Time space tradeoffs for attacks against one-way functions and PRGs

Anindya De

University of California, Berkeley

Joint work with
Luca Trevisan - UC Berkeley and Stanford University
Madhur Tulsiani - Princeton University

What is this talk about?

- Can “brute-force” attacks on cryptographic primitives be improved upon?

What is this talk about?

- Can "brute-force" attacks on cryptographic primitives be improved upon?
  - Recover a key of length k in time less than 2^k.

What is this talk about?

- Can "brute-force" attacks on cryptographic primitives be improved upon?
  - Recover a key of length k in time less than 2^k.
  - In time t, recover key with probability better than $t/2^k$.

What is this talk about?

- Can “brute-force” attacks on cryptographic primitives be improved upon?
  - Recover a key of length k in time less than 2^k.
  - In time t, recover key with probability better than $t/2^k$.
- Brute force : optimal when restricted to uniform algorithms

What is this talk about?

- Can “brute-force” attacks on cryptographic primitives be improved upon?
  - Recover a key of length k in time less than 2^k.
  - In time t, recover key with probability better than $t/2^k$.

- Brute force : optimal when restricted to uniform algorithms

- Are better (non-uniform) attacks possible against:
  - one-way functions?
  - pseudo-random generators?

Definitions of primitives

- $N = 2^n$, $[N] \cong \{0,1\}^n$.

Definitions of primitives

- $N = 2^n, \ \ [N] \cong \{0,1\}^n$.
- One-way function: $f : [N] \to [N]$ is (t, ϵ)-one way if for every algorithm A of complexity $\leq t$

$$\Pr_{x \sim \{0,1\}^n}\left[A^f(f(x)) = x' \ \mid \ f(x') = f(x)\right] \leq \epsilon$$

Definitions of primitives

- $N = 2^n, \;\; [N] \cong \{0,1\}^n$.
- One-way function: $f : [N] \to [N]$ is (t, ϵ)-one way if for every algorithm A of complexity $\leq t$

$$\Pr_{x \sim \{0,1\}^n} \left[A^f(f(x)) = x' \;\mid\; f(x') = f(x) \right] \leq \epsilon$$

- PRG: $G : [N] \to [2N]$ is a (t, ϵ)-secure PRG if for every algorithm A of complexity $\leq t$

$$\left| \Pr_{x \sim [N]} [A^G(G(x)) = 1] - \Pr_{y \sim [2N]} [A^G(y) = 1] \right| \leq \epsilon$$

Measure of Complexity

- complexity $\neq$ time, as A may compute f^{-1} in $O(\log N)$ time by storing all inverses.

Measure of Complexity

- complexity $\neq$ time, as A may compute f^{-1} in $O(\log N)$ time by storing all inverses.
- complexity $=$ pre-computed advice $+$ running time.

Measure of Complexity

- complexity $\neq$ time, as A may compute f^{-1} in $O(\log N)$ time by storing all inverses.
- complexity $=$ pre-computed advice $+$ running time.
- Can be implemented on a RAM machine with time and space t.
- Similar to circuit complexity.

Upper bounds

	Primitive	Complexity
[Hellman 80]	Permutation f	$\tilde{O}(\sqrt{N})$

Upper bounds

	Primitive	Complexity
[Hellman 80]	Permutation f	$\tilde{O}(\sqrt{N})$
[Hellman 80]	Random function f (heuristic)	$\tilde{O}(N^{2/3})$

Upper bounds

	Primitive	Complexity
[Hellman 80]	Permutation f	$\tilde{O}(\sqrt{N})$
[Hellman 80]	Random function f (heuristic)	$\tilde{O}(N^{2/3})$
[Fiat-Naor 99]	Any f, all inputs	$\tilde{O}(N^{3/4})$

Upper bounds

	Primitive	Complexity	
[Hellman 80]	Permutation f	$\tilde{O}(\sqrt{N})$	
[Hellman 80]	Random function f (heuristic)	$\tilde{O}(N^{2/3})$	
[Fiat-Naor 99]	Any f, all inputs	$\tilde{O}(N^{3/4})$	
[DTT 10]	Any f, ϵ-fraction of inputs	$\tilde{O}(\sqrt{\epsilon N})$	$\epsilon \leq N^{-1/3}$
		$\tilde{O}(\epsilon^{5/4} N^{3/4})$	$\epsilon \geq N^{-1/3}$

Upper bounds

	Primitive	Complexity	
[Hellman 80]	Permutation f	$\tilde{O}(\sqrt{N})$	
[Hellman 80]	Random function f (heuristic)	$\tilde{O}(N^{2/3})$	
[Fiat-Naor 99]	Any f, all inputs	$\tilde{O}(N^{3/4})$	
[DTT 10]	Any f, ϵ-fraction of inputs	$\tilde{O}(\sqrt{\epsilon N})$	$\epsilon \leq N^{-1/3}$
		$\tilde{O}(\epsilon^{5/4} N^{3/4})$	$\epsilon \geq N^{-1/3}$
[ACR 97]	PRG $G(x) \stackrel{\text{def}}{=} (f(x), P(x))$	$\tilde{O}(\epsilon^2 N)$	

Upper bounds

	Primitive	Complexity	
[Hellman 80]	Permutation f	$\tilde{O}(\sqrt{N})$	
[Hellman 80]	Random function f (heuristic)	$\tilde{O}(N^{2/3})$	
[Fiat-Naor 99]	Any f, all inputs	$\tilde{O}(N^{3/4})$	
[DTT 10]	Any f, ϵ-fraction of inputs	$\tilde{O}(\sqrt{\epsilon N})$	$\epsilon \leq N^{-1/3}$
		$\tilde{O}(\epsilon^{5/4} N^{3/4})$	$\epsilon \geq N^{-1/3}$
[ACR 97]	PRG $G(x) \stackrel{\text{def}}{=} (f(x), P(x))$	$\tilde{O}(\epsilon^2 N)$	
[DTT 10]	Any PRG	$\tilde{O}(\epsilon^2 N)$	

