A True Random Generator Using Human Gameplay

Mohsen Alimomeni, Reihaneh Safavi-Naini, and Setareh Sharifian

University of Calgary, Department of Computer Science, Canada
{malimome,rei,ssharifi}@ucalgary.ca

Abstract. True Randomness Generators (TRG) use the output of an entropy source to generate a sequence of symbols that is sufficiently close to a uniformly random sequence, and so can be securely used in applications that require unpredictability of each symbol. A TRG algorithm generally consists of (i) an entropy source and (ii) an extractor algorithm that uses a random *seed* to extract the randomness of the entropy source. We propose a TRG that uses the user input in a game played between the user and the computer both as the output of an entropy source, and the random seed required for the extractor. An important property of this TRG is that the (randomness) quality of its output can be flexibly adjusted. We describe the theoretical foundation of the approach and design and implement a game that instantiates the approach. We give the results of our experiments with users playing the game, and analysis of the resulting output strings. Our results support effectiveness of the approach in generating high quality randomness. We discuss our results and propose directions for future work.

1 Introduction

Many security systems and in particular crypto-algorithms use random values as an input to the system. Cryptosystems need randomness for purposes such as key generation, data padding or challenge in challenge-response protocols. In nearly all cases unpredictability of random values is critical to the security of the whole systems. Generating true randomness however is not an easy task and needs a physical entropy source. Operating systems such as Windows and Linux use special sub-systems that combine randomness from different parts of the hardware and software system to collect entropy [Mic, GPR06]. Poor choices of randomness has lead to breakdown of numerous security systems. Important examples of such failures are, attack on Netscape implementation of the SSL protocol [GD] and weakness of entropy collection in Linux and Windows Pseudo-Random Generator [GPR06, DGP09]. A recent example of the need for careful treatment of randomness in cryptographic systems was highlighted in [HDWH12, LHA+12] where the same public and private keys were generated by key generation modules and this was partly attributed to randomness that was poorly generated in Linux kernel randomness generation subsystem.

An *entropy source* uses physical processes such as noise in electronic circuits, or software processes that are"unpredictable", to output a sequence over an

S.K. Das, C. Nita-Rotaru, and M. Kantarcioglu (Eds.): GameSec 2013, LNCS 8252, pp. 10–28, 2013.

alphabet that is highly "unpredictable", where unpredictability is measured by *min-entropy* (See Definition 3). Although the output of an entropy sequence can have high level of randomness, but the underlying distribution may be far from uniform. To make the output of an entropy source to follow a uniform distribution, a post processing step is usually used. *Randomness extractors* are deterministic or probabilistic functions that transform the output of an entropy source to uniform distribution using a mapping from n to m bits (usually $m \leq n$), extracting the entropy of the source.

To guarantee randomness of their output, randomness extractors need guarantee on the randomness property of (e.g. the min-entropy) their input entropy source. Extractors that can extract randomness from sources that satisfy a lower bound on their min-entropy, are probabilistic [Sha11]. A probabilistic extractor has two inputs: an entropy source together with a second input that is called *seed*. Good probabilistic extractors use a short seed (logarithmic in the input size) to extract all the randomness (close to the min-entropy) of the input entropy source. A *True Randomness Generator (TRG)* thus consists of two modules: (i) an entropy source that generates a sequence of symbols with a lower bound on its min-entropy, followed by, (ii) a randomness extractor. In practice one needs to estimate the min-entropy of the entropy source to be able to choose appropriate parameters for the extractor. The distribution of the entropy source and its min-entropy may fluctuate over time and so a TRG needs to use an extractor that provides sufficient tolerance for these fluctuations.

Human as Entropy Source: In [HN09], Halprin et al. proposed an innovative approach to construct an entropy source using human game play. Their work built on the results in experimental psychology. It is known that humans, if asked to choose numbers randomly, will do a poor job and their choices will be biased. Wagenaar [Wag72] used experiments in which participants were asked to produce random sequences and noted that in all experiments human choices deviated from uniform distribution. In [RB92], Rapport et al. through a series of experiments showed that if human plays a competitive zero-sum game with uniform choices as the best strategy, their choices will be close to uniform. In their experiment they used matching pennies game in which each player makes a choice between head or tail using an unbiased coin, and the first player wins if both players choose the same side and the second, if they choose different side. In this game the optimal strategy of users is random selection between head and tail. Their result showed that users almost followed uniform random strategy confirming that human can be a good source of entropy if they are engaged in a strategic game and entropy generation is an indirect result of their actions.

