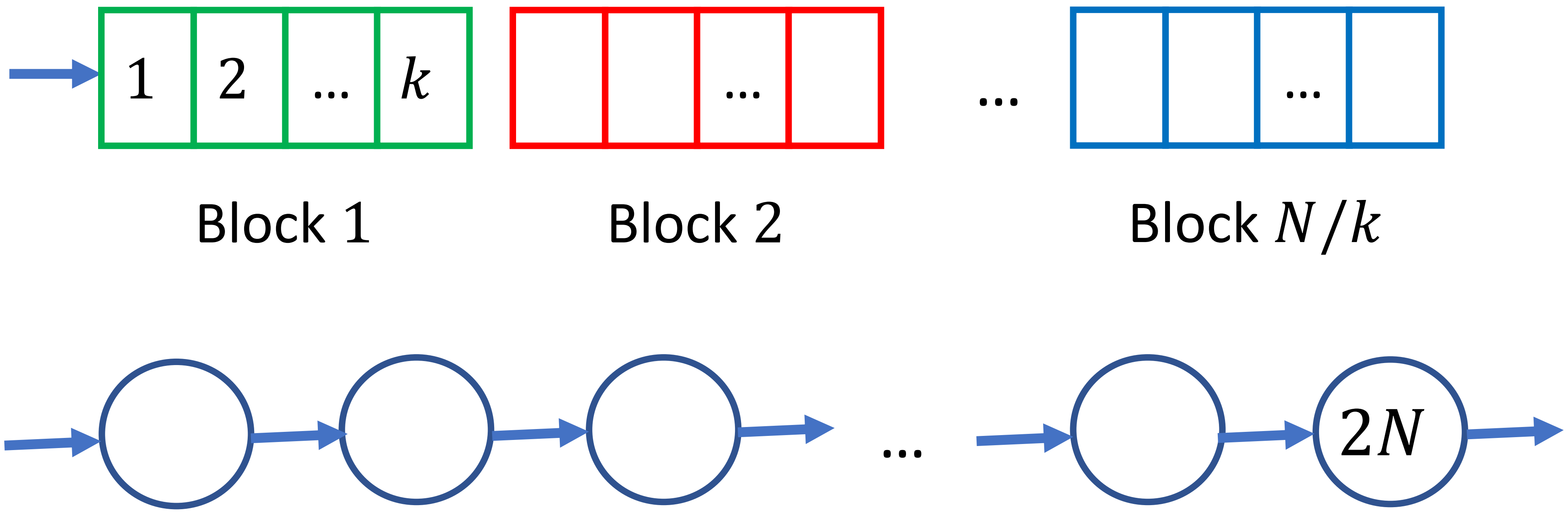
Maximally Hard k-Restricted Dynamic Graphs

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



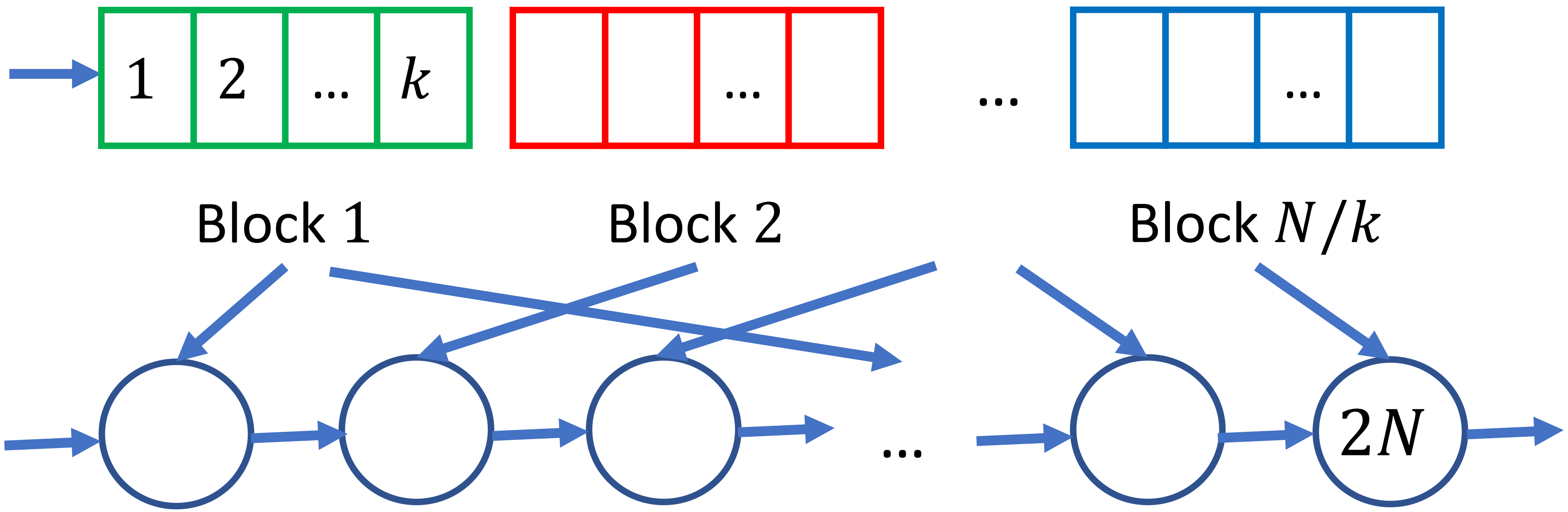
Maximally Hard k-Restricted Dynamic Graphs