Upper bounds

	Primitive	Complexity	
[Hellman 80]	Permutation f	$\tilde{O}(\sqrt{N})$	
[Hellman 80]	Random function f (heuristic)	$\tilde{O}(N^{2/3})$	
[Fiat-Naor 99]	Any f, all inputs	$\tilde{O}(N^{3/4})$	
[DTT 10]	Any f, ϵ-fraction of inputs	$\tilde{O}(\sqrt{\epsilon N})$	$\epsilon \leq N^{-1/3}$
		$\tilde{O}(\epsilon^{5/4} N^{3/4})$	$\epsilon \geq N^{-1/3}$
[ACR 97]	PRG $G(x) \stackrel{\text{def}}{=} (f(x), P(x))$	$\tilde{O}(\epsilon^2 N)$	
[DTT 10]	Any PRG	$\tilde{O}(\epsilon^2 N)$	

All above results are actually stated as time-space tradeoffs. Complexity is optimized when $T = S$.

Lower bounds

Better stated in terms of a tradeoff between T and S.

Lower bounds

Better stated in terms of a tradeoff between T and S.

	Primitive	Tradeoff
[Yao 90] [Gennaro-Trevisan 00] [Wee 05]	Permutation f, ϵ-fraction of inputs	$T \cdot S = \tilde{\Omega}(\epsilon N)$ for $T = O(\sqrt{\epsilon N})$

Lower bounds

Better stated in terms of a tradeoff between T and S.

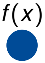
	Primitive	Tradeoff
[Yao 90] [Gennaro-Trevisan 00] [Wee 05]	Permutation f, ϵ-fraction of inputs	$T \cdot S = \tilde{\Omega}(\epsilon N)$ for $T = O(\sqrt{\epsilon N})$
[DTT 10]	Permutation f, ϵ-fraction of inputs	$T \cdot S = \tilde{\Omega}(\epsilon N)$ for any T

Lower bounds

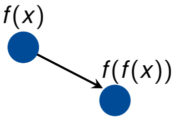
Better stated in terms of a tradeoff between T and S.

	Primitive	Tradeoff
[Yao 90] [Gennaro-Trevisan 00] [Wee 05]	Permutation f, ϵ-fraction of inputs	$T \cdot S = \tilde{\Omega}(\epsilon N)$ for $T = O(\sqrt{\epsilon N})$
[DTT 10]	Permutation f, ϵ-fraction of inputs	$T \cdot S = \tilde{\Omega}(\epsilon N)$ for any T
[DTT 10]	PRG $G \stackrel{\text{def}}{=} (f(x), P(x))$	$T \cdot S = \Omega(\epsilon^2 N)$

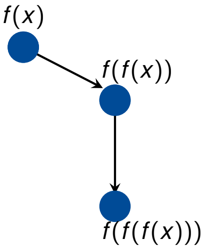
Hellman's approach for permutations



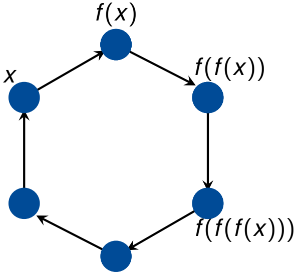
Hellman's approach for permutations



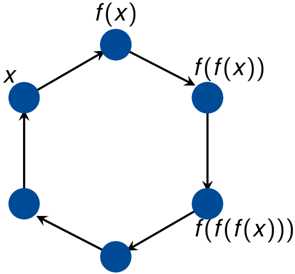
Hellman's approach for permutations



Hellman's approach for permutations

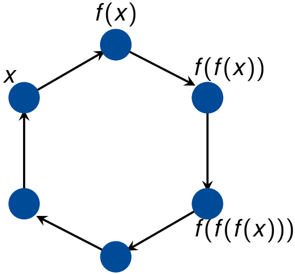


Hellman's approach for permutations



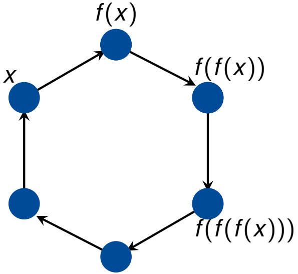
In small cycles of size less than $\sqrt{N}$, compute $f(x)$, $f(f(x))$, ...

Hellman's approach for permutations



In small cycles of size less than $\sqrt{N}$, compute $f(x)$, $f(f(x))$, ...
At some point, you hit x. $f^{-1}(x)$ is the penultimate point in the sequence.

Hellman's approach for permutations

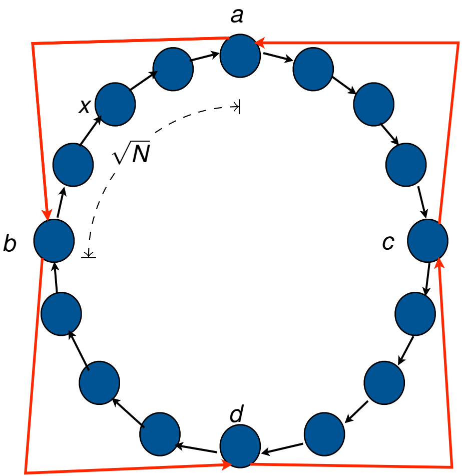


In small cycles of size less than $\sqrt{N}$, compute $f(x)$, $f(f(x))$, ...

