Acebusters
A real-time Poker Application on the Ethereum Blockchain

Johann Barbie
Email: johann@acebusters.com

***Abstract*—We present a payment channel concept for more than two participants and an efficient protocol to rotate in and out of this multi-party state channel including secure cash distribution. Of this, a busy cash-game poker table with players sitting down and standing up is a prime example.**

Our protocol has an initial phase where a participant interacts with the blockchain to join a state channel. After that, new and previous participants need to communicate only with each other over the course of many message exchanges. At any time a participant can signal intention to leave the state channel and eventually rotate out from the channel without interrupting the ongoing message flow or affecting the security of funds.

We describe the stateful contract model and the security notions that our protocol accomplishes. Moreover, we provide a reference implementation as smart contract in Solidity.

In comparison to existing payment channels, our multi-party state channels are more efficient and have additional features.

Keywords*—*state channels, smart contract, blockchain, poker.

I. Introduction

In 2011, Poker Stars and Full Tilt Poker, the biggest poker rooms at the time, were shut down due to bank fraud and money laundering. Players were unable to access their funds, and the day is remembered as "Black Friday"[1]. This and other scandals make players question the trust model of a traditional online poker room. They manage funds on behalf of the user as well as shuffle and keep the cards secure, hence, posing as a trusted third party.

Shamir et al. [2] introduce mental poker, a set of cryptographic problems to play a fair hand of poker over distance without a trusted third party [3]. This work gave rise to a subfield of cryptography called multi-party computation. The latter enables parties to jointly compute a function over their inputs while keeping those inputs private [4]. This could be the solution for a provably fair shuffle guaranteeing the secrecy of the cards without a need of a trusted third party. However, Cleve demonstrated that a fair multi-party computation without an honest majority is impossible [5].

The basic approach to managing funds and transfers without a trusted third party has been introduced by Satoshi Nakamoto as Bitcoin [6], a peer-to-peer electronic cash system. Blockchain technology like Bitcoin could have prevented the "Black Friday", if players would be able to use it to manage their funds. Unfortunately, interacting with a proof-of-work based blockchain like Bitcoin entails waiting times, which would make real-time games like poker prohibitively slow.

In recent years, the academic study of decentralized cryptocurrencies gave rise to a line of research that seeks to impose fairness in multi-party computation by means of monetary penalties. Kumaresan and Bentov show a scheme in which the parties run an initial setup phase requiring interaction with the Bitcoin blockchain, but thereafter engage in many fair secure computation executions, communicating only among themselves for as long as all parties are honest [7].

It seems as if the mere performance optimization of blockchain could incentivise a fair execution of the game through multi-party computation as well as make the trusted third party redundant regarding the management of the players' funds. Bloomer and Nishimura introduce the first optimization in this regard to Bitcoin with payment channels, circumventing the timing restrictions of blockchains [8]. This initial proposal for payment channels is limited to only two participants.

Bentov et al. introduce efficient protocols for amortized secure multi-party computation in combination with smart contracts, allowing first real-time poker games [9].

To our knowledge all existing multi-party computation protocols require all participants to initiate the game at the same time. While all participants might be available at the start of a tournament, in cash-games participants can sit down or stand up from a table at any time.

A. Our Contributions

We extend existing schemes with the following features:

Asynchronous channel participation: all participants have the ability to join and leave the channel at any time without affecting message flow or security of funds.

Ability to rebuy: when a player runs out of money, he is kept in the channel and has the ability to rebuy by interacting with the blockchain.

As a consequence, we are the first to provide a working implementation that has an acceptable user experience for cash-game tables.

We do not present a complete integration of a multi-party computation protocol with the multi-party state channel. This is due to the high cost of verification [9] of the multi-party computation proof on the Ethereum blockchain , which makes an implementation impractical.

As a temporary solution before the implementation of multi-party computation, the evaluation of the hands is per-

formed by an oracle. An oracle is an external service that has special permissions in the smart contracts to provide data.

II. Extending the State Channel

We explain the operation of a two-party state channel and then extend the functionality to multiple parties. A data structure that models the cash distribution after each hand is introduced. Lastly, a leave protocol is described that protects participants from accepting uncovered bets.

A. State Channels

State channels are a way to think about blockchain interactions which could occur on-chain, but instead get conducted of the chain, without increasing the risk of any participant. The most well known example of this strategy is the idea of payment channels in Bitcoin, which allow for instant fee-less payments to be sent directly between two parties [10].

