Gas-Efficient Decentralized Random Beacons

V.P. Abidha, Togzhan Barakbayeva, Zhuo Cai, Amir Goharshady

Gas-Efficient Decentralized Random Beacons

V.P. Abidha Togzhan Barakbayeva Zhuo Cai Amir Kafshdar Goharshady
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong, China
{abidhavp, tbarakbayeva, zcaiam, goharshady}@cse.ust.hk

***Abstract*—Decentralized random number generation is a widely-studied problem in the blockchain community and much attention has been paid to the so-called on-chain random beacons, i.e. smart contracts that generate randomness which can in turn be used in other contracts. Following the classical methodology of RANDAO, most on-chain beacons receive inputs from a large number n of participants and then aggregate them to compute a final random output. The aggregation is done in a manner that ensures the final output is uniformly random as long as at least one of the participants acts honestly. While being highly successful in providing security guarantees such as unpredictability and tamper-resistance, a major downside of these beacons is their cost. Since every participant has to call a function in the smart contract to provide their input, the total gas usage to generate a single random number is at least $\Omega(n)$.**

In this work, we propose a novel protocol that offloads most of the on-chain communication between the participants and the smart contract to an alternative off-chain communication with a dealer. This leads to a gas-efficient on-chain random beacon with only $O(1)$ gas usage per generated output. Crucially, our protocol is trustless and the dealer is unable to predict or tamper with the result. We maintain the same security guarantees as previous on-chain beacons, while significantly reducing the gas usage. We also show that our protocol is secure even if all but one of the participants, potentially including the dealer, are dishonest.

I. Introduction

RNG. A Random Number Generator (RNG), or random beacon, is an important component in many distributed protocols. Its applications range from efficient Byzantine consensus [1] to Proof-of-Stake (PoS) [2]–[4] and trusted setup of cryptographic protocols [5]. Along with the rapid growth of blockchains and decentralized finance applications [6]–[13], many on-chain RNG protocols have been proposed [14]–[24].

Smart Contracts and Gas. Ideally, DeFi applications which are usually implemented as smart contracts would want to access fresh random numbers as simply as calling a library function. There are many smart contracts that provide this functionality, most notably RANDAO [18]. These RNG smart contracts require a number of participants in the network to jointly contribute to new random numbers to achieve decentralization. They use different cryptographic techniques to ensure bias-resistance. This includes commitment schemes, publicly verifiable secret sharing (PVSS) [2], verifiable delay functions (VDF) [25]–[27], and homomorphic encryption (HE) [28]. Irrespective of the underlying cryptographic protocols, in all these contracts the participants join the RNG process by making transactions for which they are charged transaction fees in the form of gas [29]–[31]. As a result, either the participants have to pay the transaction fees themselves and thus few users would want to participate, undermining the decentralization, or the RNG service has to cover the transaction fees and thus the generation of each new random output becomes highly expensive and costs $\Omega(n)$ gas where n is the number of participants. In this work, we alleviate this issue by a novel protocol inspired by layer-2 solutions.

Off-chain/Layer-2. Many popular blockchains, such as Bitcoin and Ethereum, have a small throughput and high transaction fees. Therefore, there have been many proposals to use off-chain communication to replace on-chain transactions [32]. A quintessential example is that of payment channels such as the lightning network [33] in which a pair of users can use off-chain communication to make a large number of transactions between themselves. An on-chain transaction is required only to open or close the channel. There are many layer-2 solutions for Ethereum, as well, moving the contracts off the main chain and avoiding high gas fees [34]–[36].

Insufficiency of Rollups. Optimistic rollups are a widely-adopted layer-2 solution to increase the throughput of layer-1 blockchains and reduce/avoid gas fees [37] by aggregating a batch of offchain transactions into one blockchain state update. The rollup operator might publish an invalid final blockchain state. However, other participants can challenge the claims of rollup operators and claim the deposit from operators. We note that decentralized RNG protocols have stronger security requirements than optimistic rollups. Specifically, it is important that every participant's value x_i must be included in the aggregation function, otherwise the RNG result is unreliable and might have been tampered with by the operator. In optimistic rollups, ignoring transactions is not a major issue because the victimized user can reach out to another rollup operator, much in the same way that if a miner refuses to include a transaction in a layer-1 block, another miner will eventually pick it up. However, in our setting, there should be strong guarantees that the operator is not excluding any contributions from the participants. Thus, even with rollups, designing gas-efficient RNG protocols leveraging off-chain communication is an interesting problem.