At some point, you hit x. $f^{-1}(x)$ is the penultimate point in the sequence.

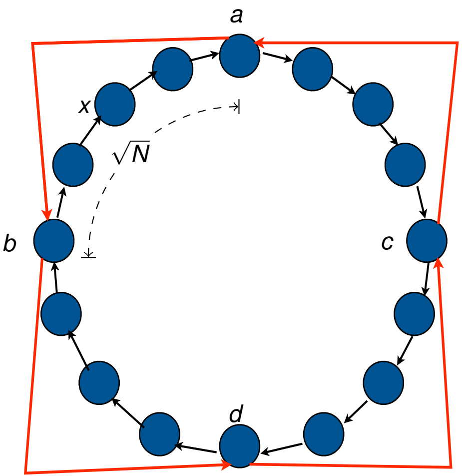
Time complexity of computation is $\tilde{O}(\sqrt{N})$.

What happens to large cycles?



In large cycles, store back-links at a distance of $\sqrt{N}$

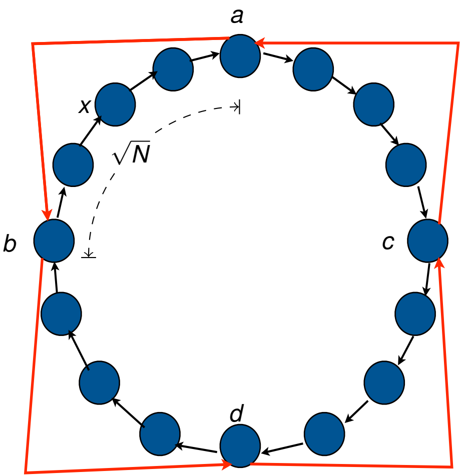
What happens to large cycles?



In large cycles, store back-links at a distance of $\sqrt{N}$

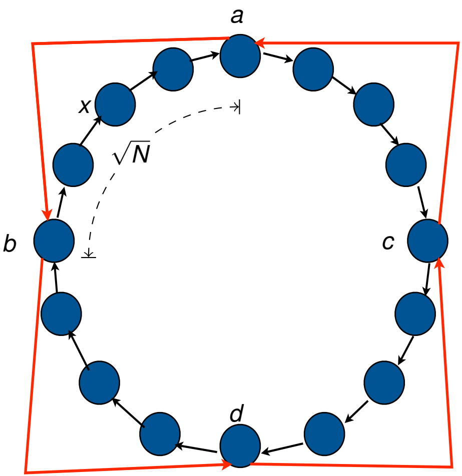
For e.g., store (a, b), (b, c), (c, d) and (d, a) in a data-structure

What happens to large cycles?



Compute $f(x)$, $f(f(x))$, ... till you hit a point in the data structure, say a

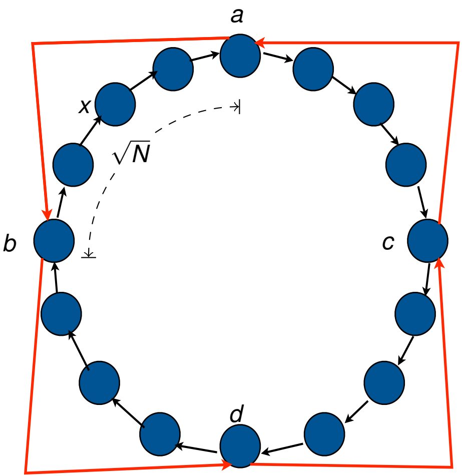
What happens to large cycles?



Compute $f(x)$, $f(f(x))$, ... till you hit a point in the data structure, say a

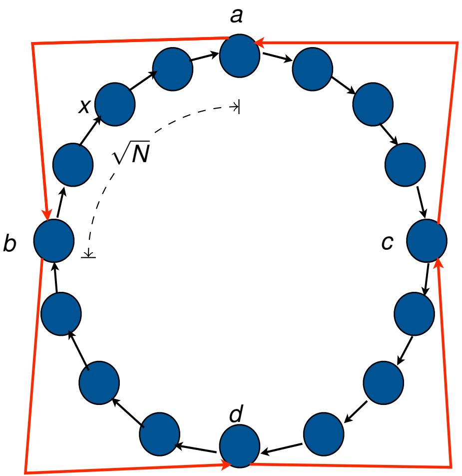
When you hit a, use back-link to go back to b

What happens to large cycles?



Now, compute $f(a)$, $f(f(a))$, ... until you hit x

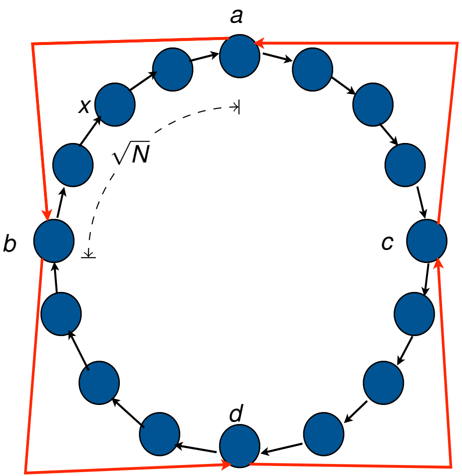
What happens to large cycles?



Now, compute $f(a)$, $f(f(a))$, ... until you hit x

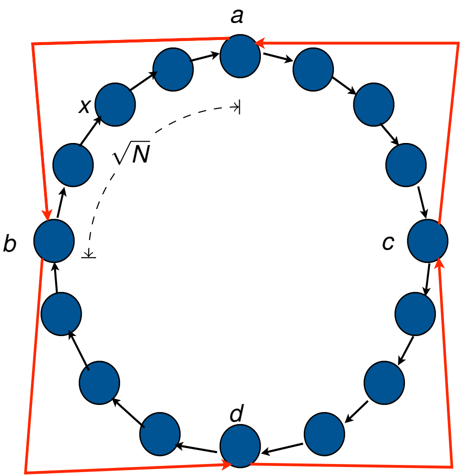
The penultimate point in the sequence is $f^{-1}(x)$

What happens to large cycles?



