A Framework for Decentralized Private Random State Generation and Maintenance for Multiplayer Gaming over Blockchain

Russell Harkanson, Carter Chiu, Yoohwan Kim, Ju-Yeon Jo
Department of Computer Science
University of Nevada, Las Vegas
Las Vegas, United States
{harkanso, chiu}@unlv.nevada.edu, {yoohwan.kim, juyeon.jo}@unlv.edu

***Abstract*—The transparency and immutability properties offered by the burgeoning field of blockchain technology make it an increasingly popular choice for applications across many domains. One such application is online gaming. Blockchain offers participants the ability to verify the fairness of games in a manner previously unattainable by the classical centralized approach. However, the introduction of blockchain incurs additional overhead and poses unique challenges necessary to overcome in order to be a viable alternative. The transparency blockchain provides opens potential avenues for collusion, particularly in multiplayer games, which must be addressed. Furthermore, the emulation of random state, a core component of online gaming, is rendered difficult in a decentralized context. No approach exists which manages random state for multiplayer gaming in a completely decentralized manner. We propose a novel approach toward this end, significantly extending the utility of blockchain technology to online gaming.**

***Index Terms*—decentralized applications, distributed ledger technology, smart contracts, blockchain, random state generation**

I. Introduction

Blockchain technology, and more generally distributed ledger technology (DLT), has driven significant innovation in recent years. The ability to record transactions in a transparent and immutable manner provides advantages in nearly every domain, from supply chain management [1] [2] [3], to financial and legal technology [4], to healthcare innovation [5]. In particular, the emergence of smart contracts enables computation on blockchain, providing the capability to perform and manage complex programmatic tasks in a verifiable fashion. In applications where security and transparency are vital requirements, blockchain and smart contract technology are increasingly seen as viable, if not essential, solutions.

An underutilized use-case for blockchain is online casino gaming. Trust is a critical aspect of online gaming as participants will be reluctant to wager money if they do not have confidence that a game is fair and opponents adhere to the rules. In a typical centralized solution, random number generation and other associated computation is handled by a casino server, and this computation is not revealed to the users. In such a setting, a player can never be entirely convinced of fairness. In other words, some level of assumed trust is mandatory. Blockchain, through smart contracts, is capable of lowering the barrier of implied trust by exposing the core functionality and inner-workings of the game. By fairly tracking the series of events and their resultant outcomes in a publicly verifiable manner, any individual actor could determine for themselves the legitimacy of the publicly-accepted outcome of any round of a game. The decentral and immutable nature of blockchain makes it resistant to manipulation by the casino or other bad-faith actors.

While the advantages of a decentralized solution to online gaming are clear, two primary challenges emerge:

A. Randomization

The first challenge is the task of random number generation and random state management. Blockchain is characterized by the absence of true pseudo-randomization. A blockchain is not able to draw a pseudo-random number like most computers due to the fact that results of a smart contract must be deterministic for repeatability, necessary for multiple nodes in the network to verify the results by running the same code from the same state.

However, even overcoming pseudo-random number generation is not enough to satisfy the requirements of an online casino. Functions such as coin flipping and dice rolling are simple randomization processes because they revolve entirely around a single, independent event. Unlike a coin flip, a random selection from a deck of cards depends upon previous selections from the same deck. This introduces a concept of state which must be properly maintained over time. Managing random state in the blockchain in a secure fashion is a non-trivial task.

B. Collusion-Avoidance

The second challenge is the mitigation of collusion. The increased transparency of blockchain is simultaneously a blessing and a curse: While this trait preserves the integrity of each exchange within the context of a classical ledger, the very same trait could be exploited by bad actors who could examine the allegedly private events of a poorly-developed card game. With all interactions on the blockchain made public, there

978-1-7281-7303-0/20/$31.00 ©2020 IEEE
DOI 10.1109/COMPSAC48688.2020.0-205

exists increased potential for colluding actors to manipulate the game, particularly in a multiplayer context. Specifically, a decentralized approach to random state management necessarily involves multiple participants, meaning that the approach must be done in such a way that actors cannot work individually or collaboratively to manipulate the random state in their favor, while still providing them an opportunity to play an active role in its management as a player of a game.