Purely Off-Chain RNG. It is possible to implement a random beacon based on an independent distributed system, without an underlying blockchain [19]. However, in order to use the result of this kind of beacon in a smart contract, one would need to create a bridge between the beacon and the blockchain, usually in the form of a centralized oracle. Using an oracle would violate both trustlessness and decentralization.

Motivation for Gas-efficient RNG. Truly decentralized RNG is possible only if we can incentivize many participants to join. With current methods, generating one fresh random number

The research was partially supported by the Hong Kong Research Grants Council ECS Project Number 26208122. T. Barakbayeva and Z. Cai were supported by the Hong Kong PhD Fellowship Scheme (HKPFS). Authors are ordered alphabetically.

takes $\Omega(n)$ units of gas, where n is the number of participants. To incentivize participation, we will have to at least cover the participants' gas fees. Thus, current methods have a cost of $\Omega(n)$ per generated random number. A gas-efficient RNG protocol that moves most of the communication off-chain can avoid this unnecessary gas usage and thus make decentralized RNG much more affordable for the end-users.

Our Contribution. Inspired by off-chain and layer-2 techniques such as optimistic rollups, we propose to offload most of the transactions of a classical RNG smart contract to off-chain communication. Our contributions are as follows:

- We present a novel RNG protocol that, while remaining trustless, moves most of the messaging off-chain.
- We show that our protocol satisfies all the desired properties of an RNG smart contract, such as tamper-resistance, unpredictability and liveness.
- As in previous random beacon smart contracts, our approach's output is guaranteed to be uniformly distributed as long as there is at least one honest participant. Thus, we are secure against any coalition of all but one of the participants, even if the coalition includes the dealer.
- In contrast to previous approaches, to generate a fresh random number our protocol only consumes constant $O(1)$ gas, whereas previous methods require $\Omega(n)$ units of gas, where n is the number of participants.

Problem Setting. In this work, we present a smart contract R that serves as a decentralized random beacon and performs RNG. We denote the number of participants, i.e. nodes willing to take part in the RNG, by n. We consider a multi-round protocol consisting of many sessions/rounds. Each session generates a fresh random number. Each participant registers in R and can contribute to every session until she withdraws. Participants should also pay a deposit at registration. Registration and withdrawal transactions are each sent only once for each participant. Therefore, their gas cost is negligible when amortized over the many sessions to which this participant contributes. Thus, we only consider the cost of one session in our analysis.

RNG with VDFs. Following many previous protocols based on VDFs [38], each participant i should choose a random value x_i. We can compute $r = Delay(Combine(x_1, x_2, \ldots, x_n))$, where $Delay$ is a pre-defined verifiable delay function (VDF), ensuring that the result is unpredictable, and $Combine$ can be any operation that combines the participant's inputs. As we will see, our approach uses a Merkle tree [39] to combine the values.

II. Our Protocol

Our protocol considers two types of users: a dealer and n participants. We assume that the participants have a secure and authenticated channel that can be used to send messages to the dealer. This is a standard assumption in many blockchain protocols that combine on-chain and off-chain communication and can easily be realized in the real-world by using any standard secure internet-based communication method. We also assume that the dealer can announce messages to every participant, either using the same channel, or on a public bulletin board which is visible to everyone. In practice, one might like to add a new type of user, a client, who pays for the costs of the random number generation and the rewards that are provided to the participants. Below, we assume these costs are borne by the dealer, but it is easy to assign them to a separate entity as needed. We first start by explaining our aggregation method and registration procedure and then present the main protocol.

Aggregation Method. Suppose each participant i has provided the input x_i to the protocol. We create a complete binary Merkle tree T with n leaves, all at the same depth, where the i-th leaf contains x_i. We then define $Combine(x_1, \ldots, x_n) = root(T)$ to be the root hash of this Merkle tree. Since T is a complete binary tree, the path from its root to the i-th leaf is uniquely determined by the binary representation of i and has length $O(\log n)$. Finally, as is standard, we apply a fixed verifiable delay function $Delay$ to obtain our final random number $r := Delay(Combine(x_1, \ldots, x_n))$.

Initialization. Our protocol is implemented as a smart contract that supports many rounds of RNG. The dealer deploys the contract on the blockchain and also sets the following values:

- The deposit d that each participant should put down to take part in RNG. This deposit is used to penalize the participant in case of dishonest behavior.
- The cost d^* of challenging a commitment. The use of d^* will become apparent further below.
- The time limits $t_1, t_2, \ldots, t_8$ for each of the steps below. Specifically, each step i can start only after time t_{i-1} and must end by time t_i. The smart contract functions mentioned in each step below enforce these time requirements. So, one step's functions are not callable when the contract is in another step. In practice, each t_i can be a timestamp or a block number.