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

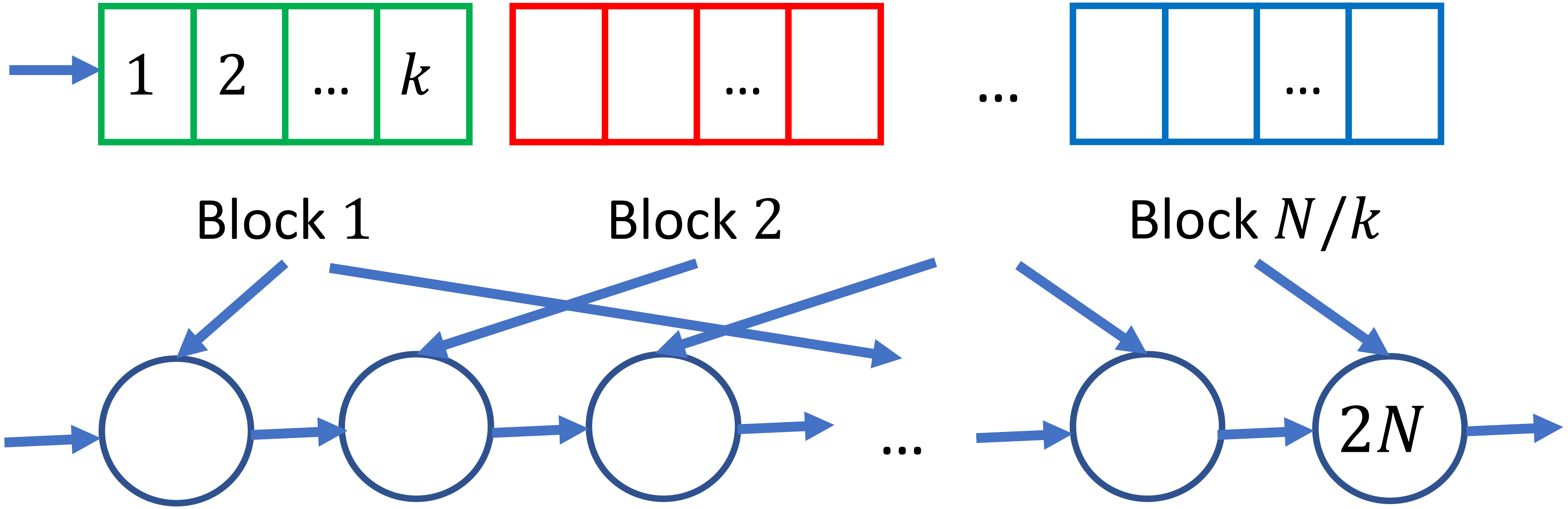


Maximally Hard k-Restricted Dynamic Graphs

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