Human Game Play for Generating Randomness. Halprin et al. used these studies to propose an entropy source using human game play against a computer. In their work human plays a zero-sum game with uniform optimal strategy against the computer. The game is an extended matching pennies game (user has more than two choices) and is played many times. The sequence resulting from human choices is considered as the output of an entropy source, and is used as the input to a randomness extractor. The result is a TRG with an output that is a random

sequence "close" to uniform. In addition to the human input sequence, the TRG uses a second source of perfect randomness to provide seed for the randomness extractor.

In this paper we propose an integrated approach where the game play between a human and the computer is used to implement the two phases of a TRG including randomness source and randomness extraction phase. That is the user's input provides the required randomness for the entropy source and the extractor both.

1.1 Our Contribution

We propose a TRG that uses human game play against a computer, as the *only* source of randomness. The game consists of a sequence of sub-games. Each sub-game is a simple two player zero-sum game between the user and the computer which can be seen as an extended matching pennies game. In each sub-game the human makes a choice among a number of alternatives, each with the same probability of winning, resulting in the user's best strategy to be random selection among their possible choices. The first game corresponds to the entropy generation step in TRG and subsequent sub-games correspond to steps of an extractor algorithm.

The TRG algorithm is based on a seeded extractor that is constructed using an *expander graph*. Expander graphs are highly connected d-regular graphs where each vertex is connected to d neighbours. This notion is captured by a measure called *spectral expansion*. It has been proved that random walks on these graphs can be used to extract randomness [AB09]. Assuming an initial probability distribution $\mathbf{p}$ on the set of vertices of the graph, it is proved [AB09] that by taking a random walk of ℓ steps from any vertex in the graph that is chosen according to $\mathbf{p}$, one ends up at a vertex that represents a distribution over the vertices that is ϵ-close to the uniform distribution. In other words starting from any distribution, each step of the random walk results in a new distribution over the vertices that is closer to uniform and so by taking sufficiently long walk, one can obtain a distribution that is ϵ-close to the uniform distribution.

We use human input to provide the required randomness in the framework above: that is for the initial distribution $\mathbf{p}$ as well as the randomness for each step of the walk. To obtain randomness from human, a sequence of games is presented to the user and the human input in the game is used in the TRG algorithm. In the first sub-game, the graph is presented to the user who will be asked to randomly choose a vertex. This choice represents a source symbol that is generated according to some unknown distribution $\mathbf{p}$; that is human choice is effectively a symbol of an entropy source. Human choices however, although have some entropy but cannot be assumed to be uniformly distributed. A subsequent random walk of length ℓ over the graph will be used to obtain an output symbol for TRG with close to uniform randomness guarantee.

To use human input for the generation of the random walk, on each vertex of the graph the user is presented with a simple game which effectively requires them to choose among the set of the neighbouring vertices. The game is

zero-sum with uniform optimal strategy and so the human input would correspond to uniform selection, and consequently one *random step* on the graph. For a given ϵ and an estimate for the min-entropy of the initial vertex selection, one can determine the number of required random steps so that the output of the TRG has the required randomness.

In the above we effectively assume *human input in a uniform optimal strategy zero-sum game is close to uniform.* This assumption is reasonable when human is presented with a few choices, (based on the experiments in [RB92]). In practice however the human input will be close to uniform and so the proposed extraction process can be seen as approximating the random walk by a high min-entropy walk. Obtaining theoretical results on the quality of output in an expander graph extractor when the random walk is replaced with a walk with high min-entropy is an interesting theoretical question. We however demonstrate feasibility of this approach experimentally.

We designed and implemented a TRG that is based on a game on a 3-regular expander graph with 10 vertices. The game consists of a sequence of sub-games. A number of screen shots of the game are shown in Figure 1. In each sub-game the human and the computer make a choice from a set of vertices. If the choices coincide, computer wins and if they do not coincide, human wins. In our implementation the human first make a choice and then the computer's choice is shown. In the first sub-game user makes a choice among the 10 vertices of the graph, and in all subsequent sub-games, among the 3 neighbours of the current vertex. We perform a number of experiments to validate our assumptions.

