Improvement on Bitcoin's Verifiable Public Randomness with Semi-Trusted Delegates

Habib Yajam, Elnaz Ebadi, MohammadTaghi Badakhshan, and MohammadAli Akhaee
School of Electrical and Computer Engineering, Engineering Faculty,
The University of Tehran
Email: {elnaz.ebadi, mtbadakhshan, habib.yajam, akhaee}@ut.ac.ir

***Abstract*—Publicly Verifiable Randomness has many useful and interesting applications in e-voting, distributed cryptography, secure multiparty computation, lotteries, sports seeding and many more. While the output of a Publicly Verifiable Randomness is provably unpredictable for a strong adversary, it is expected that its closeness to a uniform distribution is verifiable for any party. The inherent randomness in public blockchains such as Bitcoin has been the topic of several research papers and been used in lotteries and other multi-party protocols. However, recently it has been argued that an adversary can have a significant impact on the probability distribution of the output with much lower financial cost. Here, in this paper, we propose a new scheme based on the properties of Verifiable Secret Sharing protocols and Threshold Cryptosystems, that uses semi-trusted third parties to improve the security of Verifiable Public Randomness from any public blockchain. We argue that a successful attack against our scheme to impose a bias on a single bit of the output randomness requires not only a significant financial cost but also a corruption of more than k out of n trusted delegates.**

***Index Terms*—Blockchain, Randomness, Security Protocol, Public Verifiable Randomness, Bit Commitment, Bitcoin Beacon, Multi-Party Computation, Secret Sharing, Threshold Cryptosystem**

I. Introduction

Randomness has an important role in many applications such as voting, gaming, lotteries, raffles, surveys, and gambling. It is also a critical part of many cryptographic algorithms and protocols, attempting to meet security goals such as confidentiality, anonymity, and authentication [1]. Achieving some source of randomness that produces random outputs with desired properties –such as uniform distribution– in many aspects is a challenge.

While random generation has its own solutions and tools such as hardware true random generators, in many scenarios, we need a source of randomness that can be trusted by various parties with opposing interests. Examples could be sports seedings, lotteries, and cryptographic multiparty computations. Their fairness is strictly dependent on the properties of the randomness source, and the trust that all participants have in these properties. Because there could be a conflict of interest, some participants may try to alter the random source. This illustrates the need to verify the correctness of randomness publicly. In this case, we require a Publicly Verifiable Randomness Source. This randomness source must have at least three security requirements named consistency, autonomy, and unpredictability. Roughly speaking, consistency means that all participants receive the same random value, and they can trust others receive the same. Autonomy means that manipulation of the random value is infeasible. Finally, Unpredictability means that none of the participants is able to guess the output before it is accessible to all others [2]. Thus, in a verifiable random source, we can publish a proof of outputs correctness publicly and all participants can verify that the output has the desired properties, after it has been run.

978-1-5386-8274-6/18/$31.00 2018 IEEE

There have been many types of research to propose schemes for Publicly Verifiable Random Source. Deriving random bit from financial data in stock markets was one of the results of the efforts that have been made in this field. However, it has been argued that an adversary can change the announced stock prices and disturb the extracting random bits [3]. Additionally, there is the possibility of violating unpredictability feature which of the public randomness source by utilizing Artificial Intelligence tools. Another prominent example is NIST randomness beacon which uses two independent commercially available randomness sources. This approach has a fundamental difference compared to others in the way that it relies on a trusted party [4]. Presence of a trusted party can discourage its usage in many scenarios, especially after the scandals about NISTs usage of kleptography in its standards.

In order to have a Publicly Verifiable Randomness Source without relying on a trusted third party, an immediate idea is the usage of Distributed Ledger Technology [5]. A public randomness source has many similarities with Bitcoins idea of a public ledger without trusted parties.

