LEDGER
ledgerjournal.org

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.15

RESEARCH ARTICLE

Game Channels for Trustless Off-Chain Interactions in Decentralized Virtual Worlds

Daniel Kraft[*][†]

Abstract. Blockchains can be used to build multi-player online games and virtual worlds that require no central server. This concept is pioneered by Huntercoin, but it leads to large growth of the blockchain and heavy resource requirements. In this paper, we present a new protocol inspired by payment channels and sidechains that allows for trustless off-chain interactions of players in private turn-based games. They are usually performed without requiring space in the public blockchain, but if a dispute arises, the public network can be used to resolve the conflict. We also analyze the resulting security guarantees and describe possible extensions to games with shared turns and for near real-time interaction. Our proposed concept can be used to scale Huntercoin to very large or even infinite worlds and to enable almost real-time interactions between players.

1. Introduction

Blockchains and, in particular, the Bitcoin protocol have proven to be a genuine innovation that allows transactional systems to get rid of trusted parties and central instances for various interactions among peers.[14] Most of the efforts in this space are focused on financial activities, just like Bitcoin itself. See, for instance, Omni (formerly Mastercoin)[6] or tø.[17] Non-financial (or, at least, not exclusively financially motivated) systems are Namecoin,[20] StorJ,[18] and Ethereum.[9] This incomplete selection of blockchain projects demonstrates already how useful and versatile blockchain technology can be.

Another very interesting situation where blockchains can help remove trusted parties is *online gaming*.[21] This is pioneered by Huntercoin,[2,8] which uses a blockchain to implement a multi-player online game world without any central servers. Every node on the Huntercoin network has the ability to participate in the game and also to verify that the current game state is valid according to the game rules. This allows developers to build a *provably fair* multi-player world where users can earn coins and transfer them to their on-chain wallets. This aspect of the game leads to *human mining* of huntercoins. More details will be given in Section 2.

Let us, in this context, also mention another approach to human mining: *proof-of-play*, which was introduced by Motocoin.[22] Unlike Huntercoin, this blockchain is not associated to a global game state. Instead, the proof-of-work algorithm itself, which is used to mine coins and secure the blockchain, is formulated in terms of a game. This can be compared directly to the analogy of solving Sudoku puzzles given in Section 2.4 of "Mastering Bitcoin" to explain Bitcoin mining.[3]

[*] 1domobKsPZ5cWk2kXssD8p8ES1qffGUCm

[†] D. Kraft (d@domob.eu) is a mathematician and the current Huntercoin main developer.

Motocoin defines a simple video game that is based on a deterministic physics simulation. By solving a level, human miners construct a verifiable chain of commands that prove that they found a solution. This proof is then attached to blocks instead of a classical proof-of-work based on partial hash collisions as used in Bitcoin and most other blockchain systems.

While blockchain technology allows for virtual worlds and multi-player gaming in a completely decentralized, trustless environment, it adds, of course, the usual technical difficulties. In particular, storing the full history of the game world may cause large growth of the blockchain and subsequent scaling issues. This can be observed in practice on the history of the Huntercoin blockchain, as we will discuss in Subsection 2.2. Furthermore, worlds based directly on the blockchain are quantized in time by the block interval. Depending on the actual game logic, this may only be a minor nuisance but it can also be prohibitive for certain types of interactions among players. To overcome both of these issues, we propose a protocol for *off-chain interactions* between players. Our *game channels* were first introduced in a post on Bitcointalk.[13] We give a more detailed discussion for the simplest case of a turn-based interaction between two (or a few) players in Section 3. We discuss the resulting security guarantees in Section 4. The turn-based model will be extended to more general game concepts in Section 5. In Section 6, finally, we describe how game channels could be applied to enhance scalability and gaming experience for Huntercoin in the future.

2. A Guide to Huntercoin

In this section, we give a brief introduction into the important concepts of Huntercoin and describe two issues from which the current system suffers. This serves as motivation for the subsequent discussion of game channels, which we apply to tackle these issues in Section 6. Note that the description here is only concerned with the core concepts that are important for understanding the remainder of this paper. We do not describe the game mechanics or implementation in detail; for that, we refer to the codebase and description in the original announcement.[8]

Huntercoin features a native cryptocurrency (huntercoins, HUC) that can be transferred using Bitcoin-like currency transactions and is mined using proof-of-work. It allows for both Bitcoin's double SHA-256 and Litecoin's scrypt algorithm for that purpose, and can be merge-mined with Bitcoin and Litecoin. The main feature, however, is an embedded multi-player game that is set in a two-dimensional world (termed *Chronosius* by the creators). See Fig. 1 for a screenshot of the game and Youtube for a time-lapse replay of the first month of the online world.[7] For a fee paid in huntercoins, users can create *hunters* (corresponding roughly to player accounts) in this game world. This allows for *human mining*: Parts of the block rewards are not paid to the proof-of-work miner but instead placed inside the game world, where hunters can pick them up and *bank* them to their on-chain address. This is not straight-forward, however, and requires skill since other hunters can fight for and steal the coins until they are secured. This is intended to give humans a chance to "mine" huntercoins by playing the game. Due to the use of a blockchain, everyone is able to fully validate the history of all moves and verify for themselves that all human-mined coins are paid out correctly according to Huntercoin's game rules. This *provably-fair gaming* sets Huntercoin apart from existing multi-player games where a central server handles the game logic.