Experiments. We implemented the above game and experimented with nine human users playing the games. We measured min-entropy of human input in the first sub-game, that is when human is an entropy source, and also subsequent sub-games when human input is used to emulate the random walk. For the former, that is to estimate the initial distribution $\mathbf{p}$, we designed a one round game which requires the user to choose a vertex of the graph and they win if their choice is not predictable by the computer. We used NIST [BK12] tests for estimating the min-entropy of both distributions. The details of experiments are given in Section 4. Our results once again shows that humans, once engaged in a two-party zero sum game with uniform optimal strategy, are good sources of randomness. The min-entropy of human choices in the first sub-game is 2.1 bits per symbol (10 symbol) in average and in the subsequent sub-games is 1.38 bits per symbol (3 symbols) in average. These compared to the maximum available entropy of the source on corresponding number of symbols, i.e. $\log_2 10 = 3.32$ and $\log_2 3 = 1.58$, indicate that indeed the human choices are close to uniformly random and the final output of TRG is expected to be close to random.

Applications. TRGs are an essential component of computing systems. User based TRG add an extra level of assurance about the randomness source: users know that their input has been used to generate randomness. An example application of this approach is generating good random keys using user's input. Asking a user to generate a 64 bit key that is random will certainly result in a biased string. Using the approach presented in this paper, the user can select an

element of the space (say a password with 13 characters) randomly. The user choice will be used as the initial entropy source, and then subsequent games will ensure that the final 64 bits is close to uniform. Assuming a 3-regular expander graph with 10 vertices, one needs 3 steps in the expander graph to reach a 1/4-close to uniform. Section 4.1 further discusses how longer sequences can be generated.

1.2 Related Work

The idea of using a game to motivate human to generate unbiased randomness was proposed in [HN09]. Authors used the experimental results in psychology [Wag72, RB92] along with game theoretic approach to show humans playing matching pennies game generate a sequence which is close to uniform. Halprin et al. argued that this game when played between a computer and a human can be used as an entropy source. To increase the amount of randomness generated by human with each choice, Halprin et al. used an extension of this game that uses n choices to the player: the user is presented by an $n \times n$ matrix displayed on the computer screen and is asked to choose a matrix location. The user wins if their choice is the same as the square chosen by the computer. They noted that the visual representation of the game resulted in the user input to be biased as users avoided corner points and limiting squares. The sequence generated by human was used as the input to a seeded extractor (Definition5) to generate a sequence that is ϵ-close to uniform (Definition1). They provided visual representations of human choices that indicates a good spread of points. However statical and min-entropy evaluation of the system is restricted to using statistical tests on the output of the seeded extractor.

Extraction can use a general seeded extractor that will guarantee randomness of the output for *any distribution* with min-entropy (Definition 3) k, or an approach proposed in [BST03] in which the set of possible input sources is limited to a set of 2^t possible ones all with min-entropy k. The former approach requires a fresh random seed for each extraction but has the advantage that the input source can have any distribution with the required min-entropy. This latter approach however requires the input sequence to be one of the set of 2^t possible sources, but has the advantage that one can choose a function from a class of available extractors and hard code that in the systems. This means in practice no randomness is required. However no guarantee can be given about the output if the input sequence is not one of the 2^t that has been used for the design of the system and this property cannot be tested for an input sequence. Halprin et al. used the latter approach, using a t-universal hash function as the extractor. The randomness guarantee of the final result requires the assumption that the human input is one of the 2^t sources. In practice, t can not be arbitrarily large and must be small to guarantee a minimum output rate for randomness. This can pose a security risk that the actual distribution is not one of the 2^t distributions. Halprin et al. did not perform quantitative analysis of user sequences and used visual representation of the human choices to conclude the choices were random.

We note that simpler extractors such as the Von Neumann extractor [vN51] put strong requirements on their input sequence. For example Von Neumann extractor requires the input string to be a Bernoulli sequence with parameter p which is not satisfied by the sequence of human choices where successive choices may be dependent with unknown and changing distribution. The expander graph extractor works for all distributions whose min-entropy is lower bounded by a given value, and does not put extra requirements on the input sequence.

Using human input as entropy source has also been used in a different form. In computer systems, users' usage of input devices such as mouse and keyboard can be used for background entropy collection. This process is used for example in Linux based systems [GPR06]. [ZLwW+09] uses mouse movement and applies hash functions.

