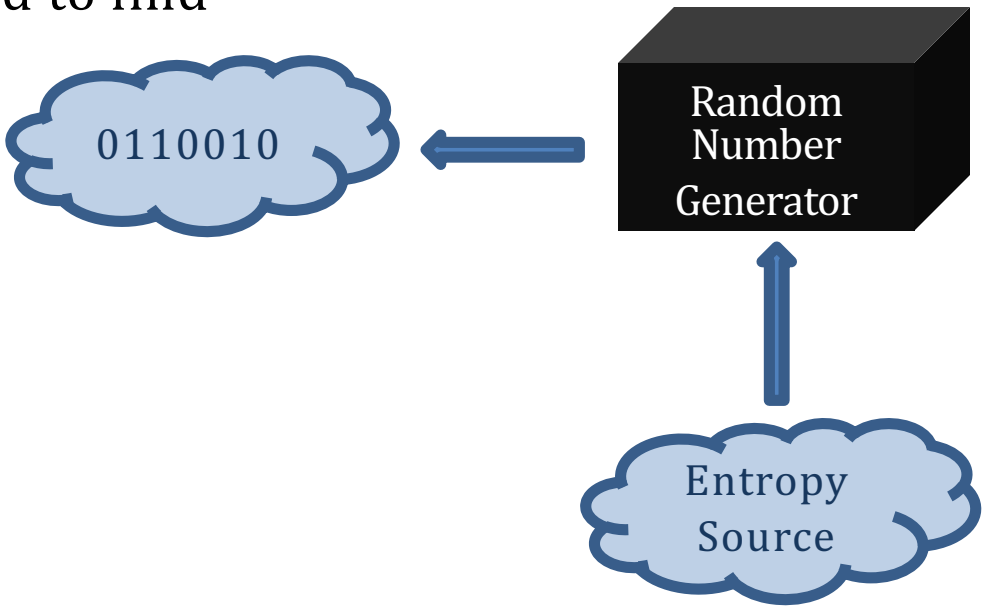
RANDOM NUMBER GENERATION USING HUMAN GAMEPLAY

Setareh Sharifian

October 2018

Randomness Generation

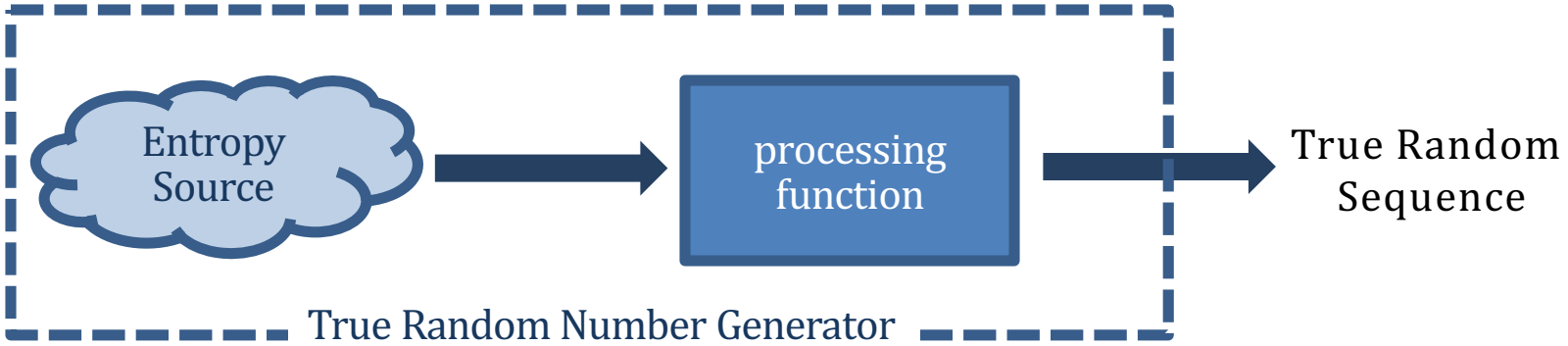
- ✓ Good randomness is hard to find



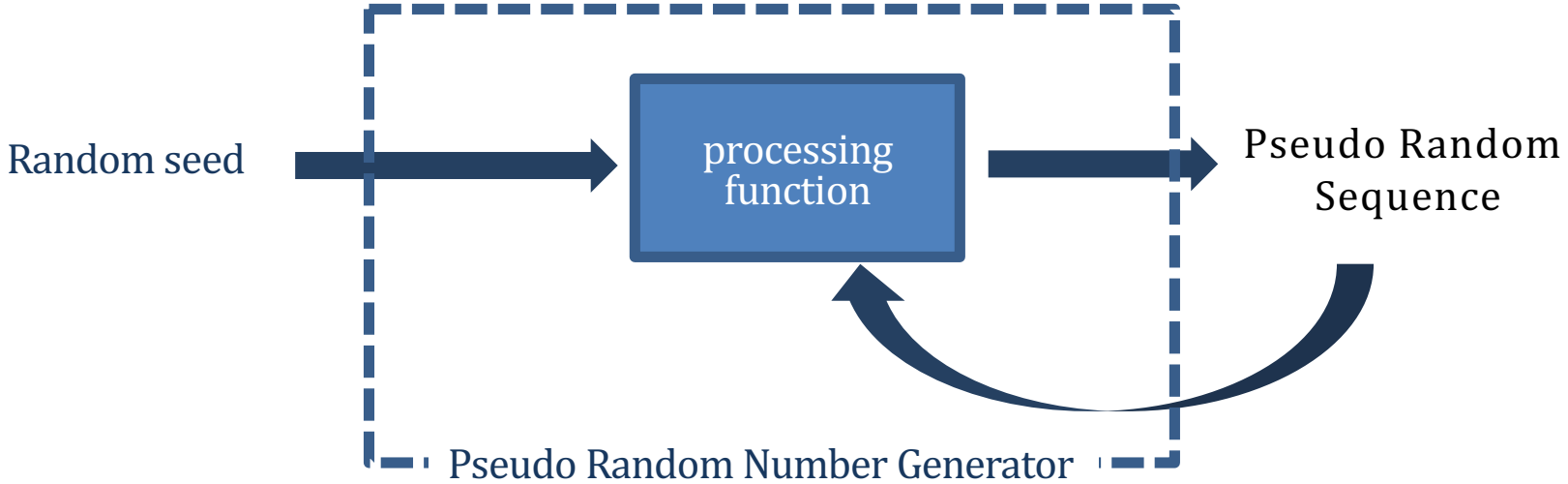
- ✓ Entropy Source
  - Physical Sources
  - Real time data
  - User input

Randomness Generation Needs an Entropy Source

- True Random Number Generator

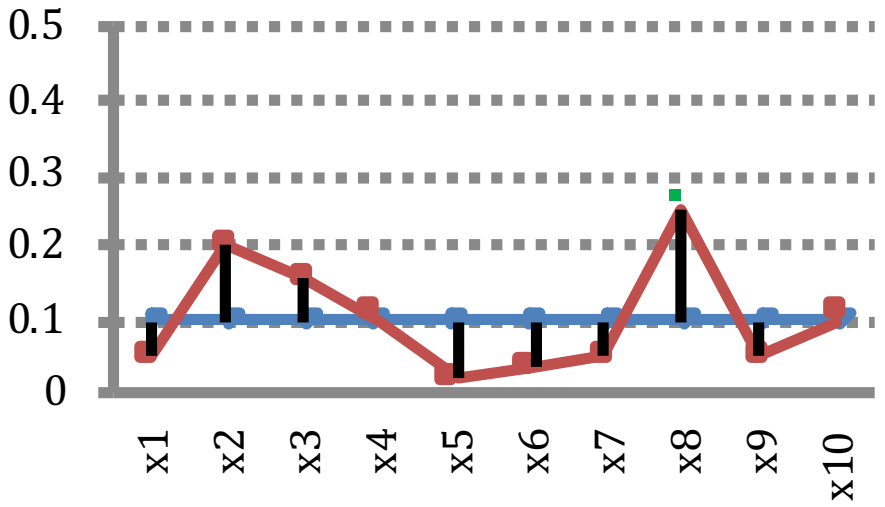


- Pseudo Random Number Generator

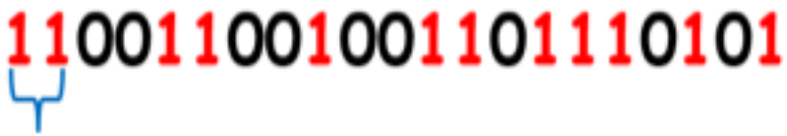


Measures of Randomness

- A sequence is random if it is sampled from uniform distribution
- To measure randomness, find/estimate the distribution of your samples and compare the distribution with uniform
  - Statistical distance
  - Shannon's entropy
  - Min-entropy
    - Min-entropy tests



- Randomness tests



Human as a Source of Randomness

✓Human Random behavior

Result: Human failed

100110010

✓Human does a poor job in generating randomness.

Idea: Design a game for leading human toward random behavior

✓Why Games?

- The competitive nature of the game makes humans act more randomly when playing games
- Playing games is more entertaining to users than simply "supplying entropy".