C. *A Stateful and Fair Framework*

We seek to develop a system which is capable of emulating the play of a variety of casino table card games on blockchain. This entails the secure generation and management of random state for a multiplayer environment in an entirely decentralized, verifiable manner. While some systems for decentralized online gaming exist, none satisfy all of the features we have enumerated to the extent that we propose. This paper outlines such an approach, greatly expanding the application space of blockchain technology to the domain of online casino gaming, yet described generally enough to be extendable and applicable for other purposes.

The remainder of our paper is structured as follows: In Section 2, we survey and discuss related work on decentralized ledger technology, secure online gaming, and the intersection thereof. Section 3 provides the goal of our framework and an overview of our approach, focusing on the composing entities and incentivization structure. Section 4 describes the interactions of entities within our proposed system, including the decentralized generation and maintenance of the random state along with the incentive structure necessary to sustain the contract. In Section 5, we discuss our implementation of the system and potential avenues for future work, and we conclude our paper in Section 6.

II. Related Work

A. *Blockchain*

Blockchain technology was first introduced by Satoshi Nakamoto in their seminal white paper [6] as the core underlying mechanism for their cryptocurrency, Bitcoin. It enables consensus between nodes in a decentralized fashion without the need for third party endorsement, thus eliminating any single point of failure. Bitcoin triggered a surge of interest in DLT, owing to its transparency and immutability properties. When coupled with smart contracts, an instrument for performing computation on a decentralized ledger formalized by Szabo [7], DLT has become an viable technology for a myriad of applications in computer science and its intersections with countless other disciplines. The first cryptocurrency to accommodate Turing-complete smart contracts was Ethereum [8]. As such, Ethereum became a popular platform for decentralized applications, including being the *de facto* platform for decentralized online gaming [9] [10] [11] [12].

B. *Random Number Generation*

Blockchain requires all computation to be reproducible in order for the network of nodes to verify the veracity of the ledger and reach consensus amongst all peers. This notion of determinism is of course at odds with the notion of randomness, and accordingly, random number generation is uniquely difficult in a DLT environment. Ethereum does not have intrinsic support for pseudorandom value generation [8] [13].

Nevertheless, the need frequently arises, particularly in gaming and wagering applications. A number of approaches exist in practice, with nearly all involving cryptographic hashing, as Wang et al. [14] identify pseudorandom number generation as a fundamental use case for hash algorithms in blockchain technology. However, the approaches vary wildly in their security. In a survey of attacks on smart contracts, Atzei et al. [15] categorize the process of generating randomness as a key source of vulnerability.

Consider the utilization of block hashes as a source of entropy, which Reutov [16] finds to be among the simplest and most frequently used methods. This approach is often done poorly, resulting in potential exploitation as occurred in a six-figure hack of the SmartBillions lottery [17]. And while properties of the blockchain such as block hashes leverage its ostensible unpredictability, a blockchain may be more malleable than one would presume. Studies from Bonneau et al. [18] and Pierrot and Wesolowski [19] find that attackers can manipulate random numbers generated from the properties of the blockchain with relatively limited computational and financial resources.

An alternative approach is the use of oracles. Oracles are web services which enable an on-chain smart contract access to external information. While effective, they represent a centralized solution that places trust in the service to provide fair pseudo-randomness, may introduce latency, and also presents a single point of failure [20] [21]. Adler et al. [21] introduce a decentralized blockchain oracle they term Astraea to address these weaknesses. They analyze Astraea to determine that correct incentivization exists through a Nash equilibrium, potentially opening the door to decentralized random number generation via oracle.