Note that all the cycles can be covered by $O(\sqrt{N})$ back-links (each back-link covering a distance of $\sqrt{N}$)

What happens to large cycles?



Note that all the cycles can be covered by $O(\sqrt{N})$ back-links (each back-link covering a distance of $\sqrt{N}$)

Also, the total time complexity is $\sqrt{N}$ as you hit a “back-link” in that time

Time and space complexity for inverting permutations

- Total time $T = \tilde{O}(\sqrt{N})$ and space $S = \tilde{O}(\sqrt{N})$.

Time and space complexity for inverting permutations

- Total time $T = \tilde{O}(\sqrt{N})$ and space $S = \tilde{O}(\sqrt{N})$.
- Can be used to invert ϵ fraction of the elements in time $T = \tilde{O}(\sqrt{\epsilon N})$ and space $S = \tilde{O}(\sqrt{\epsilon N})$
- In fact, we can achieve any time (T) space (S) tradeoff such that $T \cdot S = \epsilon N$.

Abstracting the approach for permutations

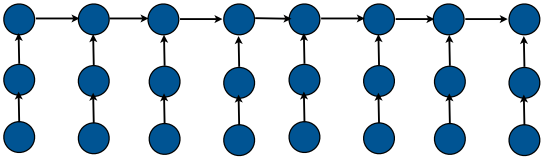
- Cover the graph ($x \to f(x)$) of f by m disjoint paths of length ℓ.

Abstracting the approach for permutations

- Cover the graph $(x \to f(x))$ of f by m disjoint paths of length ℓ.
- Gives algo with $T = \tilde{O}(\ell)$ and $S = \tilde{O}(m)$ (one back-link per path).

Abstracting the approach for permutations

- Cover the graph $(x \to f(x))$ of f by m disjoint paths of length ℓ.
- Gives algo with $T = \tilde{O}(\ell)$ and $S = \tilde{O}(m)$ (one back-link per path).
- Problem: m may have to be very large because the graph $(x \to f(x))$ may not have many long and disjoint paths.



Approach for random functions [Hellman, Fiat-Naor]

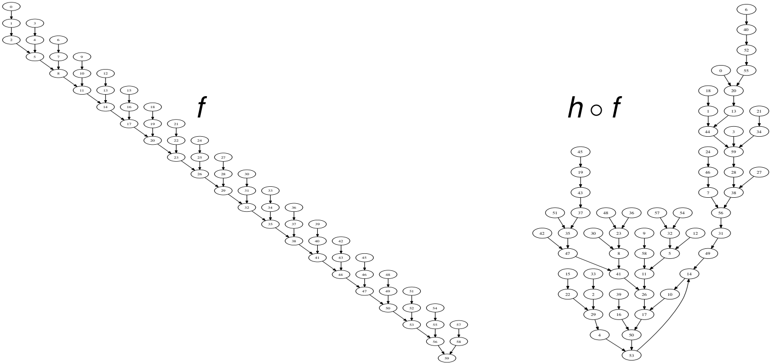
- Collision probability: $\lambda = \Pr_{x,x' \sim [N]}[f(x) = f(x')]$.

Approach for random functions [Hellman, Fiat-Naor]

- Collision probability: $\lambda = \Pr_{x,x' \sim [N]} [f(x) = f(x')]$.
- If h is a (known) permutation, then inverting $h \circ f$ suffices. If h is random and f has low collision probability, then $h \circ f$ has many long paths which are pairwise disjoint.

Approach for random functions [Hellman, Fiat-Naor]

- Collision probability: $\lambda = \Pr_{x,x' \sim [N]}[f(x) = f(x')]$.
- If h is a (known) permutation, then inverting $h \circ f$ suffices. If h is random and f has low collision probability, then $h \circ f$ has many long paths which are pairwise disjoint.



Inverting random functions ($\lambda \approx 1/N$)

- For independent random permutations $h_1, \ldots, h_r$, let $g_i = h_i \circ f$.

Inverting random functions ($\lambda \approx 1/N$)

- For independent random permutations $h_1, \ldots, h_r$, let $g_i = h_i \circ f$.
- For each g_i, can find m disjoint paths of length ℓ as long as $m \cdot \ell^2 \cdot \lambda \ll 1$. (each g_i inverts $m \cdot \ell$ elements).

Inverting random functions ($\lambda \approx 1/N$)

- For independent random permutations $h_1, \ldots, h_r$, let $g_i = h_i \circ f$.
- For each g_i, can find m disjoint paths of length ℓ as long as $m \cdot \ell^2 \cdot \lambda \ll 1$. (each g_i inverts $m \cdot \ell$ elements).
- If $g_i's$ behave independently, elements inverted by each of them are independent. Overall $O(m \cdot \ell \cdot r)$ elements inverted.