The life-cycle of a state channel can be broken down into steps as depicted in figure 1:

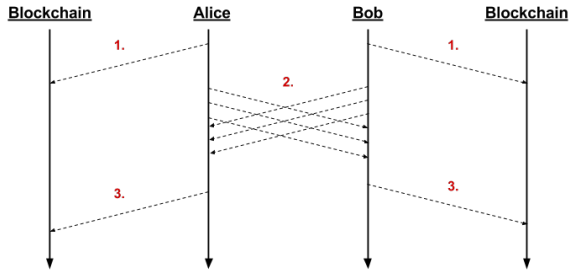

Fig. 1. A state channel settled in agreement.

1) Alice and Bob lock up some tokens by sending a transaction to the payment channel contract.
2) They can internally do transactions by signing receipts about new commitments and exchanging them. The receipts have to carry increasing sequence numbers and signatures.
3) Alice and Bob can release the bonded tokens by mutually agreeing (signing) on a resolution.

Obviously, there is a possibility that Alice and Bob will not agree in all situations. In figure 2 the dispute case is depicted:

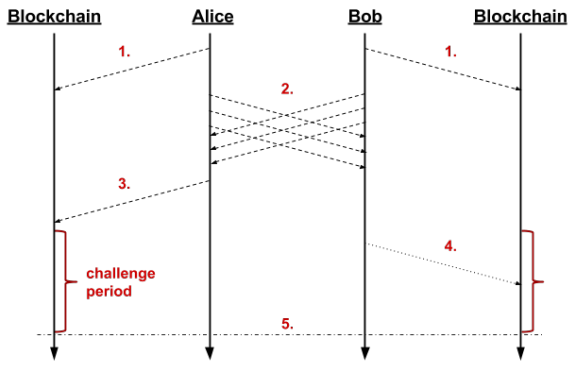

Fig. 2. A state channel settled after dispute.

1) Alice and Bob lock up some tokens by sending a transaction to the payment channel contract.
2) They can internally do transactions by signing receipts about new commitments and exchanging them. The receipts have to carry increasing sequence numbers and signatures.
3) If Alice or Bob stop cooperating, the latest receipt can always be submitted to the blockchain. This will trigger a challenge period.
4) During this period a newer receipt (higher sequence number) can be submitted.
5) After the challenge period expires, the bonded tokens are released to Alice and Bob by the ratio of the receipts with highest sequence numbers.

In effect, a state channel buffers the operations that would otherwise be written to the blockchain. Using a state channel decouples an application from the liveliness constraints of the blockchain without relaxing the security assumptions.

B. Multi-Party State Channels

An interaction of multiple participants could be modeled as a set of bilateral state channels from each player to each player. Yet, as the distribution of cash is not known beforehand, each channel would need to be funded with the full amount, resulting in a requirement of security deposits of $O(n\hat{2})$ tokens.

To be able to operate the channel with multiple parties and O(n) size of deposits, we define a commitment data structure as shown in figure 3.

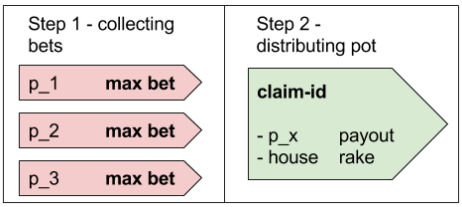

Fig. 3. Commitments and distribution

Commitments by participants carry a value, but do not have a specific recipient. Each higher value commitment by a participant replaces a lower one. When results of secure computation become available an oracle evaluates the data and issue a corresponding distribution receipt, reassigning the values from the participants' commitments to the winner(s) deducted from the multi-party computation.

Multiple rounds of commitments and distributions (hands played) can happen among the participants of the multi-party state channel. Figure 4 depicts that, as long as all parties agree on a final settlement, the protocol is similar to the two-party channel:

1) Alice, Bob and Charlie lock up some tokens under their control.
2) They can internally do transactions by signing receipts about new commitments and exchanging them. The receipts have to carry signatures, an id for the hand and increasing values.
3) They can have multiple rounds of commitment and distribution with a strictly increasing identifier.