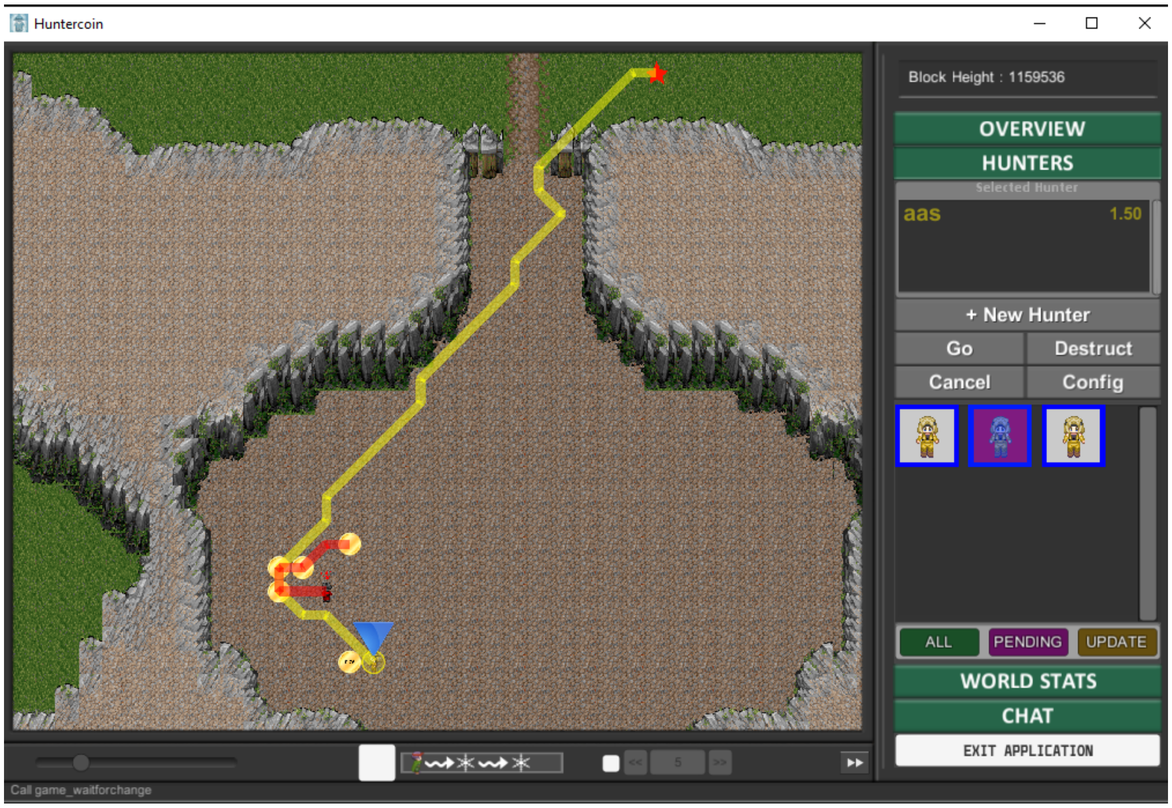


Fig. 1. Screenshot of the Unity client for Huntercoin, showing hunters collecting coins in the game world. Contributed by Andrew Colosimo.

2.1. Implementation of the Game—Let us now take a closer look at how the game is implemented internally and how it interacts with the underlying blockchain. In very abstract terms, one can interpret Bitcoin in the following way: The current ledger (Unspent Transaction Output or UTXO set) is a common state on which all nodes in the network agree. This state can be mutated by everyone through transactions, where the individual, discrete changes from one block to the next arise through *batches* of transactions (the blocks). The same abstract model also applies to Huntercoin, except that the state and transitions are more complex: In addition to the UTXO set necessary for the underlying cryptocurrency, Huntercoin's shared state also includes all information about the game world. This state evolves from block to block according to the consensus-critical game mechanics coded into the clients. Users can influence the game world by crafting special non-currency transactions: When they are included in a block, they have a certain effect on the game-state transition (again determined by the game rules).

This is, of course, a description on a very high level. For understanding the concrete implementation, the first important remark is that Huntercoin's code (but not the blockchain) is forked off Namecoin,[20] which in turn is based on the Bitcoin codebase. Each of these adds a separate layer of functionality: Bitcoin's code is responsible for managing the blockchain as well as the currency aspects (huntercoins, UTXO set, addresses, currency transactions). Namecoin implements a decentralized key/value database on top, which allows users to register unique names and subsequently attach (and update) data to the names. Roughly speaking, each name is represented by a *colored coin*. Only the owner of the corresponding private key is able to attach or update data, which is done by sending the particular coin representing a name through a special transaction. Huntercoin, finally, interprets names as individual hunters. If a name is updated through a transaction in a block, the attached value is interpreted as a particular move that changes the hunter's behavior in the game world. Such a move may set the hunter's target coordinate to have it walk on the map, or it may signal to attack a nearby enemy. The details of the game mechanics are beyond the scope of this paper.