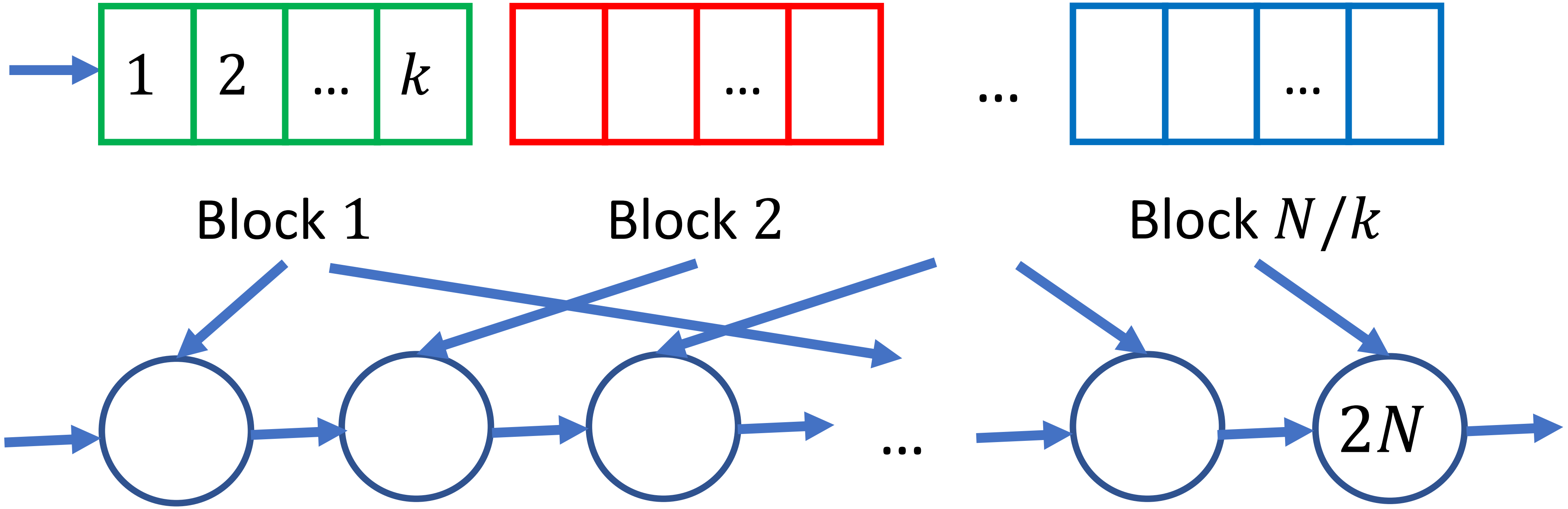


Maximally Hard k-Restricted Dynamic Graphs



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

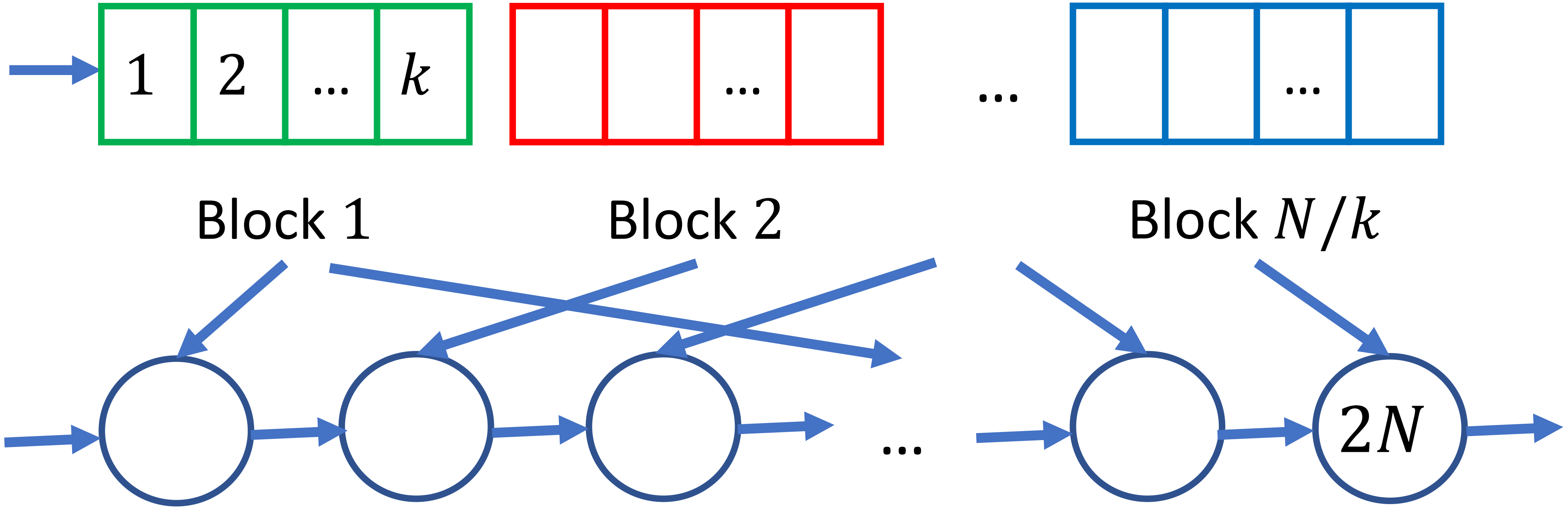
Maximally Hard k-Restricted Dynamic Graphs