A far more secure class of pseudorandom number generators leverage commitment (commit-reveal) schemes, which were first formalized by Brassard et al. [22] and applied to blockchain by Andrychowicz et al. [23]. Sako and Iguchi [24] give an approach to random number generation for the game backgammon, in which commitment schemes are employed in conjunction with the transparency of blockchain. In this manner, participants can verify the fairness of the generated numbers after the conclusion of the game. A potential vulnerability exists in this scheme; if a player colludes with the server, they can predict future dice throws and leverage the information against the other player. This is addressed by Ehara and Tada [25], who incorporate the block nonce from a proof-of-work (PoW) based system as part of the generation scheme. The authors also present three axioms that typify correct approaches for transparent and distributed random number generation:

1) Random number generation should be dependent upon

seeds from the user and the server.

2) Neither users nor the server should be able to predict or manipulate the generated numbers.
3) Both users and the server should be able to verify the correctness of the generated numbers.

Incorporating a game-theoretic approach on top of a commitment scheme, Chatterjee et al. [26] present an alternative method for secure random number generation based upon the Nash equilibrium. They define a *random bit generation* game and prove its quasi-strong equilibrium. An implementation in Solidity on the Ethereum blockchain demonstrates a throughput of roughly 177 bits per second.

A handful of well-known online gaming applications involving random value generation exist in the Ethereum ecosystem. Perhaps the simplest is CoinFlip [9], which invites users to bet on the result of a single coin flip. Similarly is iDice [10], in which participants bet on the result of a simulated die roll. Both of these games interface with oracles in order to obtaining random values.

C. Random State Management

The notion of random state poses a much more complicated challenge in a transparent decentralized environment. This is necessary to simulate the function of a deck of cards, in which selections from the deck depend upon previous selections. Secure random state management in the context of blockchain appears to be a topic entirely absent from academia. However, there are a number of decentralized applications on Ethereum that offer this functionality. Edgeless [11] features a shuffled deck for the purposes of blackjack, a single-player game. From a technical standpoint, managing state in a multiplayer setting is an additional challenge, which CoinPoker [12] performs as a facilitator for games of poker. However, in the case of CoinPoker, a central server is still responsible for managing the state of the deck, reintroducing the single point of failure concern. No system currently exists which features full decentralization of the random state in a multiplayer setting.

III. Framework Overview

Our proposed framework for the generation and management of decentralized random state is presented within the context of a multiplayer online card game. At initial inception, the working title, HoldemChain, was assigned to the framework as it was originally applied to a game of Texas Hold'em. Though there are many styles of Hold'em, each with their own variations of rules, the game is generally a betting variant of Poker where players buy into a round at a time, seated at the table along with their opponents. The duty of *dealer*, signified by a special *dealer chip*, makes its way around the table in a clockwise fashion from game to game, granting each of the players the opportunity to deal. Cards are dealt in two rounds, starting from the player to the dealers left, and ending on the dealer themself. Depending on the variant, at least one of these cards are kept private to the receiving player. Players continue to bet on their chances of having a winning hand as the game proceeds as five publicly visible shared community cards get dealt between rounds of betting.

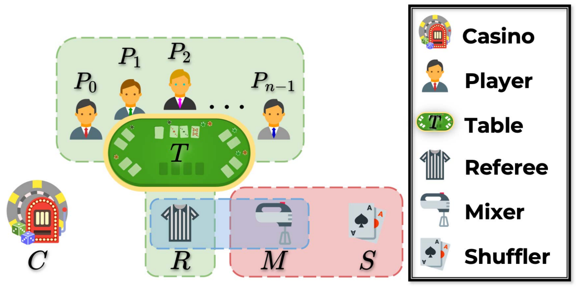

Fig. 1. Framework overview depicting associations of each entity.

A. Entities

To represent this game, along with the framework that we propose, our model employs six key entities. Fig.1 provides the graphical representations of each of these entities along with their associations.

