KMCoin: A Shared Chat Log Cash System

C Morgan
TheKittyMine@proton.me
www.thekittymine.com

February 19, 2025

Abstract

A purely chat based version of gaming cash would allow online payments to be sent directly from one player to another without infringing upon the game developers End User License Agreements. Digital signatures created by the players private key and their first mined block are used to verify transactions while reading only the shared chat log among players. In this paper, we introduce a Proof-of-Game, layer-1 consensus mechanism that eliminates centralized control.

1 Introduction

Game developers currently have complete control over their in game economy, requiring players to spend varying amounts of money over time to continue enjoying a game. By using a consensus hash derived from all the previous block and ledger information, players create an immutable string that may be shared freely to verify ledger authenticity.

In this paper, we propose a solution that eliminates single-party control in a chat based cash system, allowing players to control their own private keys while doing transactions in-game.

2 Transactions

We define an electronic coin as an immutable string of publicly verified transactions and clearly defined in-game Proof of Work mechanisms that may not be exploited by players. Player transactions are collected between blocks when a hash that solves the sending players public key is found. Since this information is now public, all players will increment the players transaction number when they see the correct transaction hash. This allows the player to send another transaction with the solution to the previous hash, repeating this process to tx number 1000.

On the player's 1000th transaction a new public key will be appended that was also created from their first block mined, private key and a "1". This changes the public key while still using previously stored data and thus reducing computation time. If the 1000th transaction hash solves the public key by

$$\text{sha256}^{1002}(FirstBlock + PrivateKey) = \text{sha256}^{2002}(FirstBlock + PrivateKey)$$

then the player has proven their knowledge of their private key and the new public key is attributed to the player by the entire network. We repeat this process infinitely while incrementing the public key nonce by one each time a player does a multiple of 1000 transactions.

3 Proof of Game + Proof-of-Work

When a player is rewarded for doing a singular objective, there is little control over preventing over-mining by exploiting the way in which this objective is executed. A single-action reward system also reduces the fun a player may experience as they often feel obligated to be as efficient as possible while mining. We have created a barrier by controlling the rate a player may be randomly selected for

a block hash attempt thus allowing players to passively enjoy the game while still getting the same random chance to mine a block as all other players. This reward system over time will equally reward all players who have done the in-game work necessary to obtain these attempts. Furthermore, as the game ecosystem grows, players will be less inclined to do redundant mining as they will be less likely for a random selection.

4 Ledger Packing + Consensus Hash + Sync Time

A major problem with a traditional Proof of Work mining system that uses computational power to guess long hashes is energy consumption and sync time. Rather than rely on the entire recorded ledger, this system builds a "current ledger" up to a fixed file size, then hashes then entire file and stores it in a final directory. Players may download the entire directory, but it is not necessary as the consensus hash is computed as

$$\text{sha256}^2(sha256^2(ledgerhashes.log)+sha256^2(playerinfo.log)+sha256^2(ledgercurrent.log))$$

where ledgerhashes.log contains each new hash created by a previously stored current ledger that reached the maximum file size and is computed as

$$\text{sha256}^2(currentledger.log) = \text{(hash to append)}.$$

The playerinfo.log document contains the wallet address, balance, transaction number and public key of the players. It is read by all players for each transaction to verify public keys and is updated each time a transaction with the correct hash or a new block is found. Each players information present on playerinfo.log only requires an average of 120 characters to store all the information, depending on player name, balance and transaction number. This means when 1 million players have joined the ecosystem and mined a block the playerinfo.log will be be less than 120Mb. Clearly ledgerhashes.log will be the smallest of the three files as it will only be 64,000 bytes after 1000 current ledgers (10Gb total ledger size stored in final directory) have exceeded the maximum file size. With these parameters in place one may simply source the three files detailed above and sync between blocks in the time it takes to download at maximum 220Mb. Furthermore, the KMCoin system was made to store 1Tb of ledger before running out of space while taking the top 100 transactions per block. Given the average transaction size is 130 bytes, and the average block size without transactions is 300 bytes produces a maximum block size of 13,000 bytes. This means at maximum transaction volume 1 million blocks would take 13Gb of ledger stored permanently in the final directory. And since block time is one minute, the length of time necessary to build this ledger would be 1 million minutes = 1.9 years, yielding a 1Tb ledger size in

$$1.9\text{yrs} * 1Tb/13Gb = 146years.$$

Once a player has mined their first block, all subsequent block hashes added by the player to the blockchain may be viewed as "salt", further increasing the complexity of the chain. No player in the ecosystem will reference a block mined by the player after their first block. This indicates that a player no longer has to run the in-game program for other players to reward their work since they have been populated on the player info page. However, to perform transactions the player must input the correct private key so that all other players can verify it solves the player's public key according to their transaction number.

5 Game Conditions

Since all players may not receive the chat at the exact time due to latency and other factors there is no time-stamp built in to the ledger but may be implemented with a p2p network much like Bitcoin [1] and is necessary to move these coins outside of the game. However, even without a time-stamp, a strictly in-game economy may be implemented given the game follows these conditions.

1. Have a process for obtaining an otherwise unattainable object through transparent work done by a player and use this special item to trade for hash "attempts".

2. Prevent "double-mining" so more than one player may never try a hash at the same time.

3. Create a random queuing method that will select players for mining.

4. Create the necessary level of perceived randomness in-game via a non-exploitable process that generates the odd or even number determining the success of the hash attempt.

5. Make an area within the game to view all saved information created by this non-exploitable process.

6. Do not allow actions by the player during block hashing that would compromise the block number. Ie. everything must happen at block time so that if a player leaves the attempt will fail and move to the next player.

6 Conclusion

We have proposed a system for an in-game blockchain complete with transactions that players and developers may utilize to regain control over their game economy while also syncing very quickly with other players. There is no infringement upon the game developers EULA and players may use this system freely at their own discretion. The system is simple and very easy to develop.

7 Acknowledgments

Special thanks to Galactic-Hermit-Crab for proof reading, the community that stepped up to support this project and the family and friends that loved me during development no matter what.

8 References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://bitcoin.org/bitcoin.pdf, 2008.