Inverting random functions ($\lambda \approx 1/N$)

- For independent random permutations $h_1, \ldots, h_r$, let $g_i = h_i \circ f$.
- For each g_i, can find m disjoint paths of length ℓ as long as $m \cdot \ell^2 \cdot \lambda \ll 1$. (each g_i inverts $m \cdot \ell$ elements).
- If $g_i' s$ behave independently, elements inverted by each of them are independent. Overall $O(m \cdot \ell \cdot r)$ elements inverted.
- Choose $m, \ell, r = \tilde{O}(N^{1/3})$.

$$\begin{aligned} T &= O(\ell \cdot r) = \tilde{O}(N^{2/3}) \\ S &= O(m \cdot r) = \tilde{O}(N^{2/3}) \end{aligned}$$

Inverting random functions ($\lambda \approx 1/N$)

- For independent random permutations $h_1, \ldots, h_r$, let $g_i = h_i \circ f$.
- For each g_i, can find m disjoint paths of length ℓ as long as $m \cdot \ell^2 \cdot \lambda \ll 1$. (each g_i inverts $m \cdot \ell$ elements).
- If $g_i's$ behave independently, elements inverted by each of them are independent. Overall $O(m \cdot \ell \cdot r)$ elements inverted.
- Choose $m, \ell, r = \tilde{O}(N^{1/3})$.

$$\begin{aligned} T &= O(\ell \cdot r) = \tilde{O}(N^{2/3}) \\ S &= O(m \cdot r) = \tilde{O}(N^{2/3}) \end{aligned}$$

- Problems: Computing $h_1, \ldots, h_r$ is hard. Heuristic works only for random f.

Inverting arbitrary functions [Fiat-Naor]

- Store a table of K elements with many pre-images. Collision probability restricted to the remaining inputs is $\approx 1/K$.

Inverting arbitrary functions [Fiat-Naor]

- Store a table of K elements with many pre-images. Collision probability restricted to the remaining inputs is $\approx 1/K$.
- Each h_i only needs to be an ℓ-wise independent hash function. Also, $h_1, \ldots, h_r$ only need to be pairwise independent.

Inverting arbitrary functions [Fiat-Naor]

- Store a table of K elements with many pre-images. Collision probability restricted to the remaining inputs is $\approx 1/K$.
- Each h_i only needs to be an ℓ-wise independent hash function. Also, $h_1, \ldots, h_r$ only need to be pairwise independent.
- Amortize time for one evauation each of $h_1, \ldots, h_r$ to $\tilde{O}(\ell + r)$.

$$T = (\text{time to compute } h_1, \ldots, h_r) \cdot \ell = \tilde{O}(\ell^2 + \ell \cdot r)$$

$$S = \tilde{O}(K + m \cdot r)$$

Inverting arbitrary functions [Fiat-Naor]

- Store a table of K elements with many pre-images. Collision probability restricted to the remaining inputs is $\approx 1/K$.
- Each h_i only needs to be an ℓ-wise independent hash function. Also, $h_1, \ldots, h_r$ only need to be pairwise independent.
- Amortize time for one evauation each of $h_1, \ldots, h_r$ to $\tilde{O}(\ell + r)$.

$$T = (\text{time to compute } h_1, \ldots, h_r) \cdot \ell = \tilde{O}(\ell^2 + \ell \cdot r)$$
$$S = \tilde{O}(K + m \cdot r)$$

- Can again choose m, l such that $m\ell^2\lambda \approx m\ell^2/K \ll 1$. Can get

$$T, S = \tilde{O}(N^{3/4})$$

by taking $K = \tilde{O}(N^{3/4})$, $r = \tilde{O}(N^{1/2})$ and $m, \ell = \tilde{O}(N^{1/4})$.

Inverting f on ϵ-fraction of inputs

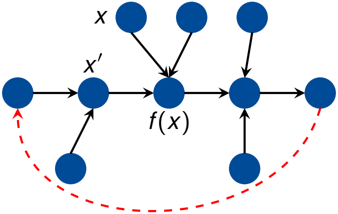
- Directly scaling the Fiat-Naor result would give complexity $(\epsilon N)^{3/4}$ (we claimed $\epsilon^{5/4} N^{3/4}$). Improved analysis using two simple ideas.

Inverting f on ϵ-fraction of inputs

- Directly scaling the Fiat-Naor result would give complexity $(\epsilon N)^{3/4}$ (we claimed $\epsilon^{5/4} N^{3/4}$). Improved analysis using two simple ideas.
- First observation: If a table of size K does not invert f with probability ϵ, then the collision probability for the rest is ϵ/K.

Inverting f on ϵ-fraction of inputs

- Directly scaling the Fiat-Naor result would give complexity $(\epsilon N)^{3/4}$ (we claimed $\epsilon^{5/4} N^{3/4}$). Improved analysis using two simple ideas.
- First observation: If a table of size K does not invert f with probability ϵ, then the collision probability for the rest is ϵ/K.
- Second Observation: The number of elements inverted by a path is not just the path length, but the the sum of indegrees of elements in the path.



Issues in analysis

- Problem: Probablities for inversion of moderately high indegree elements do not add up (with our parameters) in the graphs for $g_1, \ldots, g_r$. Also, these may not be in the table.

Issues in analysis

- Problem: Probablities for inversion of moderately high indegree elements do not add up (with our parameters) in the graphs for $g_1, \ldots, g_r$. Also, these may not be in the table.
  - Analyze these separately using a weaker bound.
  - Either weaker bound suffices or get better control on collision probability.

Issues in analysis

- Problem: Probablities for inversion of moderately high indegree elements do not add up (with our parameters) in the graphs for $g_1, \ldots, g_r$. Also, these may not be in the table.
  - Analyze these separately using a weaker bound.
  - Either weaker bound suffices or get better control on collision probability.

- Problem: Value of r is $O(1)$ for some ranges of ϵ and amortization over evaluations of $h_1, \ldots, h_r$ is not possible.

Issues in analysis

- Problem: Probablities for inversion of moderately high indegree elements do not add up (with our parameters) in the graphs for $g_1, \ldots, g_r$. Also, these may not be in the table.
  - Analyze these separately using a weaker bound.
  - Either weaker bound suffices or get better control on collision probability.