How Can we lead Human to play Randomly?

- Zero-sum Games: Matching Pennies

Player *II*

Player *I*	Head	Tail
Head	(1,-1)	(-1,1)
Tail	(-1,1)	(1,-1)

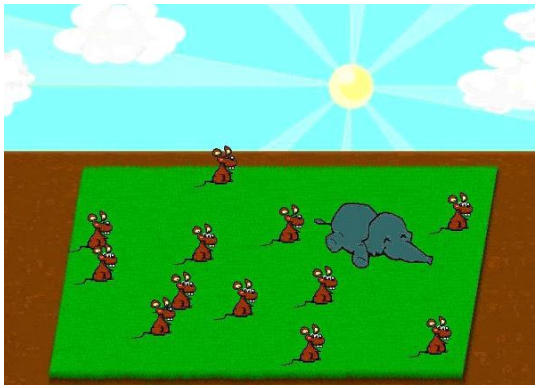
- The optimal strategy is to play uniformly random
- Idea for leading human to play randomly:
  - Design a game so that random strategy be the best strategy for players

Human's Gameplay as an Entropy Source

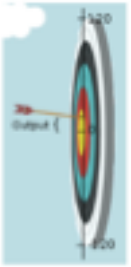
- Rapoport et al. showed that zero-sum competitive games such as matching pennies can lead human to generate better randomness but there are some issues regarding human gameplay:
  - Is the game interesting enough?
  - Is the length of the output appropriate for cryptographic applications?
  - How many times the game needs to be played to get the desired result?
  - How unpredictable is the output?

Randomness Generation from Human's Gameplay: Two Approaches

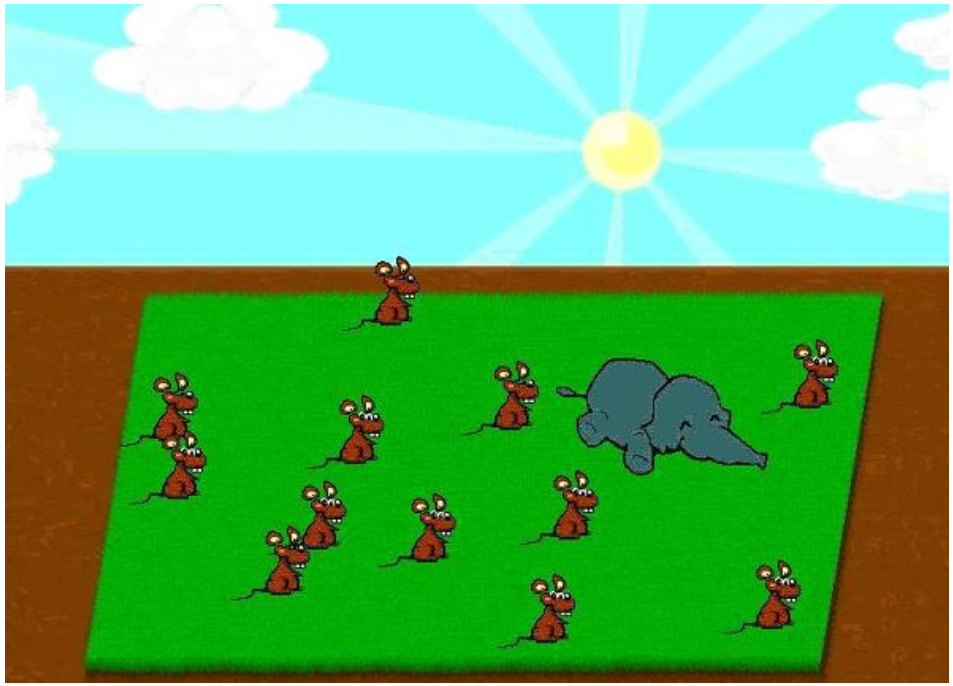
- Halprin et al. approach:
  - Use the extension of matching pennies in 2 dimensions to lead the human player to generate randomness and then apply a seeded extractor on the output of the game to generate The final output.