Let us also stress that coins can flow bidirectionally between the ordinary UTXO set and the game world: According to the Namecoin protocol, coins that are associated with a name and accordingly colored can never be "uncolored". Thus, it is possible to "destroy" coins in the UTXO set by increasing the amount locked in a name. Such an action can then be interpreted by the game engine to create coins in the game world according to certain rules. The other way round is also possible: After processing the game-state change in a particular block, the Huntercoin client may create special *game transactions* that alter the UTXO set. This is done, in particular, both when hunters are killed (then the coins corresponding to their names are sent into a transaction without outputs) and when coins are banked in the game world (then a transaction is created that sends the corresponding value on-chain to the user's designated address). These transactions are (by definition in the consensus protocol) valid without signatures, as they can be deterministically constructed and verified by each node independently according to the game rules. Putting everything together, the rules for processing a block in Huntercoin look like this:

(1) Validate the block according to Bitcoin's rules. Most importantly, check that the proof-of-work and all signatures are valid and that all transaction inputs are in the current UTXO set.
(2) For each transaction that modifies a name, verify the transaction in addition to Namecoin's rules and check that the attached move is valid according to Huntercoin's game mechanics.

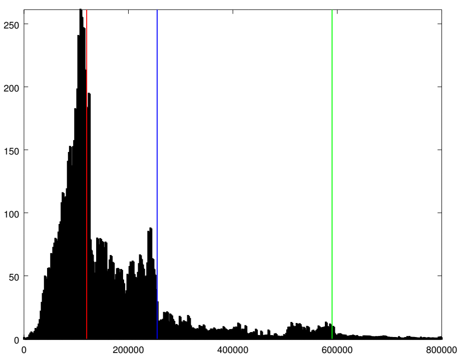
(3) If the block is valid, update the UTXO set according to all transactions.
(4) Load the current state of the game world, perform changes to the hunters according to moves in the current block, and use the game engine to compute the next game state. Save this game state together with the new UTXO set.
(5) If hunters are killed or coins are banked in the game world, construct the corresponding game transactions according to the game rules and update the UTXO set accordingly.

Finally, note that it is possible for the game to contain random events even though everything is deterministically verifiable by each node: For this, a block's hash is used as a seed to generate pseudo-random numbers that influence this block's game-state transition. In theory, miners could discard blocks with hashes that yield "random" changes they do not like, but this risk is minor in practice for several reasons: Most Huntercoin miners (merge-mining pools) are not active players and thus do not directly care about game changes. Furthermore, mining rewards incentivize them to publish blocks as soon as they are found, unless the supposed gain from cheating is larger than the block reward. So far, there is no evidence that any miner tried to mount an attack like this. In case this changes, it would be possible to adapt the protocol to prevent such an attack by using a combination of the last *n* block hashes, for instance.

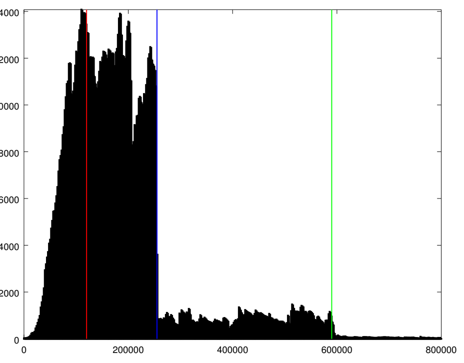
2.2. The Usual Pain with Scaling—One of the complaints users have about Huntercoin is the slow and unpredictable speed: The game is effectively quantized in time by the individual blocks. It is debatable whether or not this is a real issue for Huntercoin's current game mechanics (some players argue that the block time is actually fast for controlling multiple hunters manually during a fight), but it definitely is a major limitation. We will describe a strategy for removing this limit in Subsection 5.2.

To see the two most important issues, however, let us take a look at some metrics of the Huntercoin blockchain: Fig. 2 shows some important statistics about the first 800,000 blocks, spanning the time interval from Huntercoin's launch on 1 February 2014 to 23 July 2015. Also note that the vast majority of transactions over this time period, namely 93.8%, are moves. This clearly shows that Huntercoin achieved its goal of being first of all an online game and only second a clone of Bitcoin. Initially, Huntercoin was explicitly designed to allow move transactions without fees to be mined. This led to a very fast pace of blockchain growth and corresponding *scalability issues*. Huntercoin reached average block sizes of almost 100 KiB and a chain size of multiple GiB just a few weeks after its launch. This made it necessary to introduce mandatory fees for move transactions, which were implemented at the point in time indicated with the vertical red lines in the plots. This significantly reduced the number of transactions and the block size, which is clear from Fig. 2a. It also visibly stalled the blockchain growth in Fig. 2d, producing the first "dent" in the curve.

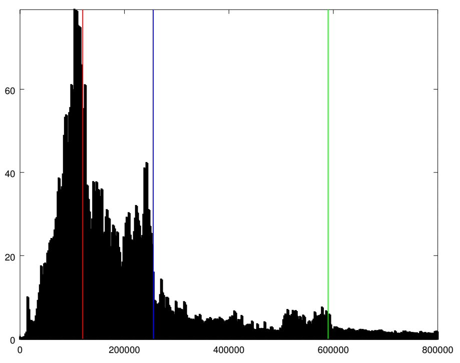
The number of hunters on the map (see Fig. 2b) remained high, though, which was mostly due to *bots* controlling large armies. This was a major issue for human players, particularly because the initial game rules made it easy for bots to dominate the map just by sheer number of hunters. The rules were refined over time to give human players a competitive edge again. Two particular changes that made it more expensive to create and control hunters are indicated with the blue and green lines. These clearly led to a reduction of the number of hunters on the map (see Fig. 2b) and further stalled the growth of the blockchain, producing two more dents in Fig. 2d. The first of these (blue line) is particularly drastic: It introduced a game rule that kills every hunter on the map at randomly chosen instances in time (when a "natural disaster" strikes).