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 $N \times N = \Omega(N^2)$

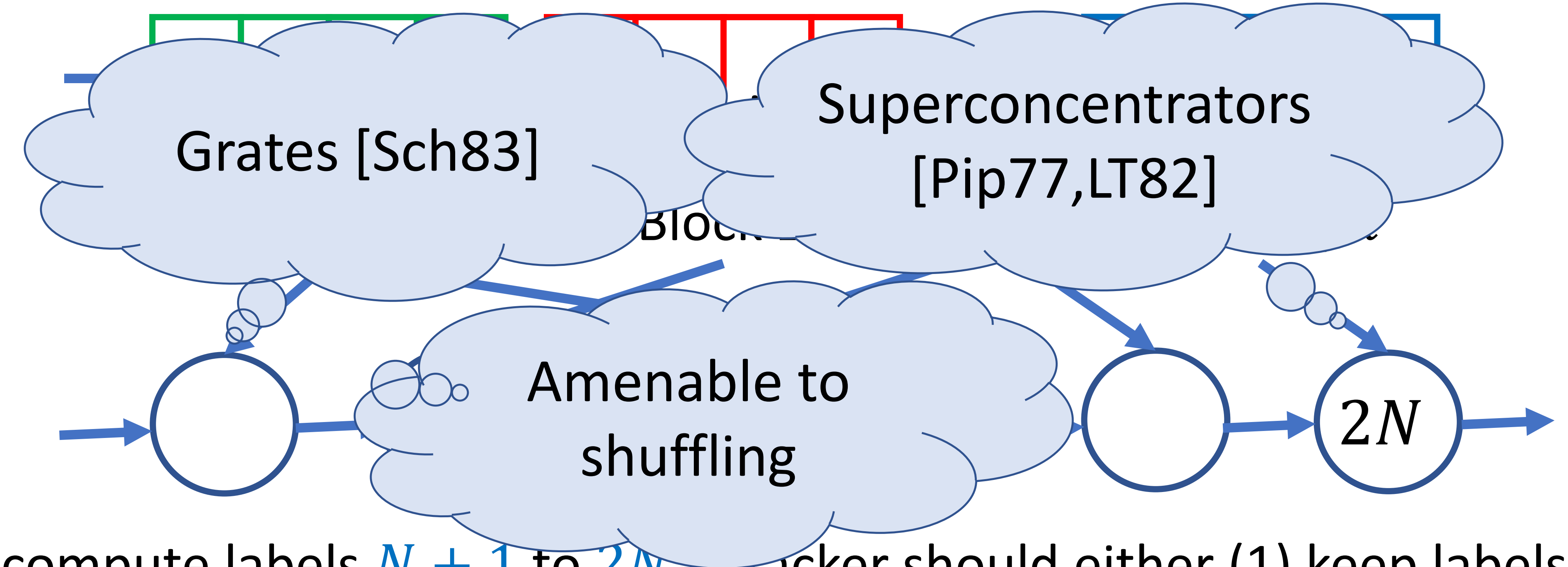
Maximally Hard k-Restricted Dynamic Graphs



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!