- Problem: Value of r is $O(1)$ for some ranges of ϵ and amortization over evaluations of $h_1, \ldots, h_r$ is not possible.
  - Use better construction based on lossless expanders of Capalbo et al. [CRVW02] and an observation of Seigel [Seigel89].
  - Take $\ell^{o(1)}$ time per evaluation.

Issues in analysis

- Problem: Probablities for inversion of moderately high indegree elements do not add up (with our parameters) in the graphs for $g_1, \ldots, g_r$. Also, these may not be in the table.
  - Analyze these separately using a weaker bound.
  - Either weaker bound suffices or get better control on collision probability.

- Problem: Value of r is $O(1)$ for some ranges of ϵ and amortization over evaluations of $h_1, \ldots, h_r$ is not possible.
  - Use better construction based on lossless expanders of Capalbo et al. [CRVW02] and an observation of Seigel [Seigel89].
  - Take $\ell^{o(1)}$ time per evaluation.

- Final complexity: $T, S \;=\; \begin{array}{ll} \tilde{O}(\sqrt{\epsilon N}) & \epsilon \leq N^{-1/3} \\ \tilde{O}(\epsilon^{5/4} N^{3/4}) & \epsilon \geq N^{-1/3} \end{array}$

Lower bound for inverting permutations

- Given A inverting f on ϵ fraction of inputs in time T and space S, want to show $T \cdot S = \Omega(\epsilon N)$.

Lower bound for inverting permutations

- Given A inverting f on ϵ fraction of inputs in time T and space S, want to show $T \cdot S = \Omega(\epsilon N)$.
- Showed by [Yao90] for $\epsilon = 1$ and [GT00], [Wee05] when $T = O(\sqrt{\epsilon N})$.

Lower bound for inverting permutations

- Given A inverting f on ϵ fraction of inputs in time T and space S, want to show $T \cdot S = \Omega(\epsilon N)$.
- Showed by [Yao90] for $\epsilon = 1$ and [GT00], [Wee05] when $T = O(\sqrt{\epsilon N})$.
- Give a simpler, "randomized" proof that works for all T. Also extends to lower bounds for PRGs.

Lower bound for inverting permutations

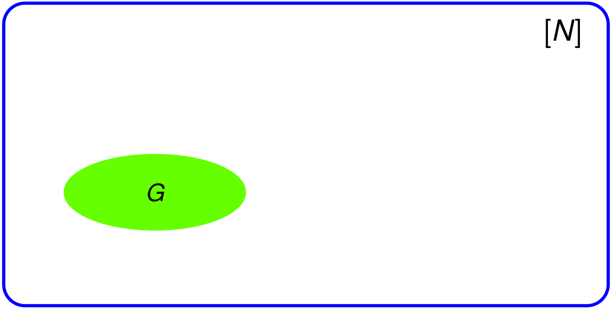
- Given A inverting f on ϵ fraction of inputs in time T and space S, want to show $T \cdot S = \Omega(\epsilon N)$.
- Showed by [Yao90] for $\epsilon = 1$ and [GT00], [Wee05] when $T = O(\sqrt{\epsilon N})$.
- Give a simpler, "randomized" proof that works for all T. Also extends to lower bounds for PRGs.
- As in [GT00], show that using A, can encode f with $\approx \log(N!) - \phi(N, T) + S$ bits for some ϕ. Thus, $S > \phi(N, T)$ giving the tradeoff between T and S.

Lower bound for inverting permutations

- Given A inverting f on ϵ fraction of inputs in time T and space S, want to show $T \cdot S = \Omega(\epsilon N)$.
- Showed by [Yao90] for $\epsilon = 1$ and [GT00], [Wee05] when $T = O(\sqrt{\epsilon N})$.
- Give a simpler, "randomized" proof that works for all T. Also extends to lower bounds for PRGs.
- As in [GT00], show that using A, can encode f with $\approx \log(N!) - \phi(N, T) + S$ bits for some ϕ. Thus, $S > \phi(N, T)$ giving the tradeoff between T and S.
- We show that using A, one can encode f using $\approx \log(N!) - \frac{\epsilon N}{100T} + S$ bits giving us the desired tradeoff.

Intuition for the encoding

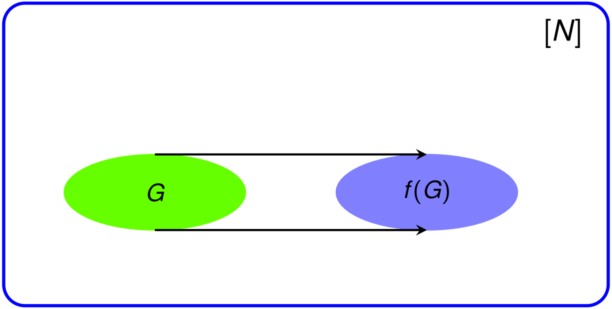
Intuition for the encoding



$$|G| = \frac{\epsilon N}{100T}$$

- A inverts G correctly.
- For all $x \in G$, A does not query any element in G.