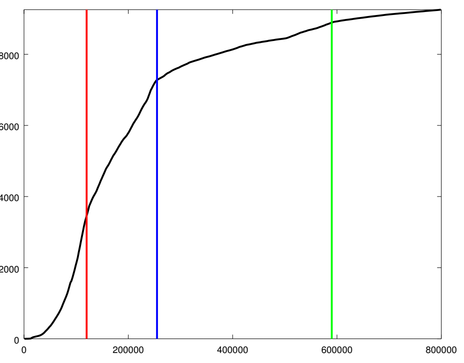
(a) Move transactions per block



(b) Active player teams on the map



(c) Average block size in KiB



(d) Size of the blockchain in MiB

Fig. 2. Statistics about the initial 800,000 blocks of the Huntercoin blockchain. The vertical lines indicate important changes to the protocol, which prevented the network from being overloaded. See the discussion in Subsection 2.2.

The first disaster, corresponding to the huge drop in players on the map in Fig. 2b at the blue line, can be seen around 2:21:40 in another time-lapse video of the game world.[5]

After these changes, it seems that most hunters are, again, controlled by humans instead of bots, and that the rate of growth of the blockchain is under control. However, scalability is still a major difficulty for Huntercoin as well as potential future competitors: The emergency fixes introduced in Huntercoin were successful, but only by introducing strict limits into the system. This is not desirable for obvious reasons, since the ultimate goal is to create a successful online game and not an artificially restricted game world. It is the purpose of the remainder of this paper to introduce an idea that could potentially allow Huntercoin and other game worlds to scale to literally infinite size (of the world itself) and large amounts of players while still keeping the size of the underlying blockchain in check. For this, we introduce the concept of *game channels* in an abstract form over the next sections, and discuss its applicability to Huntercoin in Section 6.

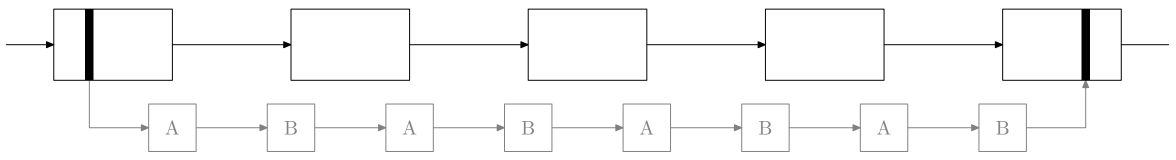
3. Game Channels for Turn-Based Interactions

In Bitcoin, an important proposal for off-chain transactions and improved scaling is the Lightning Network,[16] based on payment channels.[1] They allow for off-chain and instantaneous transactions between two parties. The involved parties need not trust each other, while the working hypothesis is that "usually" no disagreement will occur. In this case, almost all details of the interaction can be done privately without ever putting a burden on the Bitcoin network or blockchain. However, if a dispute arises, it is still possible to use the peer-to-peer network and blockchain to resolve the conflict without any trusted third party or other authority. This concept inspires the game channels we introduce below. Another important scaling innovation are pegged sidechains,[4] which also form a basis for our game-channel protocol. Indeed, our protocol will be based on private chains branching off and running parallel to the public main chain.

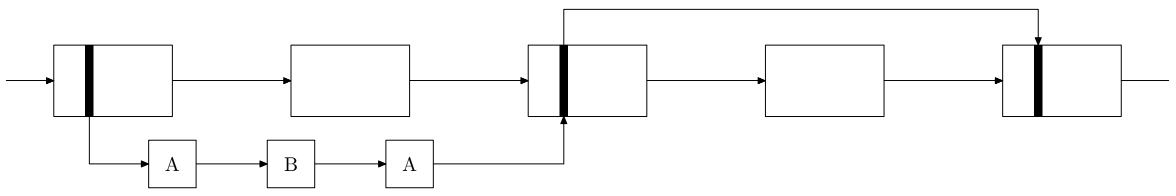
More generally, the problem of handling interactions between two or multiple parties according to some rules in a secure and trustless manner has been of interest for quite some time. One example is research about optimal contract signing.[15] There exists also more recent work based on cryptocurrencies and blockchains.[10,11] Our own protocol will be described over the next sections in an abstract way to make the important concepts and ideas as clear as possible, but it is not meant to be a state-of-the-art result about private interactions between players. Instead, our focus is the application of the protocol to enhance massively multi-player online worlds such as Huntercoin, to which we come back in Section 6. In fact, it may be possible to optimize the protocol in the future using some of the modern protocols for contract signing and private games.