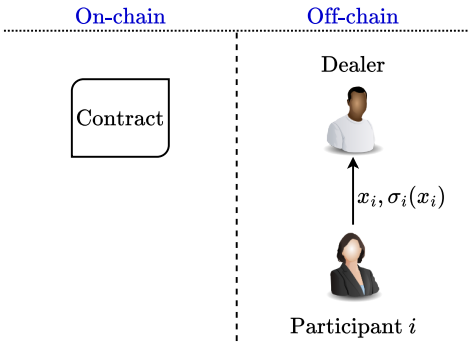
Registration. Our smart contract is an open protocol that allows anyone on the blockchain to sign up as a participant by calling its `register()` function and paying a deposit of at least d. The participant remains active as long as her remaining deposit is at least d. A participant can withdraw from the smart contract in between sessions. To do so, she can call the `withdraw()` function. The smart contract records her intention to withdraw and allows her to receive her deposit and any rewards she has accumulated at the end of the current session, or immediately if no session is in progress.

Session Creation. If there is no active session in progress, the dealer can create a new RNG session by calling `new_session()` and paying an amount ρ to the contract. ρ is the reward of the current session and will be divided among participants who take part in the session. Additionally, it is possible to set a fixed/minimum amount for ρ in the initialization phase. The dealer also pays an additional deposit d' to the contract at this point.

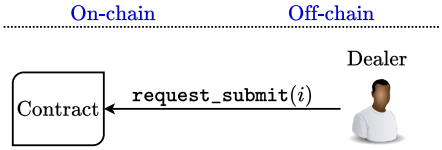
Details of the Protocol. We are now ready to provide a complete description of our protocol. In each session, our protocol has the following steps:

Step 1: Off-chain Submission. Each participant i sends her value x_i to the dealer off-chain. This x_i is her contribution to the RNG. We re-emphasize that this is using a secure and authenticated off-chain channel, meaning that the dealer not

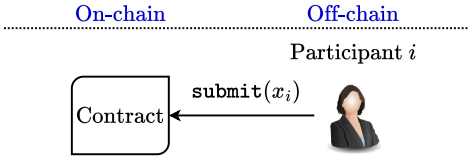
only receives x_i but also the participant's signature $\sigma_i(x_i)$ on x_i[†].



Step 2: On-chain Submission Request. If the dealer has not received the value x_i of participant i, he calls the smart contract function `request_submit`(i). It freezes i's deposit until she responds in the following step. It also freezes d^* units of the dealer's deposit. If the dealer does not call `request_submit`(i), this is interpreted as his implicit agreement that he has received x_i off-chain.



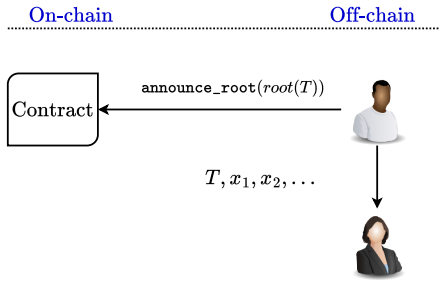
Step 3: On-chain Submission. Each participant i who was asked to submit on-chain in the previous step has to call the smart contract function `submit`(x_i). If i fails to do so by the deadline of this step, the protocol assumes $x_i = 0$ in the future steps and participant i's deposit is burned, while the dealer's deposit is unfrozen. Otherwise, the contract computes the total gas g that participant i has had to pay for the call to `submit`(x_i) and deducts $\min\{d^*, g/2\}$ from the dealer's deposit and pays it to i. In other words, the participant and dealer share the gas costs.



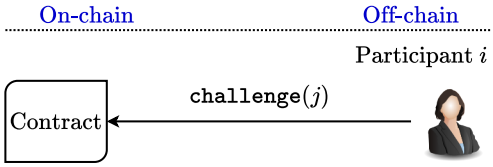
Incentives. We note that the steps above strongly incentivize the dealer and all participants to be honest and handle the values off-chain. It is in every participant's best interest to submit off-chain in Step 1, hence avoiding a costly (in terms of gas) on-chain submitting. It is in the dealer's best interest to require on-chain submitting in Step 2 for any participant who has failed to submit off-chain since he would otherwise be challenged in the following steps and loses his own deposit. Moreover, there is no incentive for the dealer to make spurious on-chain submit requests when participant i actually submitted a value x_i, since the dealer has to pay part of the gas fees.