Paper Organization. Section 2 provides the background. Section 3 outlines our approach and Section 4 gives the results of our experiments. Section 5 provides concluding remarks.

2 Preliminaries

We will use the following notations. Random variables are denoted by capital letters, such as X. A random variable X is defined over a set $\mathcal{X}$ with a probability distribution $\Pr_X$, meaning that X takes the value $x \in \mathcal{X}$ with probability $\Pr_X(x) = \Pr[X = x]$. Uniform distribution over a set $\mathcal{X}$ is denoted by $U_{\mathcal{X}}$ or U_n if $\mathcal{X} = \{0,1\}^n$. The logarithms will be in base 2 throughout the paper.

Definition 1. *Consider two random variables X and Y taking values in $\mathcal{X}$.* Statistical distance *of the two variables is given by,*

$$\Delta(X;Y) = \frac{1}{2} \sum_{x \in \mathcal{X}} \left| \Pr_X(x) - \Pr_Y(x) \right|$$

We say that X and Y are ϵ-close *if $\Delta(X;Y) \leq \epsilon$.*

A *source* on $\{0,1\}^n$ is a random variable X that takes values in $\{0,1\}^n$ with a given distribution $\Pr_X$.

Definition 2. *Let C be a class of sources on $\{0,1\}^n$. A* deterministic ϵ-extractor *for C is a function $\mathsf{ext} : \{0,1\}^n \to \{0,1\}^m$ such that for every $X \in C$, $\mathsf{ext}(X)$ is "ϵ-close" to U_m.*

Deterministic extractors exist for a limited classes of sources [Sha11]. A small random seed is used in extractors to allow for more general classes of sources to be used as input to the extractor. The more general classes of sources are the sources with a guaranteed amount of randomness. To quantify the amount of randomness in a random variable min-entropy is used.

Definition 3. *The* min entropy *of a random variable X is : $H_\infty(X) = min_x \left\{ \log \frac{1}{Pr_X(x)} \right\}$.*

Definition 4. *A random variable X is a $k-$source if $H_\infty(X) \geq k$, i.e., if $Pr_X(x) \leq 2^{-k}$. The min-entropy rate is denoted by δ and is defined as $k = \delta n$, where n is the number of source output bits.*

Using probabilistic method, it is proved [Sha11] that there exists probabilistic extractors that can extract at least k bits of randomness from a k-source.

Definition 5. *A function* $\mathsf{ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a* seeded (k, ϵ)-extractor *if for every k-source X on $\{0,1\}^n$, Ext(X,U_d) is ϵ-close to U_m, where U_m is the random variable associated with the uniform distribution on m bits.*

A relevant result on seeded extractors is the following.

Theorem 1. *[Sha11] For every $n \in \mathbb{N}$, $k \in [0, n]$ and $\epsilon > 0$, there exists a (k, ϵ)-extractor* $\mathsf{ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with $m = k + d - 2\log(\frac{1}{\epsilon}) + O(1)$.*

2.1 Expander Graphs

Expander graphs are well connected graphs in the sense that to make the graph disconnected one needs to remove relatively large number of edges. Connectivity of a graph can be quantified using measures such as the minimum number of neighbouring vertices for all sub-graphs or minimum number of edges that leave all sub-graphs (minimums are taken over all subgraphs of certain size) [HLW06]. For a d-regular graph the second eigenvalues of the adjacency matrix captures the connectivity of the graph. This measure is referred to as *spectral expansion.*

Normalized adjacency matrix of a d-regular graph with n vertices is an $n \times n$ binary matrix with $A_{i,j} = \frac{1}{d}$ if vertex i and j are connected by an edge, and zero otherwise.

Expander Graphs as Extractors. Given a graph and a starting vertex X, one can make a random walk of length ℓ, by randomly choosing one of the neighbours of X, say X_1, move to X_1, then randomly choose one of the neighbours of X_1, say X_2, and repeat this ℓ times.

Let G denote an expander graph with normalized adjacency matrix A, and let $\mathbf{p}$ denote an initial distribution on the vertices of G. After one random step from each vertex, the distribution on the vertices is given by $A\mathbf{p}$ and becomes closer to uniform. That is, the statistical distance between the distribution on the graph vertices and the uniform distribution reduces. Continuing the random walk on the graph for ℓ steps, the distribution on the vertices becomes $A^l\mathbf{p}$ and gets closer to the uniform distribution. The rate of convergence to uniform distribution for d-regular expander graphs is determined by the second eigenvalue of the normalized adjacency matrix of the graph which is denoted by λ from now on.