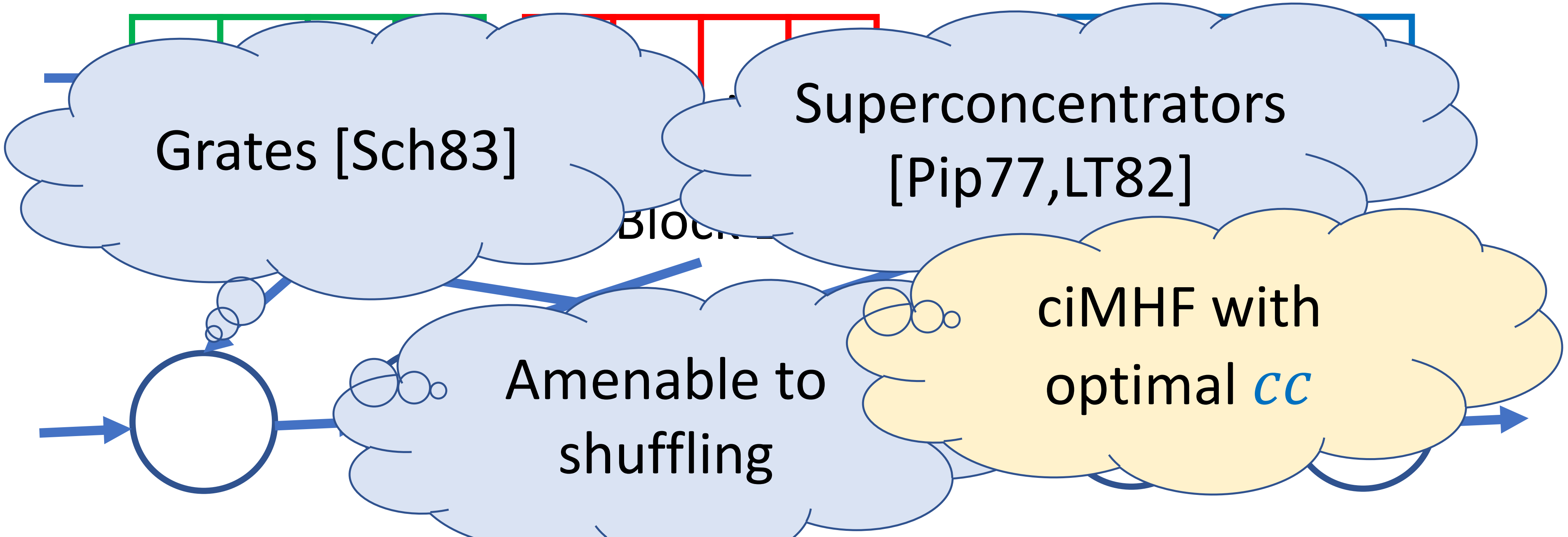
Maximally Hard k-Restricted Dynamic Graphs



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



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 $cc(f_{G,H}) = o(N^2)$

Valiant's Lemma [Val77]:
Keep labels of a small
number of key locations

To comp … (1) keep labels
on nod … nodes 1 to N
when ne …

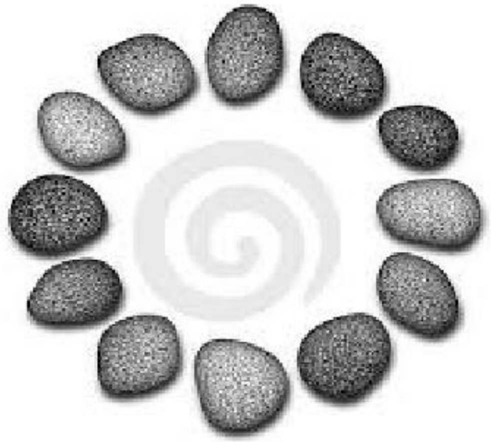
Previously: Desi … 1 to N to be very expensive to
recompute labels??

Attack: always recompute labels because no longer possible to be very expensive for small k, similar strategy to [AB16]

Summary

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 $k = o\left(N^{1/\log\log N}\right)$



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