[†]To guard against signature reuse attacks, we assume that the signatures contain a timestamp as well and is $\sigma_i(x_i, \texttt{timestamp})$.

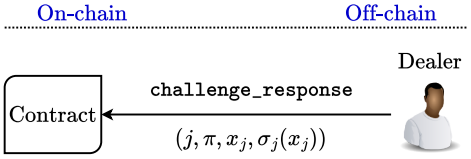
Step 4: Merkle Tree Announcement. The dealer creates a Merkle tree T with n leaves with the i-th leaf containing x_i, i.e. the contribution of participant i. He publishes T and all x_i values off-chain to all participants. This ensures that everyone can check the validity of T. Finally, he calls the smart contract function `announce_root`$(root(T))$ where $root(T)$ is the root hash of T, and will be recorded by the contract.



Step 5: On-chain Challenges. Any participant i can challenge the dealer to prove the validity the entry in the j-th leaf of T by calling `challenge`(j). This freezes a portion d^* of the challenging participant's deposit.



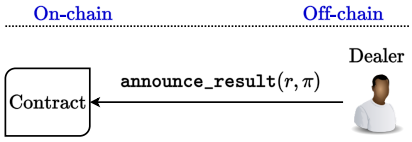
Step 6: On-chain Responses. The dealer has to respond to every challenge issued in Step 5 by calling the smart contract function `challenge_response`$(j, \pi, x_j, \sigma_j(x_j))$. Here, j is the leaf index, π is a Merkle proof consisting of the path from the root to the j-th leaf of the Merkle tree T, $\sigma_j(x_j)$ is participant j's signature proving that he had submitted x_j. The contract checks the Merkle proof and the signature.



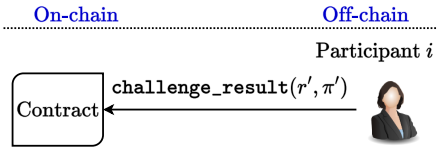
If the dealer fails to call this function in time or to provide valid values that pass the contract's checks, his deposit is confiscated and divided among the participants and the protocol ends. Otherwise, if he successfully handles the challenge, the contract computes the amount g of gas fees used in responding to the challenge and deducts $\min\{d^*, g/2\}$ from the challenging participant's deposit, paying it to the dealer and unfreezing the rest.

Incentives. It is the best response for the dealer to publish T (off-chain) that include the correct values of all participants, and announce its root. If the published tree does not contain a value x_j or modifies x_j, the dealer loses his deposit if some participant i challenges j, or gains nothing if no one challenges the j-th leaf because our RNG output uses VDF as explained in the next step.

Step 7: Verifiable Delay Function. The dealer computes $Delay(root(T)) = Delay(Combine(x_1, \ldots, x_n))$. The evaluation of a VDF leads to a result r and a proof of evaluation π. The dealer calls the function `announce_result`(r, π) of the smart contract. r is the generated random number and our RNG output. However, it is not yet finalized. The smart contract only records r and π but does *not* verify them at this stage.



Step 8: VDF Challenge. Given that $root(T)$ is publicly known, every participant can evaluate the VDF and compute (r, π) on her own machine. If a participant realizes that the dealer has cheated in the previous step and announced the wrong value of r, the participant can then call the smart contract function `challenge_result`(r', π'), providing the correct result r' and VDF evaluation proof π' to the contract. At this point, the contract verifies both claims by running the VDF verification algorithm on both (r, π) and (r', π'). The dishonest party, be it the participant or the dealer, is punished by having their deposit confiscated and paid to the other party. If no challenge is made in this step, or if all challenges are unsuccessful, the dealer can receive his deposit d' from the contract.



Incentives. If the dealer cheats in Step 7, every rational participant has an incentive to challenge him in Step 8 and win his deposit. Thus, the rational dealer has no incentive to cheat in the first place.

Participation Rewards. Any participant who completes a session (until the end of Step 8) and whose deposit is not confiscated, would be entitled to an equal share of the reward ρ of the session. All rewards will be paid to her when she later withdraws from the contract by calling `withdraw`().

