Improved Search-to-Decision Reduction for Random Local Functions

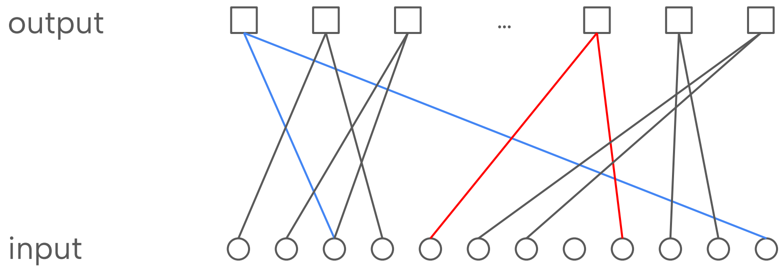
SG Crypto Workshop

Tan Kel Zin, Prashant Nalini Vasudevan

National University of Singapore

Local Cryptography

Is it possible to construct Cryptography Primitives (PRG, OWF) with constant depth circuit



Each output bit only depends on constant number of input bits

NC0 class: Functions definable by Constant Depth Circuit with bounded fan-in gates

Applications

- Fast Parallel Cryptography
- iO Constructions [LV17, JLS21, JLS22]
- Secure Computation [ADI+17, BCG+17, BCM23, BCM+24]

Feasibility of Local Cryptography

Negative Results

- Impossibility of local PRG with n^(polylog(n)) outputs [LMN93]
- Impossibility of local PRG with superlinear stretch in NC0 depth 3 [CM01]
- Impossibility of local PRG with superlinear stretch NC0 depth 4 [MST03]

Positive Results

- Existence of local OWF and sublinear stretch local PRG, assuming OWF and PRG in NC1 (Log Depth) [AIK06]
- Construction of linear stretch local PRG assuming the hardness of the average case MAX-3LIN problem [AIK08]

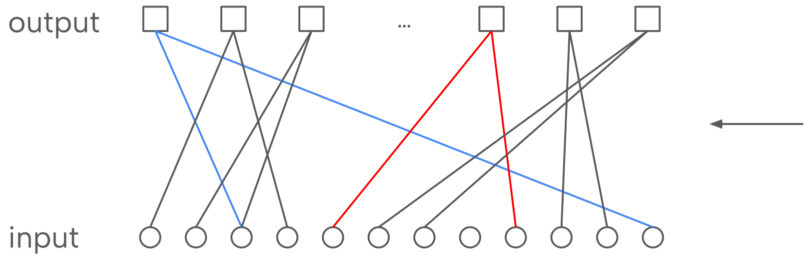
Is there a more natural Construction of local PRG and OWF?

Random Local Function [Gol00]

Fix a d–ary predicate function $P : \mathbb{F}_2^d \rightarrow \mathbb{F}_2$ for some constant d >= 3

$$P(s) = s_1 + s_2 + s_3 + s_4 s_5$$

For each output bit, randomly select input bits and feed to the predicate



Randomly sample a bipartite graph G

Inputs for 2nd output bit $S_2 = (2, 4, 1, 5, 6)$

2nd output bit $f_{G,P}(s)_2 = P(s_2, s_4, s_1, s_5, s_6)$

Candidate Local OWF: Given $(G, f_{G,P}(s))$, recover input s

$$f_{G,P} : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$$

$$G = (S_1, S_2, \ldots, S_m)$$

$$S_i = (j_1, \ldots, j_d)$$

$$f_{G,P}(s)_i = P(s_{j_1}, \ldots, s_{j_d})$$

Hardness of Inversion

$$f_{G,P} : \mathbb{F}_2^n \to \mathbb{F}_2^m$$
$$P : \mathbb{F}_2^d \to \mathbb{F}_2$$
$$G = (S_1, S_2, \ldots, S_m)$$
$$S_i = (j_1, \ldots, j_d)$$
$$f_{G,P}(s)_i = P(s_{j_1}, \ldots, s_{j_d})$$

Evidence of Hardness

- Myopic and Drunken backtracking algorithms fails on Predicate with linear components [AHI05, CEMT09, Its10]

$$P(s) = s_1 + \cdots + s_k + Q(s_{k+1}, \ldots, s_d)$$

- XOR-AND(3,2) with output length n^1.49 secure against F2-linear tests and semi-definite programming algorithms [OW14]

$$P(s) = s_1 + s_2 + s_3 + s_4 s_5$$

Negative Hardness

- Linear & Degenerate Predicate [Gol00]
- Low F2 Algebraic Degree [AL16]
- High correlation with small subset of inputs [BQ09, AL16, App16]
- Regardless of Predicate, when output size $m = O(n^{\frac{1}{2}\lfloor 2d/3 \rfloor} \log n)$ efficient algorithm exists [App16]