In this paper, we will introduce a new approach to achieve a random number utilizing the distributed ledger technology. In the proposed approach, we obtain a publicly verifiable randomness source via a collaboration of semi-trusted delegates combined over blockchain. By semi-trusted delegates, we mean a group of delegates that at most $k-1$ out of total n members are malicious.

II. Blockchain as a Public Randomness Source

As mentioned in the previous section, implementing a protocol which uses blockchain to provide public randomness source has been considered for many applications such as broadcast protocols, distributed peer to peer cryptography, and multiparty computations [6]. As this framework is supposed to

work peer to peer, creating protocols independent from trusted parties could be possible.

Cryptocurrencies such as Bitcoin[7], Zcash[8], and Monero[9], benefit from Proof of Work (PoW) consensus algorithms to resist attacks on the blockchain. In these cryptocurrencies, miners should try to find a nonce for which the value of the hash of the block containing that nonce is a number less than the value of a specific value determined by the network[7]. It is known that this value has a relatively high randomness in it. Bitcoin blocks have other randomness sources additional to the nonce. Such as the Merkle tree for transactions, previous block hash, and so on. As mentioned in [10], since The Merkle root is the hash of hundreds of Bitcoin transactions and nearly all of them contain ECDSA signatures, the amount of its entropy is likely thousands of bits. The hash function is a one-way and preimage resistant function so the only feasible way to check block validity is by computing the hash itself. Because of these characteristics of the ideal hash functions, if one wants the hash value of a new block to meet some conditions additional to being less than network target difficulty, the miner who mines that block should significantly cover more computational costs. Another important property of hash functions is the avalanche effect, consequently, the miner cannot reach similar hash values with similar nonces. Therefore, making any bias on the randomness of hashes equals to solving a much more difficult puzzle[10].

This similarly holds true for the Proof of Stake (PoS) blockchains, which means that we can also profit from the same random source that was available in Proof of Work (PoW) consensus blockchains such as Bitcoin. Most PoS blockchains are categorized into two groups. In the first group, the difficulty of hash-proof is solely determined by the network target difficulty and stakes of the miners, an example is Nxt coin [11]. In the second group, the age of the stake also matters which means that a miner who has a higher value of stacks that are older has a higher chance in generating a new block, PeerCoin is an example of this types of cryptocurrencies. Apparently, because of the nonce in each block and other unpredictable parts, both of these PoS Blockchain groups have significant randomness in each generated block that can be used for our purpose. However, each blockchain could have a different amount of randomness that depends on the hash difficulty and other network properties. This amount of randomness in each block could be measured as stated in [4]. In the way that in a PoW-based cryptocurrency demand for expensive mining tools and consumption of considerable electrical energy for protecting the network from 51 percent attack, in PoS based ones, the cost that a block generator must pay is keeping their stake high enough to be able to mine. Thus, the financial cost for an adversary to have more than 50% of the network's stake is high enough to thwart most attackers. Similar condition holds for our case where the financial costs for the adversary to cause a significant bias on the generated random number is a barrier that undermines many attackers incentive to do so.

As reported in [10], 32 near uniform bits can be extracted from 68 min-entropy bits of random strings made during mining every 10 minutes. This potential, motivated researchers to use blockchain as a public randomness source. A few pieces of research have worked on how the adversaries with different amounts of mining power can increase the statistic distance from uniform random distribution in the bits extracted from blocks. It has been proven that it is impossible without a significant financial cost which could be unreasonable in many scenarios. Besides, any adversary who intends to threaten the security of this protocols could attack the underlying blockchain itself [4].

A. Additional Randomness Input from Semi-Trusted Delegates

As stated in [4], the average cost of manipulating one bit in the block's randomness by corrupting all miners in the network is at least half of the block's reward around which is 6.25฿ or 50,000$ at the time of writing. However, in another research [12] it has been shown that manipulating probability distribution of one bit of output randomness takes much lower average cost. Thus, such an attack is reasonable where the benefit of exploiting randomness is higher than this cost. A countermeasure for thwarting attacks in these scenarios is using more blocks to extract randomness from, or using Semi-trusted delegates.