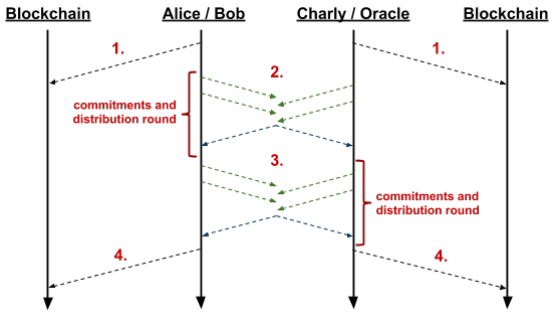

Fig. 4. A multi-party state channel settled in agreement.

4) Alice, Bob and Charlie mutually agree on a resolution. They sign the resolution after each validating the sum of all value transferred. The smart contract evaluates the resolution and releases the bonded tokens.

In a dispute case when no resolutions can be found, parties can start submitting receipts of commitments and distributions. Naturally, participants are incentivised to collect receipts in those rounds where they have a share in the distribution to submit them in the dispute case.

In the dispute period commitment receipts with higher values can overwrite those with lower value and distribution receipts with higher claimId overwrite those with lower one.

After the challenge period expires, all receipts are simply summed up on the chain and the bonded tokens are released according to the resulting balances.

C. P2P book keeping

A participant can sign a commitment of any size, participants have to check by looking at deposit and summing up all previous commitments and distributions to see if commitment doesn't exceed deposit.

D. Asynchronous participation

Rotation into a state channel requires no special protocol, once the deposit is placed, the participant can be interacted with.

Rotation of of a channel has the potential risk that a remaining party could be handed uncovered commitments by the leaving party after it has withdrawn its deposit already. To guard against this situation, we extend the on-chain state with the variables:

lastHandNetted: describes the id of the last hand at which the channel has been netted. this is also the hand at which the balance of the participant is valid.

exitHand: each participant maintains this flag, which is initialized with 0 and stays unchanged until the participant signals to leave. At that point the participant requests a leave receipt from the oracle, which identifies the latest hand id in the channel. This receipt is submited to the chain and sets the exit Hand of the participant.

lastHandNettingRequest: this is either lastHandNetted or a higher id, if the channel is in dispute.

Once the exitHand flag of a leaving participant is set on chain, other participants can clearly judge to accept or reject receipts based on their hand id. When, either through agreement, or dispute, lastHandNetted is increased to equal or beyond a participants exitHand id, the participant is payed out and removed from the channel.

III. Implementation in Solidity

https://github.com/acebusters/contracts/blob/master/contracts/Table.so

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

todo: take formula format from here: https://github.com/ethereum/research/blob/master/casper4/papers/casper_

source: https://en.wikipedia.org/wiki/Mental_poker

http://blog.cryptographyengineering.com/2012/04/poker-is-hard-especially-for.htm

IV. Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Appendix A
Proof of the First Zonklar Equation

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit

blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] U.S. Treasury Department, "Unlawful internet gambling enforcement act," 2011. [Online]. Available: http://www.ots.treas.gov/_files/422372.pdf

[2] A. Shamir, R. L. Rivest, and L. M. Adleman, "Mental poker," *The Mathematical Garden*, pp. 37–43, 1981. [Online]. Available: https://people.csail.mit.edu/rivest/ShamirRivestAdleman-MentalPoker.pdf

[3] Wikipedia, "Mental poker." [Online]. Available: https://en.wikipedia.org/wiki/Mental_poker

[4] ——, "Secure multi-party computation." [Online]. Available: https://en.wikipedia.org/wiki/Secure_multi-party_computation

[5] R. Cleve, "Limits on the security of coin flips when half the processors are faulty," in *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 1986, pp. 364–369. [Online]. Available: http://doi.acm.org/10.1145/12130.12168

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[7] I. Bentov and R. Kumaresan, "How to use bitcoin to design fair protocols," *IACR Cryptology ePrint Archive*, vol. 2014, p. 129, 2014.

[8] B. Bloomer and T. Nishimura, "Bitcoin micropayment channels," 2014. [Online]. Available: https://www.ischool.berkeley.edu/sites/default/files/student_projects/finalprojectreport_2.pdf

[9] I. Bentov, R. Kumaresan, and A. Miller, "Instantaneous decentralized poker." *CoRR*, vol. abs/1701.06726, 2017. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1701.html

[10] J. Coleman, "State channels," 2015. [Online]. Available: http://www.jeffcoleman.ca/state-channels/