Aiza: A Public, Lightweight, and Difficult to Manipulate Random Number Generator

Luke Champine
Nebulous Labs

May 3, 2014

Abstract

Aiza is a public system used for generating public, lightweight, and difficult to manipulate public random numbers. We contrive a 'Byzantine Generals Election' problem requiring the production of difficult to manipulate random numbers. Then we demonstrate that the Bitcoin hashcash algorithm can be used to accomplish such a goal. Finally, we make some modifications to the Bitcoin protocol to make a lightweight alternative for random number generation, which we call Aiza.

1 Introduction

Decentralized networks need to be able to cope with malicious behavior by a set of nodes. Decentralized networks can require the generation of random numbers. Consider the problem of the Random Byzantine Generals Election. There are a set of Byzantine Generals trying to randomly elect a commander. The election results should insure that

1. The commander is elected at random.
2. The random number cannot be easily manipulated by traitors.
3. Manipulating the random number is expensive and uncertain.

Existing solutions to this problem require both a high percentage of loyal generals and produce results that can be manipulated within some epsilon. These solutions typically have each general generating a random outcome and then somehow combining the outcomes. A different approach is needed.

The Bitcoin proof-of-work model provides this different approach. With Bitcoin, an initial value plus an arbitrary string is hashed repeatedly, until a hash matching certain properties is found. This is the hashcash algorithm. When a hash of certain (unlikely) probabilities is discovered, a 'block' is created. By taking the hash of the solution to the hashcash problem, we can generate a secure random seed. If traitor wanted to manipulate the entropy, the most that they could do is be the first to find a hash, reject or accept in on the grounds of being favorable or unfavorable, and then try to find another solution that presents a more favorable seed. All the while, the loyal generals are also attempting to solve the hashcash, so the attacker is likely to be interrupted when trying for a re-roll. Furthermore, the reward associated with bitcoin is currently 25 bitcoins, currently valued at \$11,000. A traitor would need to sacrifice the block reward in order to re-roll. Not only is it difficult for the traitor to re-roll the entropy, it results in significant financial sacrifice.

Let us assume that the loyal generals and the traitors have equal hashing resources. The goal of the traitors is to produce a random number that results in a traitorous general being elected. With no manipulation, a traitorous general will be elected 1/2 of the time. Because each block requires as part of the input the header of the previous block, the traitors cannot 'save up' hashing power. Every time a traitor finds a block, they can announce the block if they determine the hash to be favorable, and they can hide the block if the hash is not favorable. The traitors cannot stop a loyal general from announcing a block (we assume a perfect network). The traitors will produce unfavorable random numbers 1/2 of the time, meaning they relinquish control to the loyal generals in 1/2 of all cases. For 1/4 of the time, the traitors produce favorable hashes and announce the blocks. Of the 3/4 remaining blocks, the loyal generals produce favorable numbers to the traitors for 1/2 of all of them. This means the loyal generals will elect traitors 3/8 of the time. In total, a traitorous general will be elected 5/8 of the time, even when 1/2 of all resources are controlled by traitors. Additionally, each manipulation results in a reward that the traitors cannot claim which they otherwise could have claimed.

As less resources are controlled by traitors, a traitor is elected commander with lower frequency, because the total amount of manipulation possible by the traitors is reduced. This benefit is increased substantially when multiple parties are using the same method for generating random numbers. Let us say that the Persian army has the same problem while fighting a different war at the same time as the Byzantine army. If the loyal Persians can collaborate with the loyal Byzantines, then the traitorous Persians are forced to collaborate with the traitorous Byzantines. When traitorous generals in either army attempt to manipulate the entropy, they will be fighting 3 other parties who are attempting to generate entropy. Collaboration between the traitorous generals of each camp will at best reduce overall effectiveness. The traitorous Byzantines only care that a traitor Byzantine is elected commander, and the traitorous Persians only care that a traitorous Persian is elected commander. If they do collaborate and only announce blocks where both a traitor Byzantine and a traitor Persian are elected, they will need to wait until they have a random number that fills both conditions. Such a number is exponentially less likely as the number of armies participating increases. This gives the loyal generals a much higher probability of producing random numbers that the traitors have not manipulated.

2 Aiza, A New Cryptocurrency Optimized for Lightweight Random Number Generation

Bitcoin could be used to collect seeds for random number generators, however Bitcoin's primary purpose is as a financial tool. The Bitcoin blockchain is already 16GB and will only grow with time. Should Bitcoin gain popularity, the size of the blocks released from Bitcoin will also increase, as will the required bandwidth for listing on the network. When generating random numbers, all of this activity must be viewed as noise. It cannot be used for entropy, because an attacker may be able to manipulate the resulting random number by submitting contrived noise to the network.

Aiza aims to be a much lower noise, more lightweight network that produces entropy on average once every 2.5 minutes. To maximize the potential hashing resources, Aiza will be merge-mined with Bitcoin.

To reduce noise, two steps are taken:

- Each block only mines a single aiza
- Transactions must consist of integral volumes of aiza.

To reduce network weight, only the most recent 17,500 blocks are kept. This represents approximately one month of history. Beyond that, a network snapshot is used containing a list of all the wallets and the volume of aiza in each wallet. Empty wallets are not stored in the network snapshot. To reduce network weight, blocks can be at most 1kb in size, limiting the total number of transactions. The total size of the blockchain is 17,500 * 1kb, or 17.5 megabytes. We can also bound the maximum size of the network snapshot, as it is only a list of wallets, and each wallet must have an integral volume of aiza. If each wallet is 64 bytes, then the network snapshot can be at most 64 bytes * (total volume of aiza ever produced). After 4 years, this amounts to roughly 54 megabytes. Realistically, most wallets will have much more than a single aiza, meaning the snapshot will be smaller. As time advances, technology should get better and the growing size of the snapshot should be decreasingly expensive.

Another feature added to the aiza network is the ability to destroy aiza. At any time, a wallet may announce the destruction of some integral volume of aiza in its possession. This enables entities dependent on random numbers to encourage a minimum value for aiza. By purchasing and then destroying aiza, they are rewarding the miners on the network for producing random numbers. Though the same effect could be achieved by purchasing aiza and never spending them, the act of destruction ensures scarcity of the aiza.

Finally, Aiza will feature a very basic scripting system, similar to but less complex than the Bitcoin scripting system. The only goal of the Aiza scripting system is to enable a trustless exchange between Aiza and other scripted cryptocurrencies. Scripts expire after 17,500 blocks and are not stored in the network snapshot.

3 Potential Problems

Entropy is produced by taking a random input and producing a hash that is less than random. (IE fulfills some hashcash restraints) We get back to a random number by taking this non-random number, appending it to the string that produced it, and hashing their concatenation. I'm not sure that this actually produces a random number, or if it's somehow got less bits of entropy because part of the input is non-random.

The scripting system may prove less than lightweight if 17,500 blocks worth of scripts must be transversed every block. Additionally, the scripting system is only partially explained.

4 Conclusion

This relatively simple coin has been created as a lightweight solution to the problem of deterministic public entropy generation.