- Alimomeni et al. approach:
  - Use an indirect approach to entropy collection from human input . The goal is to collect randomness from user's mistakes . Players do their best to play perfectly, but cannot and so error happens. Indeed this game makes the user focussed on something else and then extract randomness from the unconscious behaviour of the user



Mice and Elephant[HN09]



- Human positions r mice
- Computer positions elephant
- Repeat until a mouse is crushed

Halprin et al. Approach

- Natural extension of Matching Pennies in two dimensions
  - Zero sum
  - The optimal strategy is to play uniformly random
- Game produces $\log_2(n)$ bits of raw data per move
- The output of the game is not uniformly random
  - Estimate the randomness and min-entropy empirically
  - Extract uniformly random strings from the output of the game

Use Randomness Extractor

Randomness Extractors

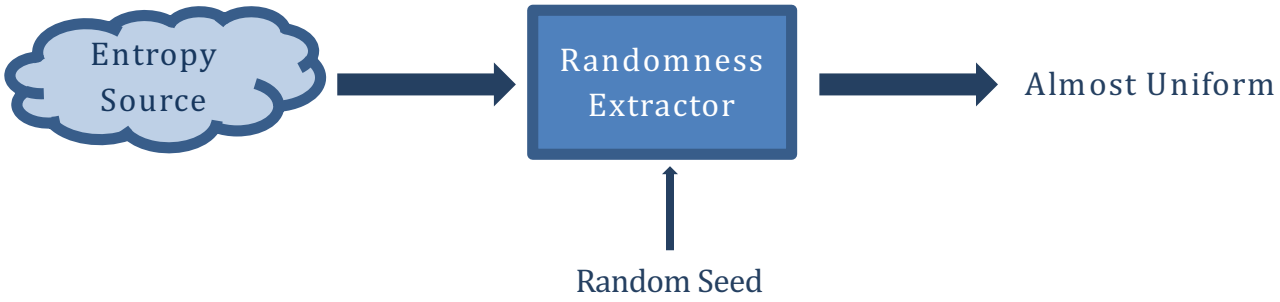
- Functions that extracts almost uniform bits from a source of imperfect randomness
  - Deterministic Extractors $(Ext(X))$

  One input with special distribution (i.e. binary IID samples)
  - Seeded Extractors $(Ext(S, X))$

  Two inputs: short uniformly random seed(S) + long input from an entropy source (X)
    - Entropy source is a k-source: a source X such that $H_\infty(X) \geq k$

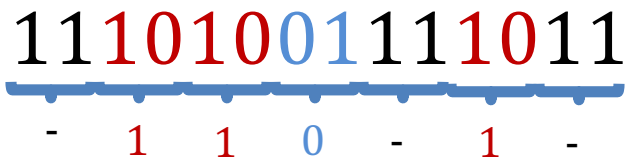
$$\text{Strong } (k, \epsilon)\text{-extractor: } SD((\mathrm{S}, \mathrm{Ext}(\mathrm{S}, \mathrm{X})); (\mathrm{S}, \mathrm{U}_{\mathrm{out}})) \leq \epsilon$$

$\mathrm{U}_{\mathrm{out}}$ is uniform distribution over the output set



Examples of Randomness Extractors:

- Deterministic Extractors
  - Von Neumann extractor: For an identical independent distributed binary sequence



- Seeded Extractors for a k-source
  - Universal Hash function: used in [HN09]

$$Ext(s, x) = h_s(x)$$
$$|Ext(s, x)| \approx k$$

  - Expander Graphs: used in [ASNS13]

$$|\mathcal{S}| \approx \log |Ext(s, x)|$$
$$|Ext(s, x)| < k$$

Hash Functions

- **Pair-wise independent hash function:**

  A family of functions $\mathcal{H} = \{h_s | s \in \mathcal{S}\}: \{0,1\}^n \rightarrow \{0,1\}^m$ is called a family of pairwise independent hash functions if for all distinct x, $y \in \{0,1\}^n$ and $a, b \in \{0,1\}^m$

$$\Pr[h_s(x) = a \wedge h_s(y) = b] \leq 2^{-2m}$$

  - *Example of pair-wise independent hash function:* $h_s: \{0,1\}^m \rightarrow \{0,1\}^m$