Suggested Predicate [AL16]

$$P(s) = s_1 + \cdots + s_k + Maj(s_{k+1}, \ldots, s_d)$$

Candidate Local PRG

Take inversion as ***search***

Distinguishing output from random is ***decision***

Search: Given $(G, f_{G,P}(s))$, find s

Decision: Given G, distinguish $f_{G,P}(s)$ from random binary string

$$f_{G,P} : \mathbb{F}_2^n \to \mathbb{F}_2^m$$
$$P : \mathbb{F}_2^d \to \mathbb{F}_2$$
$$G = (S_1, S_2, \dots, S_m)$$
$$S_i = (j_1, \dots, j_d)$$
$$f_{G,P}(s)_i = P(s_{j_1}, \dots, s_{j_d})$$

Search-to-decision reduction gives an candidate for Local PRG

Search Hardness => Decision Hardness

Local OWF => Local PRG

IDEAL : Search with m outputs is hard, then Decision with m outputs is hard

CURRENTLY : Search with m outputs is hard, then Decision with much less than m outputs is hard

Previous Result

Suppose $P : \mathbb{F}_2^d \to \mathbb{F}_2$ is ***sensitive*** (Flipping a variable changes the output)

$$P(s) = s_1 + Q(s_2, \ldots, s_d)$$

Given an ε advantage decision algorithm for ***sensitive*** predicate with output size m = poly(n)

Then there exists a search algorithm for the same predicate with

- [App12] output size $\tilde{O}(m^3/\varepsilon^2)$, locality d
- [BRT25] output size $\tilde{O}(nm/\varepsilon^2)$, locality d+1

Why is the sensitivity of predicate necessary?

Our Result

Main Theorem

Given an $\mathcal{E}$ advantage decision algorithm for any predicate with output size m = poly(n)

Then there exists a search algorithm for the same predicate with

output size $\tilde{O}(n^2m/\varepsilon^2)$, locality d

Comparison with previous result

All reductions the sensitivity of Predicate

- [App12] output size $\tilde{O}(m^3/\varepsilon^2)$, locality d
- [BRT25] output size $\tilde{O}(nm/\varepsilon^2)$, locality d+1

Our result lose a factor of n compared to [BRT25] but generalized to any predicate and maintained locality

Implication of Our Result

- Opens up the possibility of local PRG from more predicates
- Lack of sensitivity means less structure, could be harder for attacks

Technical Contribution

- Our technique differs from [App12, BRT25], relies minimally on the predicate
- Our technique of performing non-trivial mixing on the hypergraphs could be useful in other fields

Overview of Technique

Notations

Denote $x \leftarrow D$ as x is sampled from D; uniformly sample if D is a set

Denote x_i as the i-th bit of binary string x

Denote $G_{n,m,d}$ as the set of all hypergraphs with

- n vertices
- m ordered hyperedges
- d vertices in each hyperedge

$$G = (S_1, S_2, \ldots, S_m)$$
$$S_i = (j_1, \ldots, j_d)$$

Predictor

Using a Distinguisher with advantage $\mathcal{E}$

Construct algorithms $\mathsf{S}_2, \mathsf{S}_3, \ldots, \mathsf{S}_n$

Such that each S_i given an input $(G, f_{G,P}(s))$ can predict $s_1 \oplus s_i$

with small advantage $\Omega(\varepsilon/t)$ where $t = O(n \log(n/\varepsilon))$

Amplification

The prediction can be amplified with independent inputs over the same secret

Predictor Algorithm

We construct a randomised transformation T such that

- If $s_1 = s_i$, then $(T(G), f_{G,P}(s))$ is $(G, f_{G,P}(s))$
- If $s_1 \neq s_i$, then $(T(G), f_{G,P}(s))$ looks like (G, b) , where b is random

The predictor simply fed $(T(G), f_{G,P}(s))$ to the Distinguisher and output its response

Key of proof: How close are $(T(G), f_{G,P}(s))$ and (G, b) when $s_1 \neq s_i$

Transformation

Objective

- If $s_1 = s_i$, then $(T(G), f_{G,P}(s))$ is $(G, f_{G,P}(s))$
- If $s_1 \neq s_i$, then $(T(G), f_{G,P}(s))$ looks like (G, b) , where b is random

Transformation

Define Transformation $T_{a,b} : G_{n,m,d} \to G_{n,m,d}$, where $a, b \in [n]$

For each hyperedge $S_i = (j_1, \ldots, j_d)$ in $G = (S_1, S_2, \ldots, S_m)$

Transform in to $S'_i = (j'_1, \ldots, j'_d)$ and new hypergraph $G' = (S'_1, S'_2, \ldots, S'_m)$