III. Analysis

Desired Properties. Much like previous methods such as RANDAO [18], our protocol ensures the following desired properties:

- **Game-theoretic Guarantees of Honesty:** In our protocol, every party is strictly incentivized to act honestly in following the steps and to keep the communication off-chain. The incentives were already covered in the previous section. Thus, rational parties will follow the protocol.
- **Bias-resistance**: In each session, no party can manipulate the random output if he controls at most $n - 1$ participants. The Dealer cannot manipulate the random output either, even if he registers as participants or bribes other participants, as long as there exists one participant acting honestly. This is because all values x_i contribute to $root(T)$ which in turn decides $r := Delay(root(T))$. Even if the dealer wants to bias the output by carefully choosing his value after seeing other values, it is computationally infeasible to find a value such that $root(T)$ is in a negligibly small set where he can pre-evaluate the VDF. Thus, they cannot strategically manipulate the output.
- **Liveness**: In each session, no party that controls at most $n - 1$ participants can prevent the protocol from proceeding through each step and outputting a tamper-proof random number. The only case where the protocol ends without an output is when the dealer is caught cheating. However, this does not happen in practice if the dealer is honest since his cheating would cost him his deposit.
- **Unpredictability**: In each session, no party can know the random output r until after the x_i values are submitted and the VDF is evaluated. As long as the VDF security parameter and the time limits t_i are chosen suitably, the VDF evaluation cannot be completed before t_6. Thus, the VDF results are only known in Step 7.
- **Profitability**: Honest participants are guaranteed to make profits in each session. The dealer can also make a profit from customers by providing the RNG as a service.

Times and Deposits. In the initialization phase, the dealer has to decide on values for the time limits $t_1, \ldots, t_8$ and the deposits d and d^*. When starting a round, he also puts down his own deposit d'. It is possible to hard-code constraints on the d' already at the initialization phase. In setting these values, the dealer has to ensure the following:

- The deadline of each step should allow for sufficient time for all participants to make the function calls that are potentially needed in that step.
- Evaluating the verifiable delay function $Delay()$ should take strictly more than t_6 time. In other words, when the participants start submitting their values at the beginning of Step 1, no one should be able to compute the VDF until Step 6. On the other hand, t_7 should be large enough to allow the dealer and participants to evaluate the VDF in Step 7.
- The deposits should be large enough to (i) cover the gas fees that have to be reimbursed in case of unsuccessful challenges, and (ii) provide practical deterrence. Specifically, since answering the challenges in Steps 3 and 6 require the verification of a Merkle proof that consumes $\Theta(\lg n)$ gas, we must have $d^* \in \Omega(\lg n)$ and $d' \in \Omega(n \cdot \lg n)$.

Gas Usage. The major selling point and contribution of our protocol is that it significantly reduces the gas usage required for one round of RNG. As long as all parties are rational, they will follow the protocol honestly. The incentives for this were outlined in the previous section. This means that in each round the participants only communicate with the dealer off-chain and have to pay no gas fees. The dealer also pays only $O(1)$ in gas fees due to his calls to the contract functions.

References

[1] M. J. Fischer, N. A. Lynch, and M. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, pp. 374–382, 1985.

[2] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *CRYPTO*, 2017, pp. 357–388.

[3] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *CRYPTO*, 2018, pp. 66–98.

[4] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine agreements for cryptocurrencies," in *SOSP*, 2017, pp. 51–68.

[5] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *ASIACRYPT*, 2010, pp. 177–194.

[6] K. Arshi and A. K. Goharshady, "Congesting Ethereum after EIP-1559," in *IEEE ICBC*, 2024.

[7] M. A. Meybodi, A. K. Goharshady, M. R. Hooshmandasl, and A. Shakiba, "Optimal mining: Maximizing Bitcoin miners' revenues from transaction fees," in *IEEE Blockchain*, 2022, pp. 266–273.

[8] A. K. Goharshady, "Irrationality, extortion, or trusted third-parties: Why it is impossible to buy and sell physical goods securely on the blockchain," in *IEEE Blockchain*, 2021, pp. 73–81.

[9] K. Chatterjee, A. K. Goharshady, and A. Pourdamghani, "Probabilistic smart contracts: Secure randomness on the blockchain," in *IEEE ICBC*, 2019, pp. 403–412.

[10] ——, "Hybrid mining: exploiting blockchain's computational power for distributed problem solving," in *SAC*, 2019, pp. 374–381.

[11] K. Chatterjee, A. K. Goharshady, and E. K. Goharshady, "The treewidth of smart contracts," in *SAC*, 2019, pp. 400–408.