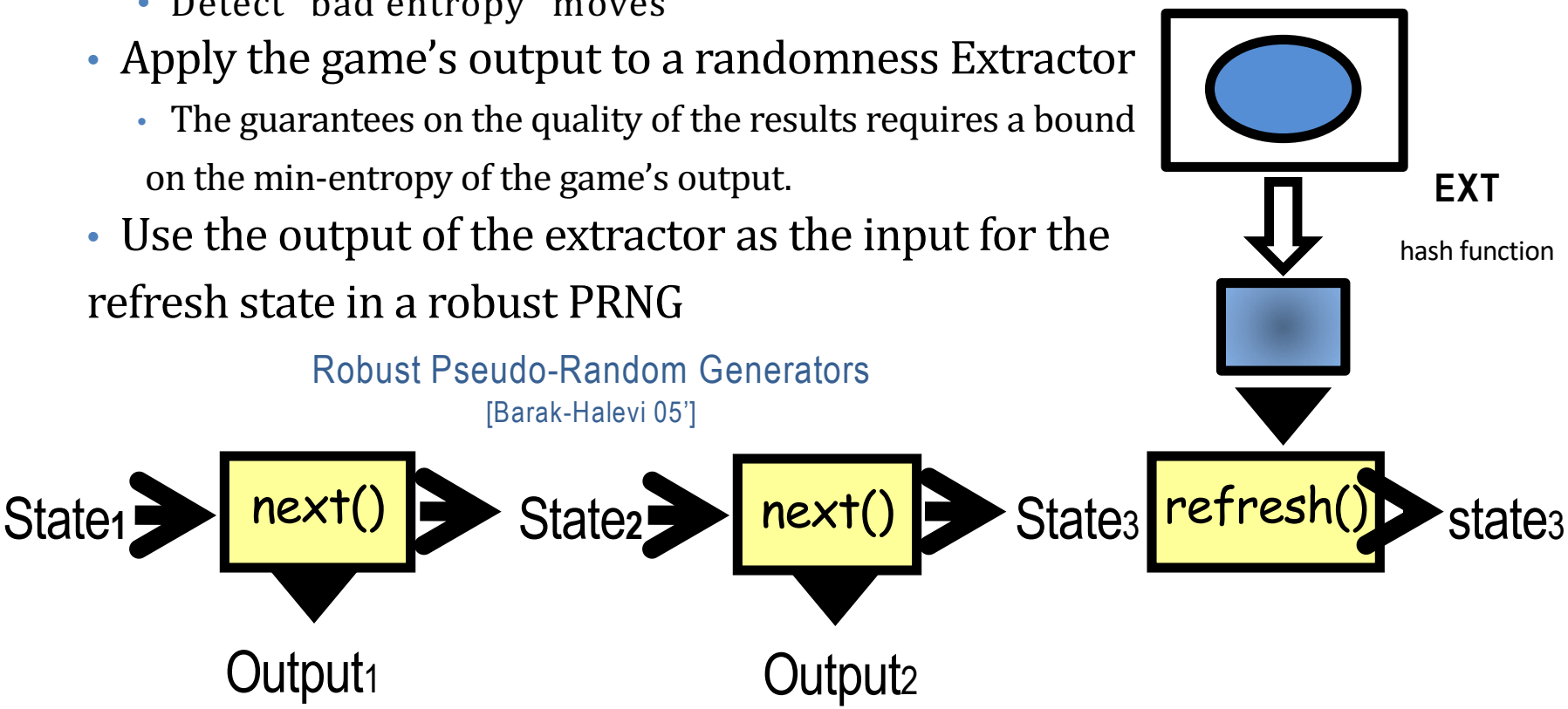
$$h_s(x) = s_0 + s_1 . x$$

$$s = (s_0, s_1) \rightarrow |\mathcal{S}| = 2^{2m}$$

  - **Leftover Hash Lemma:** Let $\mathcal{H} = \{h_s\}$ be a family of pairwise independent hash functions h_s: $\{0,1\}^n \rightarrow \{0,1\}^m$. For every ε > 0 and a randomly chosen seed s, the function $Ext(x, s) = h_s(x)$ is an $(m + 2\log 1/\varepsilon, \varepsilon)$-extractor.

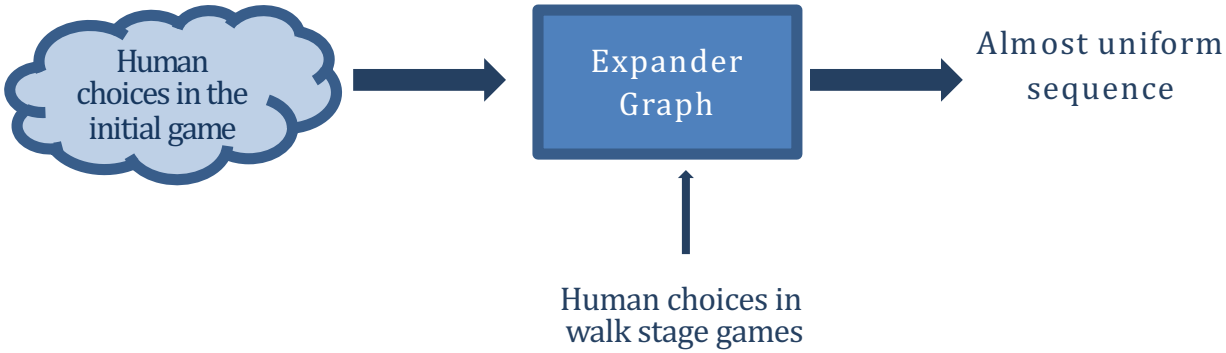
Randomness Generation in Halprin et al. Approach

