Introduction to Provably Fair Gaming Algorithms (5th Draft)

Kristóf Poduszló

July 13, 2017

Abstract

As of today, the majority of the online gaming industry utilizes black box algorithms, forcing users to trust a third-party service for generating unbiased random data. Lately, a new paradigm has started to spread through the industry, paving the way towards transparent and verifiable algorithms being a standard in digital games.

1 Introduction

Provably fair algorithms bring a new era of opportunities to the online gaming industry. Unlike black box algorithms widely used amongst luck-based games involving stake [7, 1], their fair counterparts are verifiable by anyone, including every participant of a particular game [4].

The algorithms described in this document represent the culmination of research about commitment schemes [3, 13], including, but not limited to, flipping a coin [2] [10, pp. 243–245] or playing a fair game of poker [14] [10, pp. 246–250] over the wire.

2 Concepts

2.1 Verifiability of pseudorandom outputs

Pseudorandom number generators provide a sequence of seemingly random outputs initialized by a seed. The presence of an initialization parameter provides the opportunity to use it as a key for verification of results.

A seed used in the algorithms covered by this document should consist of two main parts:

- *hostSeed*: Shall be kept in secret until the end of a particular game. Similar to a private key in asymmetrical cryptography.
- *publicSeed*: Players should only generate or contribute to it (with equal amounts of influence) after a commitment (e.g. cryptographic hash) of *hostSeed* has been broadcast to every participant of a particular game.

Remark 2.1.1. Broadcasting a commitment of *hostSeed* amongst players not only protects *hostSeed* from being revealed early, but serves as a verification of integrity, proving that during a game, *hostSeed* could not have been tampered without notice.

Using a mix of the entire *hostSeed* and *publicSeed* (e.g. by concatenating them) as an initialization parameter for randomization, every participant may have an influence on the outcome of results, with a negligible chance of manipulation[1] in favor of any entity.

Remark 2.1.2. In a peer-to-peer network, every player is also a host, resulting in the presence of multiple *hostSeeds* and *publicSeeds* possibly paired to a *privateKey* and a corresponding *publicKey* for every participant. The *publicKey* of each player may also be used as commitments.

2.2 Initialization cycle

A random *hostSeed* must be generated to initiate a new game.

- Whether only a single player is betting against an operator, a *hostSeed* may be generated by the host using any source of entropy (preferably a true random number generator).

- If multiple players are betting against an operator, a provably fair seeding event may be used to generate *hostSeed*.

  Definition 2.2.1. A provably fair seeding event [9] makes it possible to generate *publicSeed* using a trustless randomization service (e.g. the hash of a specific upcoming block in the blockchain of a cryptocurrency), disallowing participants to have a direct influence on in-game randomization.

  Remark 2.2.2. When multiple players participate in a game, *hostSeed* shall not be generated by a single entity because that would allow a coalition to gain advantage over honest players by whispering *hostSeed* early to a selected group of participants.

- The problem of multiple players betting against each other may be solved by a mental poker protocol [14], which is beyond the scope of this document.

Once *hostSeed* is revealed (optimally, at the end of a particular game), outputs generated by the algorithm become reproducible, proving that random results could not have been manipulated in favor of any entity.

2.3 Definition and properties of a provably fair algorithm

Definition 2.3.1. An algorithm behind a game is provably fair if and only if every participant has the same amount of influence on in-game randomization in a verifiable manner.

Remark 2.3.2. Participants include players and, if present, trustless seeding services.

[1]Given a commitment scheme which is computationally infeasible to break (e.g. based on a secure hash algorithm).

Proposition 2.3.3. *Necessary criteria of a provably fair algorithm*

(i) Determinism (always produce the same output given a particular input).

(ii) A combination of the entire hostSeed and publicSeed is used for generating outputs (e.g. a keyed hash function[2] like HMAC using hostSeed as key and publicSeed as message).

(iii) The integrity of hostSeed shall be verifiable by players (e.g. by publishing its hash prior to the start of every particular game).

(iv) The algorithm must be public for every participant of the game.

3 Algorithms

In this section, numerous generic fair algorithms will be proposed for games which are influenced by randomization, including, but not limited to, rolling a dice and shuffling a deck of cards.

3.1 Generating a single random output

The output generation function should be hard to invert [8, pp. 30–35] in order to protect outputs from being predictable before *hostSeed* is revealed. While any entity in possession of *hostSeed* may predict the outputs of a provably fair algorithm, there should be no concern about fairness until every player has the same amount of information (preferably nothing) about *hostSeed* during a game.

3.2 Generating a sequence of random outputs

When multiple players participate in a game with numerous betting rounds following output generation, a new *publicSeed*, influenced by every player or a trustless service, shall be used before each round in which bets may be placed.

In order to generate multiple outputs using a single set of seeds, a cryptographic *nonce* [12, pp. 397–398] should be utilized. A *nonce* used in provably fair algorithms shall be unique and predictable (e.g. it may represent the number of consecutive bets using the same *hostSeed*, assuming the probability of a *hostSeed* collision is negligible).

A *nonce* may only be used once for a particular seed set, and shall be appended to the initial *publicSeed*, producing a unique output for consecutive bets made using the same seeds.