1) **Casino** C – The casino will represent the smart contract residing on the blockchain that publicly maintains game and transaction history along with casino chip balances for each user, which is the token the contract provides.
2) **Player** P_i – A player is the role for an individual actor node that can play and bet their chips on a game. Each player maintains the state of their own drawn cards.
3) **Table** T – A table consists of a group of player nodes directly involved with the events of a game and sharing some common knowledge, along with the parameters and rules for the game it hosts. A table and its associated parameters may be requested from the casino by a prospective player before the game has begun.
4) **Referee** R – A referee is a member of the casino's network that does not bet on or play the hosted game, but instead keeps track of the game events as an unbiased, unaffiliated party.
5) **Mixer** M – A mixer is another unaffiliated individual that is not associated with the table but also never directly communicates with the table either. It is used as a middle-man between the referee and the shuffler to keep them private from one another in an onion routing fashion.
6) **Shuffler** S – A shuffler is an unaffiliated actor that maintains the state of the deck throughout the course of the game, without knowledge of the table that their deck is intended for or even the value of any individual card they hold.

B. Incentivization

All actor entities, P_i, R, M, S, are users of the smart contract network C. The latter three actors, collectively known as the *workers*, do not play any traditional role in the Hold'em game being played by the current players. Instead, actors R, M, S are *working* for the players at T as a crucial part of the structure of incentives that makes the whole framework possible.

The workers are users just like the players, and they once played the role of player in the past. They are *working* to clear their winnings from games they previously took part in. Whenever a player wins chips in a round, the value is accumulated in a separate variable that smart contract C keeps track of: uncleared winnings. To liberate these tokens, they must spend some time contributing to the network by working as a referee, mixer, or shuffler. The amount of work necessary, per chip, to clear their prior income should be open and left adjustable by the contract implementing the framework, taking into account factors such as the size of the userbase of C.

When a user needs to clear their previous winnings, they join a worker pool and become eligible for selection. If the network demands a specific class of worker, the market demand should incentivize users to select the role that is most scare by automatically scaling the amount token clearance they earn for that duty. This should naturally provide a balance in the pool of available workers.

Additionally, *work*, in context of this framework, is intended to be a completely passive experience for the end-user. Applications implementing the Application Binary Interface (ABI) of smart contract C should automatically attempt to clear prior winnings as fast as possible. This allows the user to freely join new games as their device assumes the role of a worker in the background.

IV. Entity Interaction

Over the course of game, the interactions between entities must be carefully crafted to assure an environment in which a random state can be held decentrally and in a manner that reasonably deters collusion from bad actors. The many moving parts of our framework are necessary to uphold these primary goals and broadly fall into the following four key phases before gameplay:

A. Creating a Table

To begin a game, a player is able to start a new table by requesting one from the casino. The player must first validate their random state, which is a fixed number of bytes r_0. This is achieved by a nonced hash commitment of r_0 to the chain,

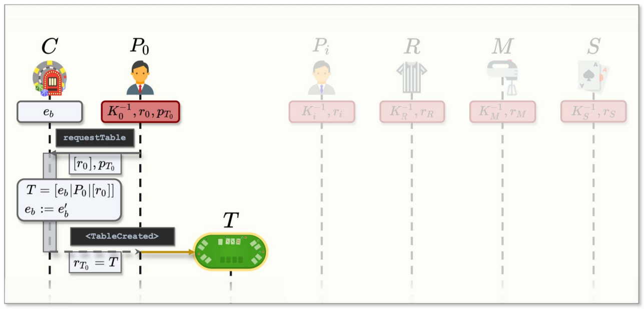

Fig. 2. Player requests a table from casino.

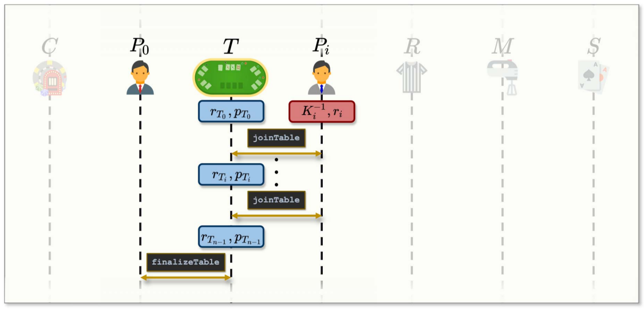

Fig. 3. Additional players join the new table.

represented as $[r_0]$. This will lock in their random seed value, and is done as a part of the request for the table along with table parameters p_{T_0}, as depicted in Fig.2.