- If j is not a or b, it remains the same
- If j is a or b, it switches to the other value with prob half

$$(1, 2, a, 3, b, a) \to (1, 2, a, 3, a, b)$$

$$\Pr[j'_k = j_k] = 1 \quad \text{if } j_k \notin \{a, b\}$$

$$\Pr[j'_k = a] = \frac{1}{2}, \ \Pr[j'_k = b] = \frac{1}{2} \quad \text{if } j_k \in \{a, b\}$$

Key Property

- The uniform distribution is stable under the transformation

  $$T(G_{n,m,d}) = G_{n,m,d}$$

- If $s_a = s_b$, the transformation has no effect on the distribution

  Output remains the same

  $P(s_1, s_a, s_3, s_a) = P(s_1, s_a, s_3, s_b)$

  $f_{G,P}(s) = f_{T_{a,b}(G),P}(s)$

  $\Longrightarrow$

  Distribution is the same

  $$(T(G), f_{G,P}(s)) = (T(G), f_{T(G),P}(s)) \approx (G, f_{G,P}(s))$$

- If $s_a \neq s_b$, transformed distribution looks closer to random (effective)

  Output might not be same

  $f_{G,P}(S) \neq f_{T_{a,b}(G),P}(S)$

  $\Longrightarrow$

  Hypergraph and Output less coupled

  $T(G)$ Less coupled with $f_{G,P}(s)$

After $t = O(n \log(nm/\varepsilon))$ transformation, hypergraph is independent of $f_{G,P}(s)$

Transformation

Define Transformation $T_{a,b} : G_{n,m,d} \to G_{n,m,d}$, where $a, b \in [n]$

For each hyperedge $S_i = (j_1, \ldots, j_d)$ in $G = (S_1, S_2, \ldots, S_m)$

- If j is not a or b, it remains the same
- If j is a or b, it switches to the other value with prob half

Key Property

After $t = O(n \log(nm/\varepsilon))$ transformation, hypergraph is independent of $f_{G,P}(s)$

For any starting hypergraph G, randomly choose ai and bi

$$T_{a_t,b_t} \circ \ldots \circ T_{a_1,b_1}(G) \approx \text{Uniform}(G_{n,m,d})$$

$$(T_{a_t,b_t} \circ \ldots \circ T_{a_1,b_1}(G), f_{G,P}(s)) \approx (G', f_{G,P}(s))$$

For a randomly sampled G′

Transformation

Define Transformation $T_{a,b} : G_{n,m,d} \to G_{n,m,d}$, where $a, b \in [n]$

For each hyperedge $S_i = (j_1, \ldots, j_d)$ in $G = (S_1, S_2, \ldots, S_m)$

- If j is not a or b, it remains the same
- If j is a or b, it switches to the other value with prob half

Proof Intuition

- Think of transformation as a markov process, $t = O(n \log(nm/\varepsilon))$ is the mixing time
- After t transformations, every value would be touched by several transformation
- The random assignment happens independently for each a or b, it quickly randomises the whole hypergraph

Summary

Question : How close are $(T(G), f_{G,P}(s))$ and (G, b) when $s_1 \neq s_i$

Answer : After $t = O(n \log(nm/\varepsilon))$ transformation,

$$(T_{a_t,b_t} \circ \ldots \circ T_{a_1,b_1}(G), f_{G,P}(s)) \approx (G, b)$$

Therefore one transformation is a step closer to random

Algorithm S_i given input $(G, f_{G,P}(s))$ predict $s_1 \oplus s_i$

Predictor samples a random $r \leftarrow [0, t-1]$ apply r random transformations T_{a_i,b_i} on G

Then apply $T_{1,i}$ once

The predictor simply fed $(T(G), f_{G,P}(s))$ to the Distinguisher and output its response

Generalization

Non-Constant Sparsity

Our technique works for random local function with non constant sparsity $d = \text{polylog}(n)$

Minor additional requirement on the advantage ε and output size m

Distinct values in the hyperedges

A common model for random local function is to have distinct values in each hyperedge

Our reduction still work with a slight loss in constant factor

Noisy Predicate

Motivated by Learning Parity with Error Problem (LPN), we showed that our reduction still applies when the predicate is added with some random noise

Conclusion

Open Problems

- Further lessen the gap of the output size between search and decision
- Utilize the reduction technique on other problems
- Find alternative family of Predicate (aside XOR-MAJ) and show OWF hardness

Thank You