Throughout this section, we will always assume that two players establish a game channel to compete in a turn-based game with each other for some amount of coins. One can extend this to more than two players, but this is outside the scope of the current work. It is also possible to reduce more general situations than alternating turns to this model, which we will discuss below in Section 5. Thus, the interaction we are interested in looks like this:

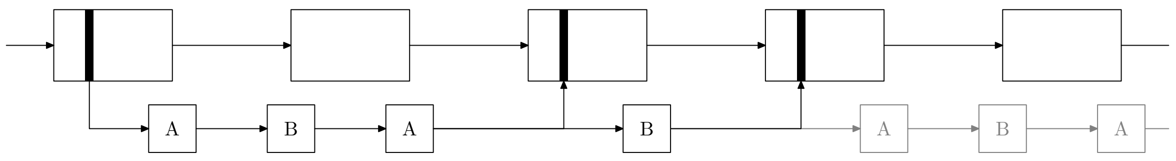
(1) Both players agree on the rules and modalities of the game and co-fund the prize. This opens the game channel.
(2) They make their moves in alternating turns. Each move modifies the current game state deterministically according to some defined rules.
(3) Certain states end the game and result in a particular distribution of the prize money



(a) Consensus about the game with no dispute at all. The private chain indicated in grey can be discarded after the game channel is closed.



(b) Alice files a dispute and receives the prize money after waiting for the threshold time to elapse. The sequence of moves (black) is recorded permanently in the public blockchain inside of the dispute transaction.



(c) Alice files a dispute, Bob resolves the dispute with his next move, and the game continues in agreement. Only the part of the game up to the dispute-resolution transaction needs to be in the public blockchain.

Fig. 3. Sketch of the blockchains involved for a game channel in various scenarios. The chain on top is the public blockchain, while the chain below is the private chain containing blocks mutually created and signed by Alice and Bob (marked with “A” and “B”, respectively). The dark bars indicate transactions related to the game channel included in the public blockchain.

between both players. The prize money is paid out to the players accordingly and the game channel is closed.

For the following discussion, the situation we have in mind is this: Both players on a game channel communicate privately and directly to each other. This can be done by a direct (and potentially authenticated and encrypted) network link. This connection is used to relay moves to each other as well as build a private blockchain. Both players are still connected to the public peer-to-peer network as well, whose miners build the public blockchain that is not controlled by any player.

3.1. Agreement about the Game—Ideally, both players follow the agreed rules and the player whose turn it is always responds in a reasonable amount of time. In this case, the game itself can be played completely off the public blockchain and proceeds in the following way:

(1) To open the channel, the agreed rules and parameters are encoded in a joint transaction sent to the network. At the very least, the transaction contains public keys that are used to identify each player throughout the current game. This *opening transaction* is co-funded by both players with their respective shares of the prize money.
(2) Once the opening transaction is confirmed, the prize money is locked on the public network and the game channel is open for private interactions between both players.
(3) For the game, the players build a private blockchain containing their moves. This can be

done without proof-of-work, since it is clear whose player's turn it is at every moment in time. Each player appends a block containing his or her next move to the private blockchain. The block is signed by the player's public key.

(4) When a move ends the game, both players construct a transaction that unlocks the prize money and pays it out to them according to the game's result. The public network accepts this transaction and transfers the locked coins accordingly if both players sign this *closing transaction*.

This is sketched in Fig. 3a. According to this protocol, only the opening and the closing transactions must be recorded in the public blockchain. The private sidechain with all individual moves of the game can be discarded once the closing transaction is published and confirmed by the public blockchain. On the other hand, if the sequence of private moves is published, everyone on the network is able to verify that all game rules were followed and that the correct players signed the moves. They can also compute the current game state resulting from each move.

3.2. Dispute Resolution—The more challenging case is, of course, if a conflict arises. Particularly the "losing" player may prefer to stop producing moves or may fail to sign the closing transaction. But it may also happen due to technical issues (like a network failure) that one of the players stops to respond, even if not intentional. It must be possible for the remaining player to recover the prize money in this situation, without giving any player the ability to defraud the other. Thus, we have to define additional protocol rules for resolving disputes. For this, let us assume that Alice made the last move, it is Bob's turn to either respond with his next move or to co-sign the closing transaction, and he fails to respond within a certain amount of time. If he disagrees with Alice instead and sends a response that Alice considers invalid, this leads to the same situation (since Bob fails to provide a *valid* response to her). In this case, the dispute resolution proceeds as follows:

(1) Alice raises a *dispute* by sending the sequence of moves made so far in a special transaction to the public network.
(2) As long as the published sequence is valid, the network confirms the dispute transaction in the public blockchain.
(3) If the game state resulting from the sequence of moves corresponds already to an end state, the network can immediately credit the resulting payments to Alice and Bob.
(4) If this is not the case and if Bob fails to give a valid response, the public network credits Alice the full prize money after a certain number of blocks in the public chain. This is similar to the lock time in Bitcoin; compare, in particular, BIP 65.[19] See Fig. 3b.
(5) On the other hand, Bob can publish his next move in another special *dispute-resolution transaction*. This transaction is accepted and confirmed by the network if it contains, indeed, a valid response to Alice's last move published in the dispute.
(6) In this case (illustrated in Fig. 3c), the dispute is resolved and unilateral payment of the prize money to Alice is cancelled. Since also Alice has now a new valid move by Bob, the private game can continue until it finishes orderly or another dispute is raised by one of the players.