Only after the table request is accepted, the contract provides a pseudo-random offset in order to avoid preselected values. This offset is a function of P_0's address and commitment along with the chain's global random entropy state e_b, itself a function of all user interactions from the casino network as a whole. This implies a dependency in the most-recently accepted block that was mined on the network hosting the contract. The mere act of using C's entropy implies an update to its value for the next user to request it, making the value difficult to pre-select for, assuming the form $e_b := e_b'$. The contract returns the result of this procedure T which becomes the new table's ID and initial random state T_0.

B. Players Join Table

With a valid table created and notarized by the smart contract, other players are free to join. The player-joining phase occurs off-chain, under private communication, visible only to the party as it assembles, as shown in Fig.3. Each new player P_i mutates the table state as they join. This is simply done by signing the previous state of the table with their blockchain private key K_i^{-1}, their Ethereum account key for example. This allows all other players at the table to verify the output value since the public key of every player is known. Thus commitments are chained in a deterministic fashion as players arrive, traceable to the initial table state T_0, which itself was the result of the initial player's commitment and the entropy of the contract. This ensures that the semi-final state of the table T_{n-1} is verifiable by any player at the table and future-provable at the end of the game when the necessary seeding values are revealed to claim winnings. When the table is finalized, the initial player P_0 signs the semi-final state of the table thus producing providing the true final table state T, completely verifiable by every seated player as they monitored the value evolve.

C. Referee Selection

A referee R is deterministically chosen based on the final table state T by selecting an address from the pool of available workers, shown in Fig.4. The ID of the table is not disclosed

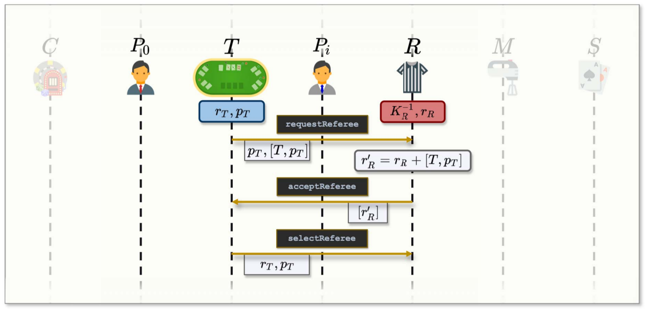

Fig. 4. The final table state is used to deterministically choose referee.

upon request, but only a committed value in case the user denies the work request so the table ID would not be leaked. The referee is also kept private from casino contract C to avoid any association. If declined, the table selects the next available address from the worker pool to offer the job. Upon a successful handshake of commitments after accepting the job, the table provides the referee with the current table state and parameters. The referee checks in to the contract, committing that they are working for a table, but keeping the table's ID private at the time. The players can observe this checkin by monitoring the chain, and are provided with another state offset to further reduce any attempts to control the state.

D. Mixer and Shuffler

The referee's offset state is used to deterministically choose a mixer in the same repeatable manner the table used to select the referee, reporting the hash committed address of the mixer to the table while keeping the mixer's actual address private. Once more, as Fig.5 portrays, this selection process is repeated allowing the mixer to find a shuffler. Like the referee but unlike the mixer, the shuffler also checks into the contract along with their hash commitment, also including the address of the mixer, used to validate their involvement. This commitment leaves the shuffler with a final pseudo-random state, influenced by the entire selection process, with a lineage tracing back to the initial table creation.