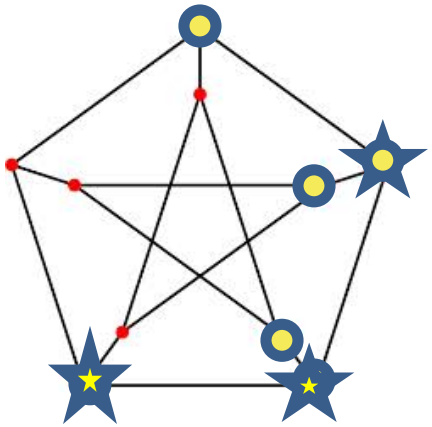
- When entropy is required start the game
- Repeat playing until **sufficient** entropy is gathered
  - According to the estimation
  - Detect "bad entropy" moves
- Apply the game's output to a randomness Extractor
  - The guarantees on the quality of the results requires a bound on the min-entropy of the game's output.
- Use the output of the extractor as the input for the refresh state in a robust PRNG



Our Approach

- Using special (expander) graphs for randomness generation from human:
  - Initial Stage-game: Choose a vertex from a graph with large number of vertices. (This provides the initial entropy)
  - Walk stage-game: At vertex V, the player can choose any of the vertices that are adjacent to V. (This provides the sequence of random walks on the expander graph.)
    - Number of random walks depends on the initial entropy and the graph topology

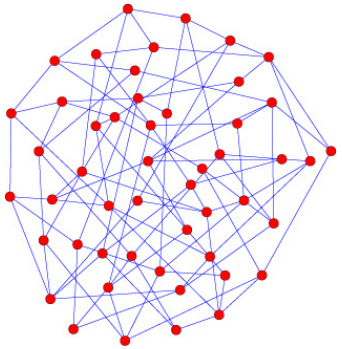




Output size: log 10 = 3.3 bits

Extraction is part of the game

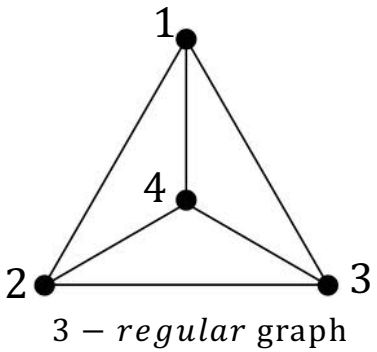
Expander Graphs



- Special types of graphs give an algorithm for extraction

  Expander graphs are *well connected* graphs.

  Expander graphs can be used as an extractor.
- For a d-regular graph the second largest eigenvalue of the adjacency matrix captures the connectivity of the graph.



$3-regular$ graph

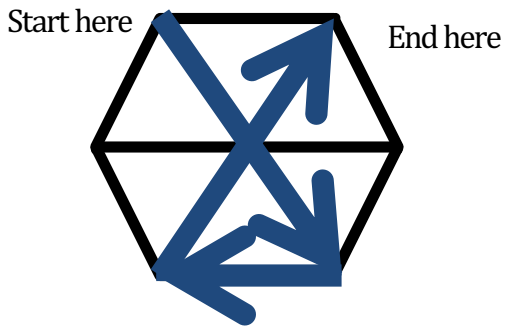
Adjacency matrix

$$\begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}$$

λ: second largest eigenvalue

Expander Graph as an Extractor

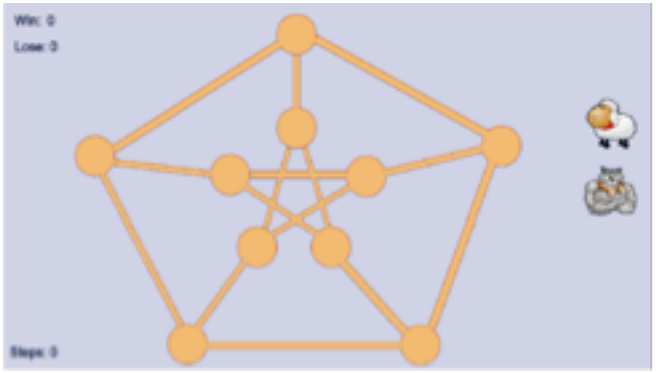
- We can do ***random walk*** on edges of a graph:



- Non-uniform distribution on the vertices and uniform distribution on the leaving edges of each vertex
  - After l random walks the distribution on the vertices becomes ε-close to uniform distribution.
- How close?
  - The second largest eigenvalue of the adjacency matrix determines how close.