[12] K. Chatterjee, A. K. Goharshady, and Y. Velner, "Quantitative analysis of smart contracts," in *ESOP*, 2018, pp. 739–767.

[13] K. Chatterjee, A. K. Goharshady, R. Ibsen-Jensen, and Y. Velner, "Ergodic mean-payoff games for the analysis of attacks in crypto-currencies," in *CONCUR*, 2018, pp. 11:1–11:17.

[14] M. Raikwar and D. Gligoroski, "SoK: Decentralized randomness beacon protocols," in *ACISP*, 2022, pp. 420–446.

[15] K. Choi, A. Manoj, and J. Bonneau, "SoK: Distributed randomness beacons," in *S&P*, 2023, pp. 75–92.

[16] G. Wang and M. Nixon, "RandChain: Practical scalable decentralized randomness attested by blockchain," in *IEEE Blockchain*, 2020, pp. 442–449.

[17] M. Krasnoselskii, G. Melnikov, and Y. Yanovich, "No-Dealer: Byzantine fault-tolerant random number generator," in *INFOCOM*, 2020, pp. 568–573.

[18] "RANDAO: A DAO working as RNG of Ethereum," 2019. [Online]. Available: https://github.com/randao/randao

[19] P. Schindler, A. Judmayer, N. Stifter, and E. R. Weippl, "HydRand: Efficient continuous distributed randomness," in *S&P*, 2020, pp. 73–89.

[20] Z. Cai and A. K. Goharshady, "Trustless and bias-resistant game-theoretic distributed randomness," in *IEEE ICBC*, 2023.

[21] ——, "Game-theoretic randomness for proof-of-stake," in *MARBLE*, 2023, pp. 28–47.

[22] J. Ballweg, Z. Cai, and A. K. Goharshady, "PureLottery: Fair leader election without decentralized random number generation," in *IEEE Blockchain*, 2023, pp. 273–280.

[23] P. Fatemi and A. K. Goharshady, "Secure and decentralized generation of secret random numbers on the blockchain," in *BCCA*. IEEE, 2023, pp. 511–517.

[24] T. Barakbayeva, Z. Cai, and A. K. Goharshady, "SRNG: An efficient decentralized approach for secret random number generation," in *IEEE ICBC*, 2024.

[25] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *CRYPTO*, 2018, pp. 757–788.

[26] N. Ephraim, C. Freitag, I. Komargodski, and R. Pass, "Continuous verifiable delay functions," in *EUROCRYPT*, 2020, pp. 125–154.

[27] B. Wesolowski, "Efficient verifiable delay functions," in *EUROCRYPT*, 2019, pp. 379–407.

[28] T. Nguyen-Van, T. Nguyen-Anh, T. Le, M. Nguyen-Ho, T. Nguyen-Van, N. Le, and K. Nguyen-An, "Scalable distributed random number generation based on homomorphic encryption," in *IEEE Blockchain*, 2019, pp. 572–579.

[29] S. Farokhnia and A. K. Goharshady, "Alleviating high gas costs by secure and trustless off-chain execution of smart contracts," in *SAC*, 2023, pp. 258–261.

[30] ——, "Reducing the gas usage of Ethereum smart contracts without a sidechain," in *IEEE ICBC*, 2023, pp. 1–3.

[31] Z. Cai, S. Farokhnia, A. K. Goharshady, and S. Hitarth, "Asparagus: Automated synthesis of parametric gas upper-bounds for smart contracts," *OOPSLA*, pp. 882–911, 2023.

[32] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Off the chain transactions," p. 360, 2019.

[33] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[34] C. Sguanci, R. Spatafora, and A. M. Vergani, "Layer 2 blockchain scaling: A survey," *arXiv preprint arXiv:2107.10881*, 2021.

[35] L. Bousfield, R. Bousfield, C. Buckland, B. Burgess, J. Colvin, E. W. Felten, S. Goldfeder, D. Goldman, B. Huddleston, H. Kalodner *et al.*, "Arbitrum nitro: A second-generation optimistic rollup," 2018.

[36] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, 2022.

[37] A. Asgaonkar, *Scaling Blockchains and the Case for Ethereum*, 2022, pp. 197–213.

[38] A. K. Lenstra and B. Wesolowski, "A random zoo: sloth, unicorn, and trx," *IACR Cryptol. ePrint Arch.*, p. 366, 2015.

[39] R. C. Merkle, "Method of providing digital signatures," 1982, US Patent 4,309,569.