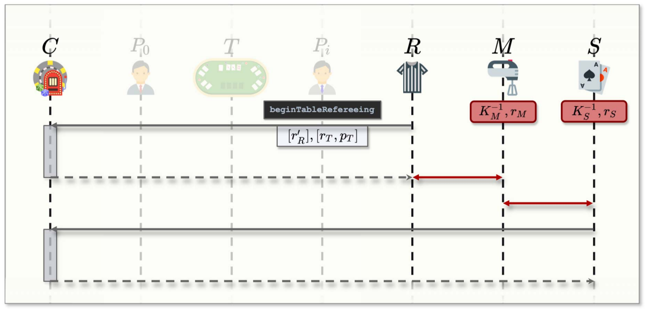

Fig. 5. The referee picks a mixer, and the mixer picks a shuffler.

E. Gameplay

The shuffler's state is used to seed the shuffled order of 52 indices, one for each card. The shuffler needs to hold on to the deck and pass a card on to the mixer when queried. Any pass between entities can be done by encrypting the card with the recipient's public key, especially when the referee needs to hand a private card to a single player at the table. The shuffler and even the mixer do not know the table they are working for so the chance of collusion is reduced. The information is further obfuscated from the shuffler since the card indices they hold will be offset one final time when the card arrives at the table since the table's state will remap the indices once more, resulting in their final value.

At this point, every actor has a committed a state value, at least to another actor, thus making the rest of the gameplay deterministic so long as the events are properly logged. After the completion of a game, the workers can reveal their secrets to confirm the game was played out as intended. This step is necessary for the workers to be credited their token clearances, as per their incentive. All players know the history of the initial table state, so any one of them, most likely the winner, can prove the results are valid given the reveals from the workers and the commitments that were made on-chain.

V. Development and Future Work

With the theoretical framework in place, we have started development of the framework with most of our efforts dedicated to generating the random state as described and using those values to select workers. Certain aspects of the framework are left tunable to taste, such as the credits for clearing the winnings of workers. With an active and large enough network of users, further testing could be performed through observation of the market to provide useful suggestions for efficient worker credit models.

We implemented our approach on the Ethereum platform, with smart contract development on Solidity 0.4.25. This decision was made owing to Ethereum's status as the *de facto* blockchain for online gaming applications. We utilized a Node.js web server, with the web.js library and MetaMask to facilitate communication between the interface and blockchain. Cryptography was managed using OpenSSL. Testing and analysis was performed first on a Ganache local blockchain and then on the Ropsten testnet.

VI. Conclusion

Online gaming is one of countless domains with the potential to benefit from blockchain solutions. The transparency afforded by decentralized ledger technology provides players with the capability to verify the fairness of their games by eliminating the potential for abuse present in existing centralized systems. This paper explored one such approach to multiplayer online gaming with decentralized random state management. We have shown it is possible to maintain a prolonged random state with dependencies on previous states over long enough periods to construct a multiplayer card game while maintaining complete decentralization, offering advantages not present in any existing system.

Acknowledgment

We would like to give additional thanks to Dr. Yoohwan Kim for his invaluable insight and guidance, without which this work would not have been possible.

References

[1] S. Madumidha, P. S. Ranjani, S. S. Varsinee, and P. S. Sundari, "Transparency and traceability: In food supply chain system using blockchain technology with internet of things," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, April 2019, pp. 983–987.

[2] S. Malik, V. Dedeoglu, S. S. Kanhere, and R. Jurdak, "Trustchain: Trust management in blockchain and iot supported supply chains," in *2019 IEEE International Conference on Blockchain (Blockchain)*, July 2019, pp. 184–193.

[3] B. M. A. L. Basnayake and C. Rajapakse, "A blockchain-based decentralized system to ensure the transparency of organic food supply chain," in *2019 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, March 2019, pp. 103–107.

[4] Z. Meng, T. Morizumi, S. Miyata, and H. Kinoshita, "Design scheme of copyright management system based on digital watermarking and blockchain," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 02, July 2018, pp. 359–364.

[5] G. Carter, H. Shahriar, and S. Sneha, "Blockchain-based interoperable electronic health record sharing framework," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, Jul 2019, pp. 452–457.