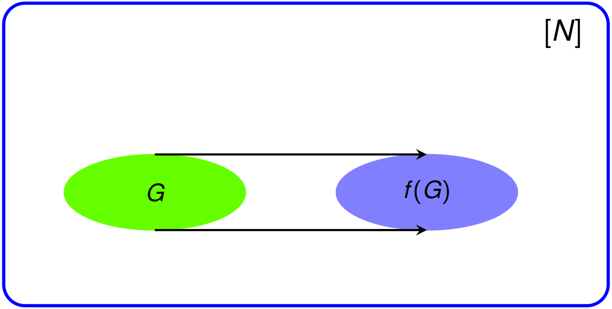
Intuition for the encoding



$$|G| = \frac{\epsilon N}{100T}$$

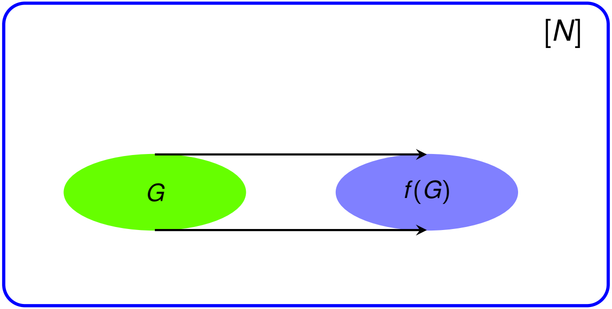
- A inverts G correctly.
- For all $x \in G$, A does not query any element in G.

Intuition for the encoding



$$|G| = \frac{\epsilon N}{100T}$$

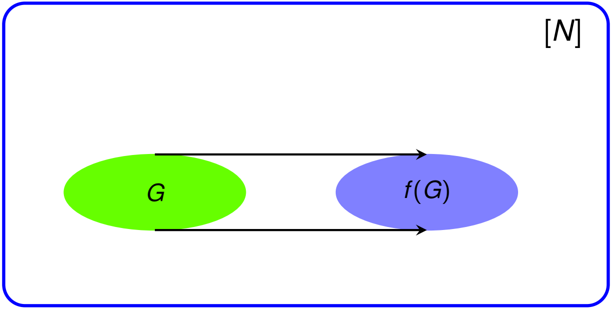
Intuition for the encoding



$$|G| = \frac{\epsilon N}{100T}$$

- Complexity of encoding :=
  - Size of G
  - Specify set $f(G)$
  - Specify the map f^{-1} on $[N] - f(G)$

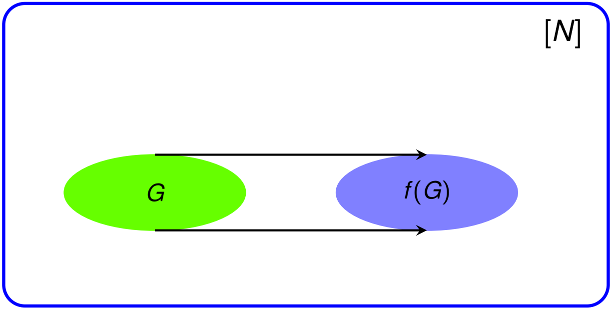
Intuition for the encoding



$$|G| = \frac{\epsilon N}{100T}$$

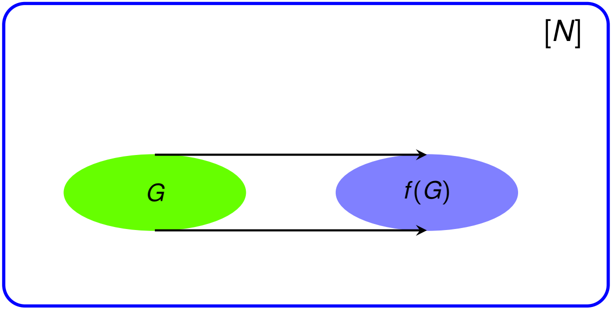
- Complexity of encoding :=
  - Size of G
  - Specify set $f(G)$
  - Specify the map f^{-1} on $[N] - f(G)$
- This information along with A suffices to specify f entirely

Intuition for the encoding



$$|G| = \frac{\epsilon N}{100T}$$

Intuition for the encoding



$$|G| = \frac{\epsilon N}{100T}$$

- Total complexity of encoding : $2\log\binom{N}{|G|} + \log(N - |G|)!$
- Putting $|G| = \frac{\epsilon N}{100T}$, we get that $S + \frac{\epsilon N}{T}\log(T^2/\epsilon^2 N) \geq 0$

Upshot of the analysis

- Provided $T \leq \epsilon\sqrt{N}$, $TS = \tilde{\Omega}(\epsilon N)$

Upshot of the analysis

- Provided $T \leq \epsilon\sqrt{N}$, $TS = \tilde{\Omega}(\epsilon N)$
- This was the analysis by Gennaro and Trevisan [GT00]

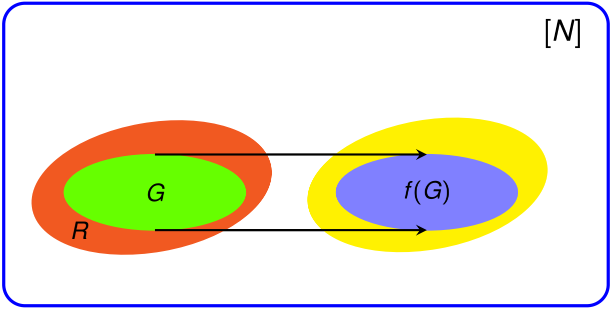
Upshot of the analysis

- Provided $T \leq \epsilon\sqrt{N}$, $TS = \tilde{\Omega}(\epsilon N)$
- This was the analysis by Gennaro and Trevisan [GT00]
- The analysis was improved by Wee [Wee05] who showed $TS = \tilde{\Omega}(\epsilon N)$ provided $T \leq \sqrt{\epsilon N}$

Upshot of the analysis

- Provided $T \leq \epsilon\sqrt{N}$, $TS = \tilde{\Omega}(\epsilon N)$
- This was the analysis by Gennaro and Trevisan [GT00]
- The analysis was improved by Wee [Wee05] who showed $TS = \tilde{\Omega}(\epsilon N)$ provided $T \leq \sqrt{\epsilon N}$
- There is still a gap because “deterministically” deciding on G is very expensive.