Lemma 1. *[AB09, lemma 21.3] Let G be a regular graph over n vertices and $\boldsymbol{p}$ be an initial distribution over G's vertices. Then we have:*

$$\left\|A^l\boldsymbol{p} - U_n\right\|_2 \leq \lambda^l.$$

where $\|.\|_2$ *is the l2 norm defined as* $\|X\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$*, considering* X *to be the vector* $(x_1, x_2, \ldots, x_n)$*. Note that* U_n *and* $A^l\boldsymbol{p}$ *are considered as vectors in above.*

The random walk on an expander graph explained above gives the following extractor construction.

Lemma 2. *[AB09, lemma 21.27] Let* $\epsilon > 0$*. For every* n *and* $k \leq n$*, there exists an explicit* (k, ϵ)*-extractor* $\mathsf{ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^n$ *where* $t = O(n - k + \log 1/\epsilon)$*.*

The above lemma assumes an expander graph with $\lambda = 1/2$, but in general for an arbitrary λ and min-entropy k, we can derive the following theorem from the above lemmas:

Theorem 2. *Let* U_n *be the uniform distribution and* X *be a* k*-source with probability distribution* $\boldsymbol{p}$ *over* $\{0,1\}^n$*. Let* G *be a* d*-regular expander graph over* 2^n *vertices with normalized adjacency matrix* A*. For a random-walk of length* l *over the graph starting from a vertex selected according to distribution* $\boldsymbol{p}$*, we have*

$$\Delta(A^l\boldsymbol{p}; U_n) \leq \frac{1}{2}\lambda^l\sqrt{n}(2^{-k/2} + 2^{-n/2})$$

The proof of the above theorem follows from the proof of Lemmas 1 and 2

$$\Delta(A^l\mathbf{p}; U_n) = \frac{1}{2}\sum_{a\in\{0,1\}^n} \left|\Pr[A^l\mathbf{p} = a] - \Pr[U_n = a]\right| \tag{1}$$

$$\leq \frac{1}{2}\sqrt{n}\left\|A^l\mathbf{p} - U_n\right\|_2 \tag{2}$$

$$\leq \frac{1}{2}\sqrt{n}\lambda^l\left\|\mathbf{p} - U_n\right\|_2 \tag{3}$$

$$= \frac{1}{2}\sqrt{n}\lambda^l(2^{-k/2} + 2^{-n/2}) \tag{4}$$

where equation (1) follows from the definition of statistical distance, equation (2) is followed from the relation between l2 and l1 norms, i.e. $|V| \leq \sqrt{n}\,\|V\|_2$, equation (3) comes from the proof of lemmas 1 and 2, and equation (4) follows from linear algebra facts ($\|\mathbf{p} - U_n\|_2 \leq \sqrt{\|\mathbf{p}\|_2^2 + \|U_n\|_2^2}$) and that min-entropy of $\mathbf{p}$ is k, which gives $\|\mathbf{p}\|_2^2 \leq 2^{-k}$.

□

For an expander graph G, given ϵ and k as the min-entropy of the initial distribution on vertices, we can compute the number of required steps of the random-walk on the expander graph so that the distribution on the graph vertices becomes ϵ-close to uniform distribution. Note that min-entropy of the initial vertex distribution results in closeness to uniform distribution, but the random walk will amplify this closeness.

Let $\lambda = 2^{-\alpha}$ and $\epsilon = 2^{-\beta}$. To be ϵ-close to uniform, we must have $\frac{1}{2}\sqrt{n}\lambda^l(2^{-k/2} + 2^{-n/2}) \leq \epsilon$. This gives us the following lower bound on l:

$$l \geq \frac{1}{\alpha}[\beta + \log(\sqrt{n}) + \log(2^{-k/2} + 2^{-n/2})] \tag{5}$$

The above bound requires the value λ for the graph. Equation 5 shows that for a given ϵ and min-entropy, smaller λ correspond to shorter random walk. So one needs to find graphs with smaller λ.
The following theorem shows that regular graphs have small λ.