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

[7] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997. [Online]. Available: https://ojphi.org/ojs/index.php/fm/article/view/548

[8] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[9] Coinflip, "Coinflip," https://coinflip.4dapp.io, accessed on April 20, 2019.

[10] iDice, "idice.io: Ethereum casino gambling dice game," https://idice.io, accessed on April 20, 2019.

[11] Edgeless, "Edgeless," https://edgeless.io/home, accessed on April 20, 2019.

[12] CoinPoker, "Coinpoker: Play poker online using cryptocurrency," https://coinpoker.com, accessed on April 20, 2019.

[13] Y. Hirai, "Defining the ethereum virtual machine for interactive theorem provers," in *Financial Cryptography and Data Security*, M. Brenner, K. Rohloff, J. Bonneau, A. Miller, P. Y. Ryan, V. Teague, A. Bracciali, M. Sala, F. Pintore, and M. Jakobsson, Eds. Cham: Springer International Publishing, 2017, pp. 520–535.

[14] L. Wang, X. Shen, J. Li, J. Shao, and Y. Yang, "Cryptographic primitives in blockchains," *Journal of Network and Computer Applications*, vol. 127, pp. 43 – 58, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S108480451830362X

[15] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts sok," in *Proceedings of the 6th International Conference on Principles of Security and Trust - Volume 10204*. New York, NY, USA: Springer-Verlag New York, Inc., 2017, pp. 164–186. [Online]. Available: https://doi.org/10.1007/978-3-662-54455-6_8

[16] A. Reutov, "Predicting random numbers in ethereum smart contracts," https://blog.positive.com/predicting-random-numbers-in-ethereum-smart-contracts-e5358c6b8620, accessed on April 20, 2019.

[17] SmartBillions, "Smartbillions hackathon smart contract hacked with $120000!" https://medium.com/@SmartBillions/smartbillions-hackathon-smart-contract-hacked-with-120-000-b62a66b34268, accessed on April 20, 2019.

[18] J. Bonneau, J. Clark, and S. Goldfeder, "On bitcoin as a public randomness source," *IACR Cryptology ePrint Archive*, vol. 2015, p. 1015, 2015.

[19] C. Pierrot and B. Wesolowski, "Malleability of the blockchain's entropy," *Cryptography and Communications*, vol. 10, no. 1, pp. 211–233, Jan 2018. [Online]. Available: https://doi.org/10.1007/s12095-017-0264-3

[20] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, "The blockchain as a software connector," in *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, April 2016, pp. 182–191.

[21] J. Adler, R. Berryhill, A. G. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A decentralized blockchain oracle," *CoRR*, vol. abs/1808.00528, 2018. [Online]. Available: http://arxiv.org/abs/1808.00528

[22] G. Brassard, D. Chaum, and C. Crépeau, "Minimum disclosure proofs of knowledge," *J. Comput. Syst. Sci.*, vol. 37, no. 2, pp. 156–189, Oct. 1988. [Online]. Available: http://dx.doi.org/10.1016/0022-0000(88)90005-0

[23] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Secure multiparty computations on bitcoin," in *2014 IEEE Symposium on Security and Privacy*, May 2014, pp. 443–458.

[24] K. Sako and K. Iguchi, "Fair coin flipping for online games using blockchain," in *Proceedings of The 2017 Symposium on Cryptography and Information Security (SCIS2017)*, 2017, pp. 1F2–4.

[25] Y. Ehara and M. Tada, "How to generate transparent random numbers using blockchain," in *2018 International Symposium on Information Theory and Its Applications (ISITA)*, Oct 2018, pp. 169–173.

[26] K. Chatterjee, A. K. Goharshady, and A. Pourdamghani, "Probabilistic smart contracts: Secure randomness on the blockchain," *CoRR*, vol. abs/1902.07986, 2019. [Online]. Available: http://arxiv.org/abs/1902.07986