In the remaining of this section, we discuss how a publicly verifiable randomness source via a collaboration of semi-trusted delegates over blockchain can be obtained. The protocol generates a random value in two stages: the commitment stage and the reveal stage.

In the commitment stage, each delegate generates a private random nonce and calculates its hash, then he signs it and sends the result to the blockchain. This stage guarantees that no opponent delegate can change his nonce later when she knows others inputs. On the other hand, despite the order of delegates commitment, no one can choose the random nonce based on the other nonces. This is because that he can not guess the other participants random nonce from its hashed value. As a consequence, there exists a negligible chance to cause bias in the distribution of random output in this stage.

In reveal stage, each delegate sends his private nonce to the blockchain and everyone can validate whether the revealed nonce is exactly the nonce that he has chosen in the previous stage. This validation can be done by calculating its hash and comparing the result to its corresponding value that the delegate has committed in the previous stage. At the end of this stage and when all participants revealed their random nonce, the final random output can be generated by calculating the XOR of all revealed nonces.

The output value is a publicly verifiable random number even if only one honest delegate participates in the protocol and privately creates a pseudo-random number in the first stage. Nevertheless, the participant who reveals last can also cause the protocol to fail. Since each algorithm that generates the final random number (e.g. XOR) is deterministic and open, the latest participant can calculate the final answer by

inspection of the other nonces (Without letting them know the answer). The last delegate can reveal if the answer is in favor of him and might not reveal if it is not. In this way the others can never guess the final random number and the protocol will not finish successfully. Since the other participants may not know his favor and the adversary can collude with some delegates, It is hard for others to decide what to do due to this action.

If at least one delegate does not reveal, an early idea is that the other participants compensate his action by replacing his nonce with an extra random number (e.g. blockchain [13]). However, this choice is equal to select a random number from another publicly verifiable source without taking advantages of the proposed protocol. Thus, it will have all of the shortcomings discussed before; besides, the adversary can try his luck one more time to get his desirable random value. The main problem with this protocol is that the last delegate has a complete knowledge of all other nonces of participants. we tackle this problem by designing a new scheme based on threshold cryptosystem.

III. Preliminaries

In this section, we briefly review the two protocols that are closely related to our proposed protocol: Secret Sharing, and Threshold Cryptosystems.

A. Secret Sharing

Secret sharing was introduced by Shamir and Blakely. The basis of the idea was built up on Lagrange interpolation polynomials and vector spaces [13]. This algorithm is used for sharing a secret parameter between groups of individuals. The secret only can be reconstructed when the shares are combined together. Decryption the secret shared value needs the cooperation of participants and individual shares are not useful alone. Due to the importance of publicly verifiable secret sharing with sufficient information rate, these schemes have made a lot of progress. In our scheme, to ensure the protocol against the malicious behavior of participants, it is necessary for the secret value to be accessible to a subset of participants that has more members than a specific value k. As a result, a threshold scheme is an appropriate option and will be discussed in the next section.

B. Threshold Cryptosystems

The suggested scheme in [14] encrypts messages using a public key which the corresponding private key is a shared value among the participants. A (k,n) threshold cryptosystem attempts to let participants access the secret shared value with the condition that at least k of n participants are cooperating for decryption. The goal is to divide secret S into n shares $S_0, S_1, \ldots, S_{n-1}$ such that knowledge of k or more shares makes S easy to compute otherwise its computation will be infeasible. The Proposed scheme eliminates the need for trusted parties to select and distribute secret keys. Finding the secret key is dependent on the cooperation among participants. In the case of more than $k-1$ cheaters, the security of the algorithm is compromised. Thus, there exists an assumption that the majority of participants are honest.