4. Security Analysis

We proceed now with a more detailed analysis of the security guarantees offered by the protocol of Section 3. In order to allow for trustless interaction, game channels must, of course, ensure that no player can cheat with respect to the game rules. Furthermore, the dispute resolution of Subsection 3.2 must ensure that any conflicts can be resolved in Alice's favor if Bob fails to respond reasonably, while ensuring at the same time that Bob is given a chance to rectify disputes filed in error. And, in fact, it turns out that our protocol fulfills all of these requirements. In particular, it has the following properties:

[Integrity] The current game state is always well-defined and verifiable for every party that has the private blockchain of moves. Since each player is required to sign their moves, it is impossible to forge a move. Furthermore, note that it is also not possible for any player to replace or take back a move already made: If Alice publishes at any time two or more signed moves based on the same game state, Bob may choose either to build his next move on. Since Bob and only Bob is able to produce a valid next block (signed by his public key), he will always be able to produce the longest valid blockchain in a dispute claim. For this, he can pick for himself which move of Alice to include if she publishes multiple ones. This ensures that the game rules are followed by both players and strictly alternating turns are taken with no way to forge the history once a move has been made.

[Non-Stalling] Alice is guaranteed to always receive Bob's next move within some well-defined time or else be declared winner if Bob fails to respond. This is guaranteed by the dispute process, which pays her the full prize money unless Bob publishes a valid move within the threshold time. Furthermore, as soon as a final end state of the game is reached, Alice can immediately claim her rightful share of the prize money by publishing the full game history if Bob refuses to sign the closing transaction. Note that the threshold time is given as a *number of blocks* according to Subsection 3.2. The corresponding real time is a random variable following a well-known distribution.[12] It is also possible to define the threshold time directly in terms of real time (in the same way as Bitcoin's lock time works), but this is out-of-scope for the current work.

[Fraud Proofness] If Alice publishes a dispute claim while Bob has, in fact, no intention to stop responding according to the game rules, Alice can not use the dispute process to defraud him. As long as Bob is able to get the dispute-resolution transaction mined (which we assume is the case, at least if he pays a reasonable transaction fee and mining is not controlled by Alice alone), he is always able to prevent that from happening: Since Alice's dispute claim must contain a sequence of moves with the last one by her, it is up to Bob alone to produce a longer sequence that will be accepted by the network as dispute resolution and thus cancel the unilateral payment to Alice.

Let us now also mention two undesirable situations that can still occur with the proposed game-channel implementation: First, if a player drops out of the game *unintentionally* due to a network failure or other problem, the game is lost for him or her unless the threshold time is long enough for them to restore network access and still respond in time. This cannot be prevented since the goal of game channels and the dispute process is precisely to guarantee non-stalling. The threshold time is part of the initial contract agreed upon by both players, thus this is not "unfair" in any way.

Second, non-stalling guarantees only a certain maximum time between moves. This time must, furthermore, be chosen much larger than the expected time for each move in the game. Thus, it is possible for an attacker to deliberately delay the game and bloat the public blockchain with dispute transactions in the process. We believe that this is unlikely to happen on a larger scale since the behavior produces no meaningful economic benefit to the attacker. Furthermore, since the players initially both agreed to the rules and expressed an interest to open the game channel, they have, in general, a stake in the game network and thus no interest to disturb it unnecessarily. As there is not yet any public experience with similar concepts on a large scale (like the Lightning Network), however, it remains to be seen how well this assumption holds true in practice.

5. Extensions to More General Situations

So far, we only considered the situation of two players and a game that is formulated in terms of alternating turns between both players. This is, of course, a model that fits not to every potentially desirable application of game channels. In this section, we explain how two more general situations can be reduced back to a turn-based model so that game channels as described in Section 3 apply.

5.1. Shared Turns—Let us first assume that the game is still turn-based, but in a way that both players perform moves *at the same time*. In other words, the timeline of the game is quantized into turns, but each turn is not "controlled" by one of the players. Instead, they can *both* perform a certain action at every turn, and the resulting game state is determined from the last state by the combination of both turns. This is the game model that applies to Huntercoin, for instance. The difficulty in this situation is that learning the move of one's opponent may give an unfair advantage to a player that has not yet decided about a move. To prevent this effect, one can use a combination of *micro-turns* and hash commitments. Let us assume that Alice and Bob play a game with shared turns. On the protocol level, the game is still split into a series of alternating turns, where each shared turn consists of four micro-turns in the following way:

(1) Alice publishes a hash of her move.
(2) Bob publishes a hash of his move.
(3) Alice reveals her move, which must match the previously published hash to be accepted.
(4) Bob reveals his move in the same way.

See Fig. 4. As before, each player signs the corresponding micro-move by his or her public key. After the fourth micro-move has been made, both players can compute the next game state. Furthermore, due to the hash commitment in item (2), Alice knows that Bob must have fixed his move without knowledge about her plans and vice versa. Thus, this really leads to genuine shared turns without giving any player an unfair advantage. There only remains one difficulty: If the set of possible or plausible moves for Alice is limited, Bob may enumerate and check all of them against the hash to compute a preimage. In this case, Alice has to include a random salt into the move description.