Important Properties of the Proposed Approach

- A framework for theoretic design of a TRNG from human gameplay
- Use a simple and well played game in TRNG
- Minimize the dependency of TRNG to other randomness sources
- High randomness generation rate



Randomness Tests

- Two purpouses:
  1. Estimate the min-entropy
     - Does the game provide sufficient entropy for extraction? (min-entropy estimation of initial stage game)
     - How many times the player should play the game?
  2. Evaluating the randomness of outputs
     - Can we accept the output as an output of a good RNG?

The Concept of a Statistical Test [Ma92]

- T is a deterministic algorithm
- $B = \{0,1\}$
- The input is a sample sequence

$$T: B^N \rightarrow \{reject, Accept\}$$

- T is called a statistical test that divides the set B^N of binary N length sequences $s^N = s_1, s_2, \ldots, s_N$ into a set S_T:

$$S^T = \{s^N : T(s^N) = reject\} \subseteq B^N$$

of bad or non-random sequences and the remaining set of good or random sequences.

Min Entropy

- The min entropy for a discrete random variable *X* is the lower bound on its entropy.
- It is also measured in bits. It is often used as worst-case measure of the uncertainty associated with observations .
- For a given discrete probability distribution

  the min-entropy is defined as

$$p_1, p_2, p_3, \ldots, p_m$$

$$H_\infty(X) = -log_2(max(p_1, p_2, p_3, \ldots, p_n))$$

Min-entropy Estimation: NIST Tests

Determining if the Data is IID (independent and identically distributed)

- Hypothesis : the distribution of samples is IID.
- Shuffling Tests
  - A shuffled version of a dataset should be just as likely to have occurred as the original dataset
- Chi-square test
  - A test for independence:

    find dependencies in the existing samples of the dataset
  - A test for goodness of fit:

    checks whether the data subsets produced from the main dataset follow the same distribution.

Min-entropy Tests for IID Sources

- The min-entropy estimation is determined using the most common value estimate :
  - Finds the proportion $\hat{p}$ of the most common value in the input dataset
  - Calculate an upper bound on the probability of the most common value p_u as

$$p_u = \min\left(1, \hat{p} + 2.576\sqrt{\frac{\hat{P}(1-\hat{P})}{L}}\right),$$

    Where L is the size of the database
  - The estimated min-entropy is $-\log_2(p_u)$

Example: If the dataset is S = (0, 1, 1, 2, 0, 1, 2, 2, 0, 1, 0, 1, 1, 0, 2, 2, 1, 0, 2, 1), with L = 20, the most common value is 1, with $\hat{p} = 0.4$. $p_u = 0.4 + 2.576\sqrt{0.012} = 0.6822$. The min-entropy estimate is $-\log_2(0.6822) = 0.5518$

Min-entropy Tests for Non-IID Sources

- Collision test
- Partial collection test
- Frequency test
- Markov test
- Compression test

- The final reported min-entropy is the minimum of all the above tests outputs

Collision Test

- Measure the mean time to the first collision in the dataset
- Continue the process to find at least 1000 collisions
  - Technical point: if the samples are too long (more than 20-bits) more collisions are required or alternatively the dataset should be mapped to a smaller one before running the collision test
- Calculate the mean collision time of the samples
- Determine the probability distribution that has the minimum possible entropy for the calculated mean collision time.
- Return the min-entropy of the resulted distribution as the estimation for the min-entropy of the dataset

Using Min-entropy Tests

- We called our designed game “sheep & wolf”.
  - The game is implemented using HTML 5 technology
  - Can be played by Internet browser and on touchscreen devices
- We asked 9 players to play the game (The initial stage game and walk stage games) for at least 1000 times
  - !Ethics Approval is required if you are asking human to play your game
- The output of the game is one of the vertices of the 10-vertex graph
  - Each vertex is mapped to a 3 or 4 bits binary string ($3 < \log_2 10 < 4$)

Using Min-entropy Tests-Cont

- The IID test showed that are samples are not following an IID distribution
- We used non-IID tests for estimating the min-entropy

User	1	2	3	4	5	6	7	8	9
Total shots	1770	2141	3980	3439	2021	652	1685	905	983
Min-entropy (per bit)	0.45	0.49	0.61	0.65	0.52	0.49	0.47	0.55	0.51

Table 5.1: Min-entropy of users' input in the first stage-game of Wolf and Sheep game [ASNS13, Table 1]

User	1	2	3	4	5	6	7	8	9
Total shots	370	369	1009	1786	560	1071	4821	1190	1065
Min-entropy (per bit)	0.49	0.51	0.64	0.75	0.78	0.72	0.83	0.61	0.79

Table 5.3: Min-entropy of users' inputs in walk stage-games in "Wolf and Sheep" [ASNS13, Table 3]

NIST Statistical Test Suite

- It is a statistical package by NIST (National Institute of Standards and Technology) which is used to test the randomness of binary sequences.
- It consists of 15 tests.
- These tests focus on a variety of different types of non-randomness that could exist in a sequence, and can be used to test the randomness of long binary sequences.