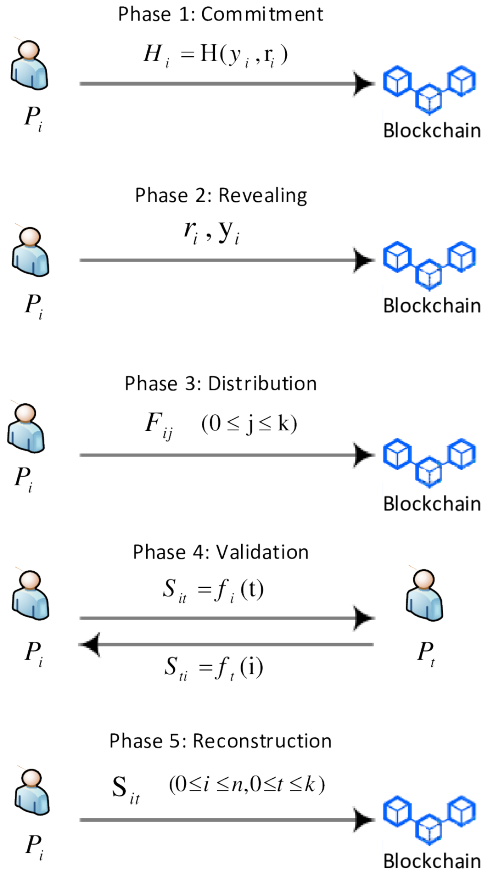

Fig. 1. Schematic View of Our Proposed Scheme

IV. The Proposed Scheme

Here, we discuss the details of our proposed scheme by describing each phase of the protocol and what participants do in each step. Prior to the explanation of the proposed protocol, we emphasize that the proposed scheme is heavily inspired by the threshold cryptosystem introduced by [14]. Similar schemes could be probably derived from other threshold cryptosystems that are not reliant on any trusted third party. Figure 1 schematically describes the overall transactions done by each party.

As it has already been mentioned that the blockchain-based public randomness source with semi-trusted delegates proposed in [10] does not provide fairness in the sense that the last participant –or delegate– who should reveal her share has the advantage of knowing the output of the randomness source before revealing her share. Thus, she can attack the protocol by not revealing hers in case the result is not in her favor. This would significantly bias the probability distribution in her favor or could result in a form of denial of service attack.

In our proposed scheme we provide means of hiding the randomness output from any subset of delegates that are not abiding by the rules of the protocol. Therefore, due to the fact that the last participant to reveal her share is unaware of the output; she would not have an incentive in not revealing her share, and even if she does not reveal her own share, other could easily ignore her role in randomness generation without causing a significant bias in the distribution of random output.

On the other hand, the proposed scheme maintains the public verifiability that is required for such a randomness source. In other words, any entity that has not an active role in the randomness generation could verify that the protocol is done correctly. And, if that there is no group of corrupted delegates with less than k members the output is provably random unless there is a participant who can solve the discrete logarithm problem. Additionally, if there exist such a group of corrupted delegates forcing a significant bias on the output random value has a considerable financial cost which has been discussed in previous literature [12].

Our protocol has five phases, in all of them, the participants –or delegates– record some information into a public blockchain which can be accessed by any user who is willing to verify the correctness of the protocol. The recording to the blockchain acts as a form of broadcasting that is not time-bound. On the other hand, the same blockchain or any other public blockchain which has a significant randomness in its blocks and is considerably robust against 51% attacks could be employed by the system for generating the random value R which will be added to the randomness generated by the delegates. The same blockchain can be used for the acquiring additional randomness that would be added to the randomness generated by the delegates.