Theoretically, an arbitrarily large output sequence can be generated using a bijective mathematical function $\mathrm{f} : \mathbb{N} \to \mathbb{R}$ (e.g. $f(x) = x$), agreed upon the initialization cycle of a game (until a commitment about *hostSeed* is made), as a *nonce* sequence provider.

Multiple parameters may be used to construct a *nonce* if necessary (e.g. when shuffling a deck of cards in a turn-based game, *nonce* should consist of both the round identifier and the shuffle state).

[2] Unforgeability protects outputs from being predictable before *hostSeed* is revealed. Using unkeyed hash functions or pseudorandom number generators is strongly discouraged.

4 Examples

Remark. Ensuring uniform distribution of random outputs is not in the scope of this document.

4.1 Generating a single random integer

The following functions generate a random integer based on a variant of the practically non-invertible HMAC (hash-based message authentication code) [11] function using *hostSeed* as key and *publicSeed* as message.

Algorithm 4.1.1 Generating a random integer in the range $[min, max[$

function RandomInt($hostSeed, publicSeed, min, max$)
  return min + ($HMAC(hostSeed, publicSeed)$ mod $(max - min)$)
end function

Algorithm 4.1.2 Rolling a dice

function RollDice($hostSeed, publicSeed$)
  return $RandomInt(hostSeed, publicSeed, 1, 6)$
end function

4.2 Generating a sequence of random integers

If multiple random outputs are required throughout a particular game, a *nonce* may be used to produce a sequence of random results. A *nonce* should be concatenated to *publicSeed* using a separator (e.g. " : ").

Algorithm 4.2.1 Shuffling an array (Fisher–Yates shuffle [6, 5])

function Shuffle($hostSeed, publicSeed, array$)
  $n \leftarrow array.length$
  for $i \leftarrow 0, n-2$ **do**
    $j \leftarrow RandomInt(hostSeed, publicSeed + \text{" : "} + i, i, n)$
    $Swap(array[i], array[j])$
  end for
end function

References

[1] *Are Online Casinos Rigged? We Explore the Facts.* URL: https://casino.org/rigged-casino-guide (visited on 06/19/2017).

[2] Manuel Blum. "Coin Flipping by Telephone - A Protocol for Solving Impossible Problems". In: *SIGACT News* 15.1 (Winter-Spring Jan. 1983), pp. 23–27. ISSN: 0163-5700. DOI: 10.1145/1008908.1008911.

[3] Gilles Brassard, David Chaum, and Claude Crépeau. "Minimum Disclosure Proofs of Knowledge". In: *Journal of Computer and System Sciences* 37.2 (Oct. 1988), pp. 156–189. ISSN: 0022-0000. DOI: 10.1016/0022-0000(88)90005-0.

[4] Vitalik Buterin. *The Bitcoin Gambling Diaspora.* Aug. 3, 2013. URL: https://bitcoinmagazine.com/articles/the-bitcoin-gambling-diaspora-1375548799 (visited on 06/18/2017).

[5] Richard Durstenfeld. "Algorithm 235: Random permutation". In: *Communications of the ACM* 7.7 (July 1964), p. 420. ISSN: 0001-0782. DOI: 10.1145/364520.364540.

[6] Ronald A. Fisher and Frank Yates. *Statistical tables for biological, agricultural and medical research.* 3rd ed. 1948, pp. 26–27.

[7] Sally M. Gainsbury, Jonathan Parke, and Niko Suhonen. "Consumer attitudes towards internet gambling: perceptions of responsible gambling policies, consumer protection, and regulation of online gambling sites". In: *Computers in Human Behavior* 29.1 (Jan. 2013), pp. 235–245. DOI: 10.1016/j.chb.2012.08.010.

[8] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools.* Cambridge University Press, Jan. 18, 2007. ISBN: 9780521035361. DOI: 10.1017/CBO9780511546891.

[9] Ryan Havar. *Bustabit.com Provably Fair Seeding Event.* Jan. 12, 2015. URL: https://bitcointalk.org/index.php?topic=922898 (visited on 07/02/2017).

[10] James S. Kraft and Lawrence C. Washington. *An Introduction to Number Theory with Cryptography.* CRC Press, Aug. 1, 2013. ISBN: 9781482214420. URL: https://books.google.com/books?id=mYLSBQAAQBAJ.

[11] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. *HMAC: Keyed-Hashing for Message Authentication.* RFC 2104. RFC Editor, Feb. 1997. DOI: 10.17487/RFC2104.

[12] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* 1st ed. Discrete Mathematics and Its Applications. CRC Press, Oct. 16, 1996. ISBN: 9780849385230.

[13] Moni Naor. "Bit Commitment Using Pseudorandomness". In: *Journal of Cryptology* 4.2 (Jan. 1991), pp. 151–158. ISSN: 0933-2790. DOI: 10.1007/BF00196774.

[14] Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman. "Mental Poker". In: *The Mathematical Gardner.* Ed. by David A. Klarner. 1981, pp. 37–43. ISBN: 9781468466867. DOI: 10.1007/978-1-4684-6686-7_5.