Theorem 3. *[AB09, section 21.2.1] For a constant $d \in \mathbb{N}$, any d-regular, N-vertex graph G satisfies $\lambda \geq 2\sqrt{d-1}/d(1-o(1))$ where the $o(1)$ term vanishes as $N \to \infty$.*

Ramanujan graphs are d-regular graphs that achieve $\lambda \geq 2\sqrt{d-1}/d$ and so are excellent as spectral expanders. For a fixed d and large N, the d-regular N-vertex Ramanujan graph minimizes the λ. There are several explicit constructions of Ramanujan graphs. Here we explain one of the simpler constructions.

2.2 A Simple Explicit Construction for Expander Graphs

There are explicit constructions of expander graphs that can be efficiently generated. That is vertices are indexed by $i \in I$ and there is an algorithm that for any i, generates the index of its neighbours. For example the *p-cycle with inverse chords* construction gives us a 3-regular expander graph with p vertices, p is prime, in which a vertex X is labelled by $x \in \{0, p-1\}$ and the neighbour vertices have indexes $x-1$, $x+1$ and x^{-1}. Here all arithmetic are mod p and 0^{-1} is defined to be 0. The spectral expansion of this graph is very close to the above-mentioned bound. The construction is due to Lubotzky-Phillips-Sarnak [LPS86] and the proof that the construction is a Ramanujan graph, uses deep mathematical results. The λ for this graph is upper bounded by 0.94. Other explicit constructions of expander graphs use graph product techniques such as Zig-Zag product and replacement product [RVW00].

2.3 Game Theoretic Definitions

A game consists of a set of *players*, each with a set of available *actions* and a specification of payoffs for each pair of actions. An *action profile* is a tuple of all players' actions. An assumption in game theory is that players play rationally; that is they aim to maximize their payoff, given some belief about the other players' actions. Thus, each player has *preferences* about the action profile. For describing players' preferences we use *utility function*. An strategic game is defined as follow.

Definition 6. *A strategic game with ordinal preferences consists of,*

- *A finite set of players,* $N = \{1, ..., N\}$*;*
- *A set of actions for player* i *denoted by* A_i*;*
  Action profile*:* $a = (a_1, ..., a_n) \in A = A_1 \times ... \times A_n$*;*
- *Preferences over the set of action profiles based on the* Utility Function *that captures payoffs.*
  Utility function *for player* i*:* $u_i : A \to R$ *, which* R *is the set of real numbers. We say that player* i *prefers* a *to* b *iff* $u_i(a) > u_i(b)$

A *pure strategy* specifies the actions of a player in all possible situations that he will be in.

Mixed strategy of a player in a strategic game means more than one action is played with a positive probability by the player. The set of actions with non-zero probability form the *support of mixed strategy.* Mixed strategy is specified by a set of probability distributions. The following theorem shows the importance of mixed strategies [Osb04].

Theorem 4. *Every strategic game in which each player has finitely many actions has a mixed strategy Nash equilibrium.*

A finite two-player strategic game can be represented by a table. In such a representation "Row" player is player 1 and "column" player is player 2. That is, rows correspond to actions $a_1 \in A_1$ and columns correspond to actions $a_2 \in A_2$. Each cell in the table includes a pair of payoffs, starting with the row player followed by the column player. For a two-party game, each player action is a pure strategy and a mixed strategy is a probability distribution on the set of actions available to them.

Halprin et al. used an extended form of *"matching pennies"* using a two dimensional array defining the choices of the players. In basic matching pennies game with Head and Tail as possible actions for each player the payoff table is as follow:

-	Head	Tail
Head	(1,-1)	(-1,1)
Tail	(-1,1)	(1,-1)

Matching pennies game is a zero-sum game and for any action profile a we have $u_1(a) + u_2(a) = 0$. From the table it can be seen that there is no pure strategy Nash equilibrium for matching pennies. It is easy however to show that *the best strategy for both players in the game is uniform distribution on the set of possible actions.* This is easy to see intuitively and also show formally using a distribution p and $1 - p$ to denote the probability that the first player chooses Head or Tail, respectively, and then requiring p to be chosen such that the player 2 remains completely indifference about the choice of player 1. That is, $u_2(Head) = u_2(Tail)$ which gives $-p + 1(1 - p) = p - (1 - p)$ and $p = 1/2$. A similar argument shows that the best strategy of player 2 is uniform also.