Note that a new phase should not be started unless all the participants are finished with the previous one. The following elaborates each phase of the protocol. In what follows, n is the total number of delegates and k is the number of delegates required for reconstructing the random value x. Also, g is an element of a cyclic multiplicative group G of order q in which discrete logarithm is assumed to be a hard problem.

a) Phase1 (Commitment): Each delegate P_i for $(i = 1, \ldots, n)$ chooses a random value $x_i \in Z_q$ then computes $y_i = g^{x_i}$. Assume that H is a commitment function –a secure cryptographic hash function can be an example–, the delegate P_i generates another random value r_i with at least $log_2(||Z_q||)$ random bits and calculates $H_i = H(y_i, r_i)$ then signs and records it into the blockchain.

b) Phase2 (Revealing): When all delegates have recorded their commitments (this could be understood by monitoring the blockchain state), all of them reveal their shares by submitting transactions containing y_i and r_i values to the blockchain. Now, any party can calculate $y = \prod_{i=1}^{n} y_i$. The value y that equals to g^x is publicly known but no single party can calculate x unless she is capable of calculating discrete logarithm in G.

At this point of protocol, the participants should extract some randomness R from some blocks of the blockchain. The block number of those blocks should have already been determined before they have been generated. This, in fact, stops any attacker who does not control at least k members of delegates from manipulating the block randomness to produce a random value desirable for her.

c) Phase3 (Distribution): In this phase, the delegates use some form of verifiable secret sharing to distribute the random value x such that any subset of delegates with k members can reconstruct the value. Each delegate P_i chooses a random polynomial function $f_i(z)$ in which all coefficients are members of Z_q and the degree of the polynomial is less than k. Note that the value of $f_{i,0}$ should be equal to x_i.

$$f_i(z) = f_{i,0} + f_{i,1} * z + f_{i,2} * z^2 + \cdots + f_{i,k-1} * z^{k-1} \quad (1)$$

Then each delegate P_i records the values of $F_{i,j} = g^{f_{i,j}}$ for $j = 0, \ldots, k-1$ to the blockchain. It should be noted that the value of $F_{i,0}$ is the same as already recorded y_i. After that, P_i records the encrypted values of $s_{i,t} = f_i(t)$ for $t = 1, \ldots, n$ with the public key of the other delegate P_t. This encrypted value is signed by the private key of P_i.

d) Phase 4 (Validation): Each delegate P_t decrypts and verifies the value of $s_{i,t}$ recorded by the other delegate P_i for $i = 1, \ldots, n$ by checking the following equation:

$$g^{s_{i,t}} = \prod_{j=0}^{k-1} F_{i,j}^{t^j} \quad (2)$$

If the equation (2) does not hold for the information received from any delegate P_i the delegate P_t should publish the unencrypted value of $s_{i,t}$ to the blockchain. This will show that the user P_i is cheating. Since the encrypted value of $s_{i,t}$ is already on the blockchain it is verifiable for the public that the value could be encrypted by the public key of P_t to what is already submitted by P_i.

In the case that equation (2) is valid for all $s_{i,t} (i = 1, \ldots, n)$ user P_t records a signed $y = g^x$ value to the blockchain demonstrating the shares received by other delegates is validated by him.

e) Phase 5 (Reconstruction): When all delegates have signed h, any coalition of k delegates should be able to reconstruct the value of x. To this end, all delegates submit their decrypted $s_{i,t}$ values to the blockchain although the first k ones are enough. By solving the system of linear equations $s_{i,t} = f_{i,0} + f_{i,1} * t + \cdots + f_{i,k-1} * t^{k-1}$ for $t = 1, \ldots, n$ submitted by P_i from any subset of delegates with k members, any party can calculate the values of x_i and correspondingly $x = \sum_{i=1}^{n} x_i$ can be calculated. The theorem 3.1 in [14] proves the system of equations can be solved by any such subset of members.

After recovering the value x, it should be fed into a randomness extraction function concatenated by the value R which has been derived from the blockchain's randomness in Phase 2 of the scheme. In (3) the Ext function is a random extractor function, an example of such a function is a secure cryptographic hash algorithm.