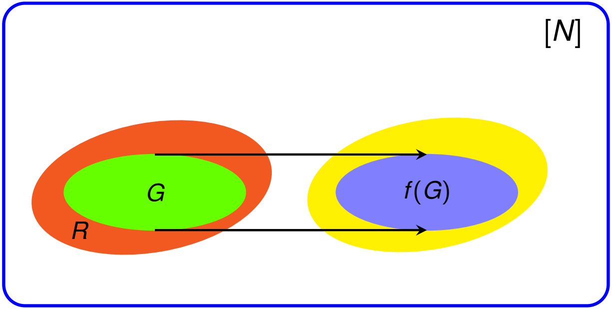
Randomized encoding



$$|G| = \frac{\epsilon N}{100T}$$

$$|R| = \frac{N}{10T}$$

Randomized encoding

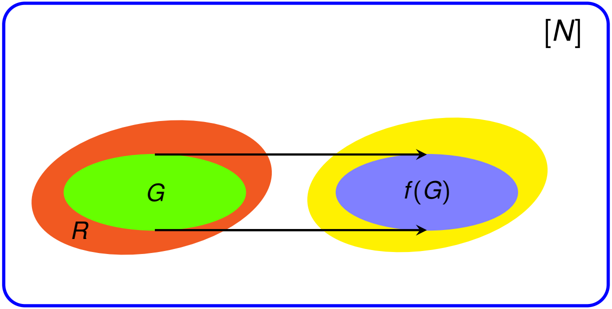


$$|G| = \frac{\epsilon N}{100T}$$

$$|R| = \frac{N}{10T}$$

- Choose R to be a set of size $N/10T$ uniformly at random.

Randomized encoding

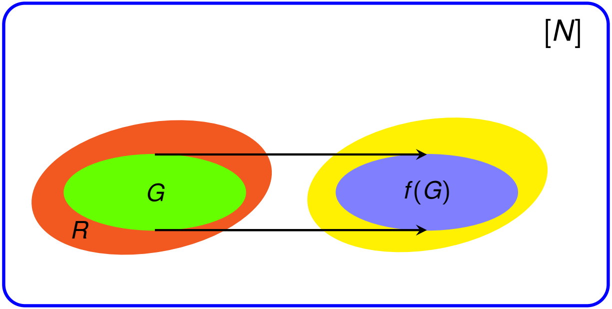


$$|G| = \frac{\epsilon N}{100T}$$

$$|R| = \frac{N}{10T}$$

- Choose R to be a set of size $N/10T$ uniformly at random.
- With high probability, this contains a set G of size $\frac{\epsilon N}{100T}$ such that
    - A inverts G correctly.
    - For all $x \in G$, A does not query any element in R

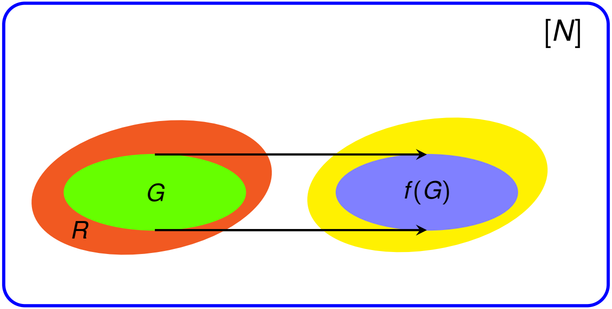
Randomized encoding



$$|G| = \frac{\epsilon N}{100T}$$

$$|R| = \frac{N}{10T}$$

Randomized encoding

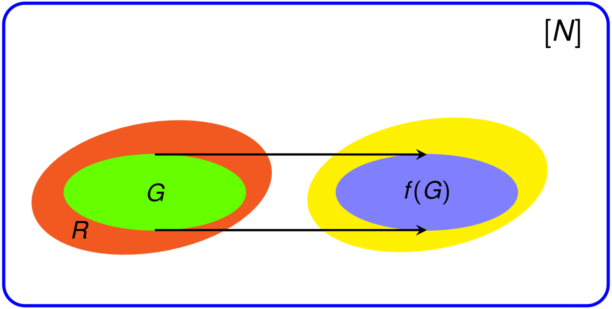


$$|G| = \frac{\epsilon N}{100T}$$

$$|R| = \frac{N}{10T}$$

- Some savings in the analysis as the identity of R is already known
- Once we know f outside R, we need to know “G in R” as opposed to “G in [N]” - main source of saving

Randomized encoding



$$|G| = \frac{\epsilon N}{100T}$$

$$|R| = \frac{N}{10T}$$

- Some savings in the analysis as the identity of R is already known
- Once we know f outside R, we need to know "G in R" as opposed to "G in [N]" - main source of saving
- In all, we can describe the permutation in $\log(N!) - \epsilon N/100T + S$ bits which gives us the result.

Conclusions

- Non-uniform attacks can do better than uniform attacks on one-way functions and PRGs

Conclusions

- Non-uniform attacks can do better than uniform attacks on one-way functions and PRGs
- The best provable upper bound for one-way functions on all inputs remains $N^{3/4}$ and $N^{2/3}$ is the best for "Hellman"-style arguments (Barkan, Biham and Shamir)

Conclusions

- Non-uniform attacks can do better than uniform attacks on one-way functions and PRGs
- The best provable upper bound for one-way functions on all inputs remains $N^{3/4}$ and $N^{2/3}$ is the best for "Hellman"-style arguments (Barkan, Biham and Shamir)
- Techniques for proving lower bounds do not seem to do any better for one-way functions than permutations i.e. $\Omega(N^{1/2})$.

Thank You

Questions?