5.2. Near Real-Time Interactions—Another important class of games requires (near) *real-time interaction*. Instead of clearly discrete time steps, there is a seemingly continuous flow of time and each player may take an action at arbitrary moments. It is possible to mimic such a real-time game with a game channel as well. Of course, the processing power of the players'

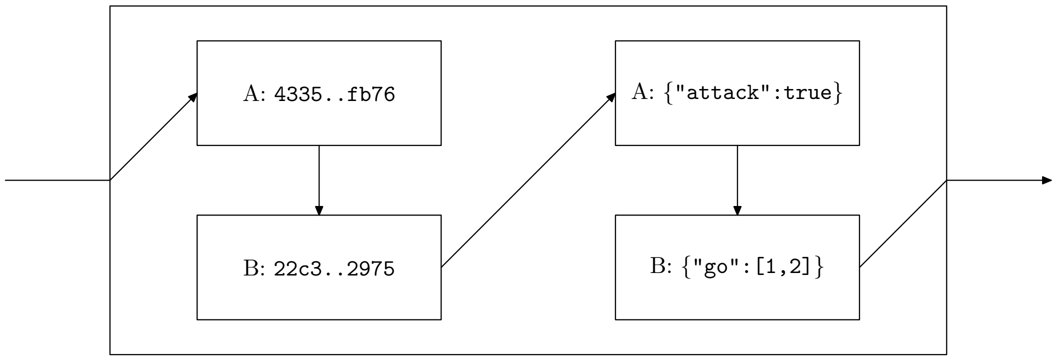


Fig. 4. Illustration of a single shared turn that is made up from four micro-turns alternating between Alice and Bob. The hash commitment is computed using SHA-256.

computers as well as the bandwidth and latency of their network connection still leads to discrete steps in time. They are, however, no longer limited by the block time of a public blockchain or by some other external constraint. As computing power and network technology progress, these restrictions are naturally relaxed; in the future, only the speed of light may remain as the ultimate limit for latency of global communication.

In order to fit such a near real-time interaction into the turn-based model of a game channel, we only have to allow "do nothing" as an explicit move available to each player. Then, the players' clients still take turns adding blocks to a private blockchain—this time, however, blocks are produced as frequently as the computing power of each client allows. Most of the blocks will be "empty" and refer to a do-nothing move, unless one of the players actually decided to trigger an action at the current moment in time. This leads, of course, to a rapid growth of the private blockchain. Since it can be discarded after the game if no disagreement occurs, this is, however, not a big issue.

6. Applications to Huntercoin

We are finally ready to describe possible solutions to the issues under which Huntercoin in its current form suffers (as described in Subsection 2.2). The main issue identified is that of scalability. We believe that game channels can be used to greatly improve scalability while at the same time also enabling much larger (possibly even infinite) game worlds. This can be achieved by *sharding* the world. Each shard corresponds to a part of the world where a group of hunters is located close to each other. All of them propagate the world in a game channel, such that their interactions can be discarded unless a disagreement about the game mechanics arises. This way, the public Huntercoin blockchain no longer needs to record all moves throughout the entire game history, and validating nodes are not obliged to process moves for each and every hunter. Instead, nodes only process moves in shards in which they are currently playing themselves, and the public blockchain is only necessary for global management of the shards and for dispute resolution. Furthermore, in such a design one can even use near real-time interactions according to Subsection 5.2 to speed up the gaming experience beyond the block time.

Note that there is not yet a functional implementation of this idea, and there are many ways to implement the details of this sharding and to adapt the precise game rules to shards. For instance, a major decision will be how in-world rewards are distributed among the shards. Let us, nevertheless, give some more details of a possible design: As described above, each shard that is currently active (*i.e.*, where hunters are currently playing) is associated with a game channel. In addition, a special address is associated to the shard. When a hunter wants to enter (including when the shard is initially created by the first who enters), the player has to pay a certain fee which will be locked in the shard address. This corresponds to the deposits of prize money in game channels as described above. Mining rewards that are assigned to be placed in a certain shard are also deposited into this address. Instead of using a plain multisig address, however, the protocol requires that *all players that are currently in the shard* have to sign transactions paying from the shard address. In other words, the set of signers is *dynamically determined* according to the current game state. This allows the game to securely distribute coins that are rightfully earned inside the shard to the correct player, as long as everyone agrees that the game rules have been followed. Disputes, on the other hand, can be handled according to the outline given in Subsection 3.2. Finally, when a hunter leaves the shard again, the initial deposit will be refunded. If the player refuses to cooperate (*e.g.*, fails to sign blocks), the other players can also raise a dispute to kick the offending player out of the shard, possibly killing the corresponding hunter. The deposited coins can then be used to compensate the cooperating players for the delays and inconveniences due to the dispute. This way, players are incentivized to continue processing moves for the shard while active there, since failing to do that strips them of their deposited coins.

7. Conclusion

In this paper, we have described a method for handling private interactions of players or, more generally, agents, following defined rules. While such protocols are not new and our model is not state-of-the-art in the field, we showed how game channels can be used to greatly improve the current design of blockchain-based decentralized multi-player game worlds. These worlds (such as the pioneering Huntercoin) suffer from the fact that moves are limited in time to block intervals, and they are constrained by the practically available blockchain space and processing power of full nodes. Based on game channels and the approach sketched in Section 6, these limits can be lifted and potentially infinite game worlds implemented in an efficient but still decentralized, secure and fair way.

For the future, it remains to implement game channels and such next-generation game worlds in a practical system. It also remains to be seen how common disputes will be; we have reason to believe that most of the time, shards will be processed with consensus among the players, so that only a minor amount of blockchain resources will be necessary for dispute resolution. If it turns out that conflicts are much more common, one probably needs further research to refine the protocol outlined in Subsection 3.2. It may be possible to apply insights from optimal contract signing and other related fields to do that.