TABLE I
ESTIMATED SPACE REQUIRED FOR EACH PHASE OF PROTOCOL

	Estimated Size	Est. Gas Consumption (Kgas)
Phase 1	$2na$	5120
Phase 2	$3na$	7680
Phase 3	$(2n(n-1)+2nk)a$	209920
Phase 5	$2na$	5120
Phase 6	$2nka$	51200
Total	$(2n^2+4nk+5n)a$	279040

$$r = Ext(x||R) \tag{3}$$

The value r is the final public random value produced by the scheme. It should be mentioned that the value x is publicly random (consistent, autonomous, and unpredictable) if there is no colliding group of delegates with more than k members while R is publicly random if there is no entity in the blockchain with at least more than 51% hashpower in the blockchain. As a result, the concatenation of values x and R in the input of Ext function makes a successful attack against the security of the system much harder since manipulation of one part needs huge amount of computaion power and another part requires corruption of more at least k semi-trusted delegates.

A. Security Properties

The security of the proposed scheme is based on the security of the Threshold Cryptosystem introduced in [14] and analysis provided by [12] and [4]. As it is the mentioned in [14] the underlying Threshold Cryptosystem is secure only if $2k-1 < n$ the same holds true for the proposed scheme.

B. Memory Cost on Blockchain

In the proposed scheme, most of the cost is associated with writing into a public blockchain for which almost all public blockchains require a fee. This means that the size of the information that should be recorded on the blockchain is probably the most important concern. In the following, we provide an estimation of the total size of the information submitted into blockchain separated by phases. Table I contains estimated amounts of data required to be recorded on the blockchain in each phase. The security parameter is denoted by a and it is assumed that group G is on Elliptic Curves, thus, the size of the values should be of order $O(\sqrt{(}a))$. In order to give a tangible example, the cost is estimated for executing protocol on Ethereum network. Gas determines transaction fees in Ethereum network. The calculated results are based on the assumption of $n = 32$, $k = 10$ and 80 kilo gas for storing 64 byte data on the Ethereum blockchain.

V. CONCLUSION

As mentioned previously, Bitcoin and other cryptocurrencies based on blockchain introduced a new approach to make a publicly verifiable randomness source. The necessity of access to a verifiable randomness source for cryptography and other applications led us to propose a scheme which employs threshold secret sharing and utilizes blockchain to meet the security requirements. Our proposed protocol has five phases for storing information on blockchain to ensure public verifiability which are the fundamental features of publicly verifiable randomness sources.

REFERENCES

[1] S. GOLDWASSER, S. KLEIN, E. MOSSEL, and O. TAMUZ, "Publicly verifiable randomness," 2018.

[2] M. J. Fischer, M. Iorga, and R. Peralta, "A public randomness service," in *Security and Cryptography (SECRYPT), 2011 Proceedings of the International Conference on*, pp. 434–438, IEEE, 2011.

[3] J. Clark and U. Hengartner, "On the use of financial data as a random beacon.," *EVT/WOTE*, vol. 89, 2010.

[4] I. Bentov, A. Gabizon, and D. Zuckerman, "Bitcoin beacon," *arXiv preprint arXiv:1605.04559*, 2016.

[5] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 281–310, Springer, 2015.

[6] M. Andrychowicz and S. Dziembowski, "Distributed cryptography based on the proofs of work.," *IACR Cryptology ePrint Archive*, vol. 2014, p. 796, 2014.

[7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[8] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Security and Privacy (SP), 2014 IEEE Symposium on*, pp. 459–474, IEEE, 2014.

[9] N. Van Saberhagen, "Cryptonote v 2. 0," 2013.

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

[11] N. community, "Nxt whitepaper," 2014.

[12] C. Pierrot and B. Wesolowski, "Malleability of the blockchains entropy," *Cryptography and Communications*, vol. 10, no. 1, pp. 211–233, 2018.

[13] Q. Peng and Y. Tian, "Publicly verifiable secret sharing scheme and its application with almost optimal information rate," *Security and Communication Networks*, vol. 9, no. 18, pp. 6227–6238, 2016.

[14] T. P. Pedersen, "A threshold cryptosystem without a trusted party," in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 522–526, Springer, 1991.