Modified Randomness Tests for a Small Dataset [LS07]

- LinearComplexity
- LempelZiv
- SpectralFourier
- Kolmogorov-Smirnov
- PeriodsInString
- HammingWeight
- HammingCorrelation
- HammingIndependence
- RunTest
- RandomWalk

Randomness Tests Reporting

- For each test a p-value is calculated
- Fix the significance level α (for example 0.05 or 0.001)
- α gives an acceptance margin for p-values
- Accept the randomness of the sequence only if the p-value is not in the margin area determined by α
- It is recommended to report p-values as well

The Results of Tests for "Wolf & Sheep" Game

Statistical Test	p-value
LinearComplexity	0.51
LempelZiv	0.73
SpectralFourier	0.34
Kolmogorov-Smirnov	0.23
PeriodsInString	0.13
HammingWeight	0.55
HammingCorrelation	0.69
HammingIndependence	0.29
RunTest	0.60
RandomWalk	0.67

Table 5.5: Statistical tests for the "Wolf and Sheep" game [ASNS13, Table 2]

Link for Tests

- Downloading tests and examples:
  http://simul.iro.umontreal.ca/testu01/tu01.html
- Running the tests:

  http://www.pcg-random.org/posts/how-to-test-with-testu01.html

References

- [RB92] Amnon Rapoport and David V Budescu. Generation of random series in twoperson strictly competitive games. Journal of Experimental Psychology: General, 121(3):352, 1992.
- [HN09] Ran Halprin and Moni Naor. Games for extracting randomness. In Proceedings of the 5th Symposium on Usable Privacy and Security, page 12. ACM, 2009.
- [BH05] Boaz Barak and Shai Halevi. A model and architecture for pseudorandom generation with applications to /dev/random. In Proceedings of the 12th ACM Conference on Computer and Communications Security, pages 203–212. ACM, 2005.
- [ASNS13] Mohsen Alimomeni, Reihaneh Safavi-Naini, and Setareh Sharifian. A true random generator using human gameplay. In Decision and Game Theory for Security, pages 10–28. Springer, 2013.
- [LS07] Pierre L'Ecuyer and Richard Simard. Testu01: AC library for empirical testing of random number generators. ACM Transactions on Mathematical Software (TOMS), 33(4):22, 2007.
- [Ma92] Maurer, Ueli M. "A universal statistical test for random bit generators." *Journal of cryptology* 5.2 (1992): 89-105.
- [BT12] Bellare, Mihir, and Stefano Tessaro. "Polynomial-time, semantically-secure encryption achieving the secrecy capacity." *arXiv preprint arXiv:1201.3160* (2012).

Thank you

Using Min-entropy Tests-Cont

- Average min-entropy in the initial game:0.52
- Average min-entropy in the walk stage-games:0.68
- Observations:
  - The min-entropy of the human player's choices is more when the alternatives are less.
  - Relatively high min-entropy of players' choices in these stage- games validates our assumption regarding uniformity of choices in walk stage-games.
- Theoretical design parameter:
  - After playing the game for 3 times the output is sufficiently random

Partial Collision Test

- Computes the number of distinct values in the output space
  - Partition the dataset into non-overlapping subsets of size n
  - Computes the number of distinct values in each subset of n samples
- Calculate the mean number of distinct values in each subset of n samples
- Determine the probability distribution that has the minimum possible entropy for the calculated mean number of distinct values.
- Return the min-entropy of the resulted distribution as the estimation for the min-entropy of the dataset

Frequency Test

- Step through the dataset, count the number of times that each sample value is observed
- Record the occurrence of each value and continue the process until the end of the dataset
- Determine the most likely sample value, based on the frequency counts gathered.
- The min-entropy frequency statistic is the negative logarithm of the probability of the most likely sample value

Markov Test & Compression Test

- Markov Test
  - A Markov model is used as a template for describing the dataset
  - The min-entropy is estimated by measuring the dependency between successive outputs of the source
- Compression Test
  - Calculate the compression values for the dataset based on the Maurer Universal Statistic
  - Determine the mean, which is the Maurer statistic
  - Determine the probability distribution that has the minimum possible entropy for the calculated mean values.
  - Return the min-entropy of the resulted distribution as the estimation for the min-entropy of the dataset