Acknowledgement

The author would like to thank Andrew Colosimo, alias snailbrain, for his thoughts about decentralized virtual worlds on blockchains. In collaboration with Mikhail Sindeyev, he turned this vision into Huntercoin. The original idea for game channels came up during a discussion with Andrew about the future of Huntercoin. This work is supported by the Austrian Science Fund (FWF) and the International Research Training Group IGDK 1754.

Conflict of Interest

The author is the current main developer of both the Namecoin and Huntercoin free-software clients. He is a share holder in Crypto Realities Ltd., which is committed to further the development of blockchain-based gaming and virtual worlds, but not involved in any proprietary development for the company.

Notes and References

[1] No Author. "Rapidly-adjusted (micro)payments to a pre-determined party." *Bitcoin Wiki* (18 October 2015) `https://en.bitcoin.it/wiki/Contracts#Example_7:_Rapidly-adjusted_.28micro.29payments_to_a_pre-determined_party`

[2] Alexander, R. "HunterCoin: The Massive Multiplayer Online Cryptocoin Game (MMOCG)." *Bitcoin Magazine* (29 August 2014) `https://bitcoinmagazine.com/articles/huntercoin-the-massive-multiplayer-online-cryptocoin-game-mmocg-1409336751`

[3] Antonopoulos, A. M. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. Sebastopol: O'Reilly Media (2014)

[4] Back, A., *et al*. "Enabling Blockchain Innovations with Pegged Sidechains." No Publisher (2014) `https://www.blockstream.com/sidechains.pdf`

[5] Pseudonymous (BGB HUC). "Huntercoin (HUC) Ttime Lapse Blocks 0–499,999." *Youtube* (27 December 2014) `https://www.youtube.com/watch?v=czUU3Z8spUQ`

[6] Buterin, V. "Mastercoin: A Second-Generation Protocol on the Bitcoin Blockchain." *Bitcoin Magazine* (4 November 2013) `https://bitcoinmagazine.com/articles/mastercoin-a-second-generation-protocol-on-the-bitcoin-blockchain-1383603310`

[7] Colosimo, A. (Chronokings). "Huntercoin - Day 1 to Day 32 - Time Lapse - Human Mine-able Crypto Currency." *Youtube* (5 March 2014) `https://www.youtube.com/watch?v=Q41RW6bxpM4`

[8] Colosimo, A. (snailbrain). "[ANN][HUC] Huntercoin - Human Mining - Decentralized MMO and Crypto Game." *Bitcointalk* (27 January 2014) `https://bitcointalk.org/index.php?topic=435170.0`

[9] Ethereum. "A Next-Generation Smart Contract and Decentralized Application Platform." *Github* (18 October 2015) `https://github.com/ethereum/wiki/wiki/White-Paper`

[10] Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C. "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts." *Cryptology ePrint Archive* **2015/675** (2015) `https://eprint.iacr.org/2015/675`

[11] Kumaresan, R., Moran, T., Bentov, I. "How to Use Bitcoin to Play Internet Poker." No Publisher (2014) `http://www.cs.technion.ac.il/~ranjit/papers/poker.pdf`

[12] Kraft, D. "Difficulty Control for Blockchain-Based Consensus Systems." *Peer-to-Peer Networking and Applications* **9.2** 397–413 (2016) DOI:10.1007/s12083-015-0347-x

[13] Kraft, D. (domob). "Game Channels for Near Real-Time Interaction among Players." *Bitcointalk* (16 October 2015) `https://bitcointalk.org/index.php?topic=435170.msg12699299#msg12699299`

[14] Nakamoto, S. "Bitcoin: A Peer-to-Peer Electronic Cash System." No Publisher (2008) `https://bitcoin.org/bitcoin.pdf`

[15] Pfitzmann, B., Schunter, M., Waidner, M. "Optimal Efficiency of Optimistic Contract Signing." *17th Symposium on Principles of Distributed Computing*, New York: ACM 113–122 (1998)

[16] Poon, J., Dryja, T. "The Bitcoin Lightning Network." No Publisher (2015) `https://lightning.network/lightning-network-paper.pdf`

[17] Rizzo, P. "Overstock Unveils Blockchain Trading Platform at Nasdaq Event." *CoinDesk* (5 August 2015) `http://www.coindesk.com/overstock-unveils-blockchain-trading-platform-to/`

[18] Spaven, E. “Cloud Storage Startup Storj Raises 910 BTC in Crowdsale.” *CoinDesk* (22 August 2014) `http://www.coindesk.com/cloud-storage-startup-storj-raises-910-btc-crowdsale/`

[19] Todd, P. “BIP 65: OP_CHECKLOCKTIMEVERIFY.” *Github* (18 October 2015) `https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki`

[20] Pseudonymous (vinced). “[announce] Namecoin - a distributed naming system based on Bitcoin.” *Bitcointalk* (18 April 2011) `https://bitcointalk.org/index.php?topic=6017`

[21] Wagner, A. “Cryptocurrencies in Video Games: Preview Roundup.” *Bitcoin Magazine* (21 November 2014) `https://bitcoinmagazine.com/articles/cryptocurrencies-in-video-games-preview-roundup-1416609489`

[22] Pseudonymous (WilliamLie2). “[ANN][MOTO] Motocoin.” *Bitcointalk* (1 May 2014) `https://bitcointalk.org/index.php?topic=591724.0`

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2016.15