Random Re-Ordering of the Parties in the Consensus Protocol

Andrey Sobol[1], Volodymyr G. Skobelev[5], Julian Konchunas[1], Viktor Radchenko[4], Sabina Sachtachtinskagia[1,3], Oleksandr Letychevskyi[5], Volodymyr Peschanenko[6], and Maxim Orlovsky[1,2]

[1] Pandora Core AG
[2] BICA Labs
[3] AUEB
[4] Garuda AI
[5] Glushkov Institute of Cybernetics of NAS of Ukraine
[6] Kherson State University

Abstract. Generation of a publicly verifiable bias-resistant distributed randomness is one of the actual problems in blockchain and its various applications. The complexity of this problem increases significantly for consensus algorithm operating on a decentralized network topology on the assumption that there are neither a trusted third party nor a trusted dealer. Such situation is caused by the fact that the logical structure of algorithms intended to solve the subtasks typical for this problem becomes much more complicated. Besides, there arise some subtasks caused by the complete distribution of the analyzed blockchain network. One of such nontrivial subtasks is the implementation of random re-ordering of the parties, based on generated randomness. This random reordering defines the roles of the parties in the next epoch, and is intended to support equal access of the parties to the functioning of the blockchain network. We present a simplified version of the generation of a publicly verifiable reliable distributed randomness for the consensus protocol operating on a decentralized network topology on the assumption that there are neither a trusted third party nor a trusted dealer. On this base we solve the problem of the random re-ordering for parties which will participate in the implementation of the next epoch.

Keywords: Distributed randomness · Public verifiability · Random reordering · Consensus protocols

1 Introduction

A completely distributed blockchain operating on a decentralized network topology on the assumption that there are no trusted third parties or dealers, can be considered as the backbone of the emerging open-access distributed Virtual Machines [1] for decentralized, token-driven resource management.

Thus, the functioning of such blockchain networks significantly relies on the performance of the used consensus mechanisms. It is worth to point out that

many of these mechanisms can be considered in terms of a uniform framework based on Zero-Knowledge (ZK) Proof systems [2]. Some survey of consensus mechanisms in blockchain networks has been presented in [3].

It is generally accepted that public verification means that any party that does not necessarily participate in the randomness generation can audit the protocol execution a posteriori with the aim to attest that the randomness source is reliable and unbiased.

The concept of a public randomness beacon that relies on a trusted third party has been proposed in [4]. The necessity to use public randomness sources effectively has increased sharply for blockchain networks [5].

An approach for generation of a distributed randomness beacon that guarantees output delivery and uniformly distributed randomness for the parties that use it, as long as a majority of them are honest, has been proposed in [6], on the assumption that a dealer participates in the proposed protocols.

An important feature of this approach is that any party that does not necessarily participate in the randomness generation can audit the protocol execution a posteriori with the aim to be convinced that the randomness source is reliable and unbiased. However the requirement of the presence of the dealer does not allow to use these constructions directly for a completely distributed network on which there is neither a trusted third party nor a dealer.

Our aim is to generate some uniformly distributed randomness for completely distributed blockchain operating on a decentralized network topology, in the assumption that there are neither the trusted third party nor the dealer, and to apply this distributed randomness for the random re-ordering of parties for the implementation of the next epoch.

1.1 Related Works

The basic scheme for secret sharing has been proposed in [7], and guaranties the correct output only in the case when all parties are honest. Due to constructions considered in [8], it has been established in [9] that there exists some poly-time threshold verifiable secret sharing (VSS) protocol, on the assumptions that the majority of the parties are honest and that some broadcast channel is available.

In [10] an approach to construct publicly verifiable secret sharing (PVSS) protocols has been proposed. This protocol gives the ability to the parties to verify their own shares, but also anybody can verify that the parties has received correct shares.

The model of non-interactive PVSS has been proposed in [11]. Some other PVSS schemes have been presented in [12–14]. Unfortunately, PVSS schemes, presented in [10–14], lead to high computational cost. Critical survey of these and some others PVSS has been presented in [15]. Lowering of the computational cost has been one of the main aims in protocols presented in [6].

It is well known that the randomness can be manipulated by the parties of the blockchain. To prevent these manipulations some delay functions [5, 16, 17] can be used, so that when any malicious party computes the random output, it

is too late to manipulate it. Thus, the delay function gives the chance to verify that the randomness has not been manipulated.

Hash-based signature schemes that are most often used in PVSS, are RSA [18] and ECDSA [19]. Security of these schemes are based on algebraic assumptions, i.e. security of RSA relies on the difficulties of solving the factorizing large numbers problem, while security of ECDSA relies on the difficulties of solving the discrete logarithm problem.

It is worth to note that if any of these assumptions is violated, for example, due to the development of a quantum computer, then the corresponding signature scheme is damaged forever.

The Merkle Signature Scheme [20] depends only on a secure hash function and a secure one-time signature. Some variants of this scheme has been developed: an improved Merkle signature scheme (CMSS) [21] in which two authentication trees are used, GMSS [22] which uses a scheduling strategy to precompute upcoming signatures, XMSS [23] in which a hash tree is used to reduce the authenticity of many pseudo-randomly generated one-time signature keys to one public XMSS key, XMSS-MT [24] which is a multi Tree XMSS intended to provide a large number of signatures, and SPHINCS [25] which is some many-time signature scheme that uses a hyper-tree, i.e. a tree of trees. Software implementations for some of above listed variants of Merkle Signature Scheme have been analyzed in [26].

To provide equal opportunities for an involvement of parties in a completely distributed blockchain operating on a decentralized network topology, unbiased random generation of the re-ordering for the parties can be used.

The basic algorithm, called the Fisher—Yates shuffle [27], has been presented in [28, 29], and is as follows (A is the given array with N elements, such that $A[i] = i$ for all $i = 1, \ldots, N$).

RandomPerm(A, N)
begin
for $i = 1$ to $N - 1$
do
choose the integer j uniformly at random from the set $\{i, ..., N\}$;
swap $A[i]$ and $A[j]$;
end do
end

This algorithm guarantee that the probability that $A[i] = i$ equals to N^{-1} for any $i \in \{i, ..., N\}$. Moreover, the expected number of fixed points in a random permutation equals to 1, i.e. it is independent of the integer N.

It is evident that the subtle aspect for implementation of this algorithm consists of how to choose uniformly and randomly an element of the given set. Surveys of methods proposed for generation of permutations by computer have been presented in [30, 31].

1.2 Our contribution

To achieve the scalability for implementation of completely distributed blockchain operating on a decentralized network topology on the assumption that there are neither a trusted third party nor a dealer, the set of parties $\mathcal{P} = \{P_1, \ldots, P_N\}$ that take the part in the implementation of the current epoch are partitioned into 3 groups $\mathcal{P}_j = \{P_1^{(j)}, \ldots, P_{n_j}^{(j)}\}$ $(j = 1, 2, 3)$, where $\mathcal{P}_1$ is the set of ZK validators, and the subset $\mathcal{P}_2$ is the set of Random Part (RP) validators. The subset $\mathcal{P}_3$ consists of overwhelming number of parties, but these parties only take part in the commit-delay-reveal scheme pointed in the next Section.

It is assumed that $|\mathcal{P}_2| \ll |\mathcal{P} \setminus \mathcal{P}_2|$ (for example, it is enough to consider that $|\mathcal{P}_2| \leq 0.05|\mathcal{P} \setminus \mathcal{P}_2|$). The necessity of this inequality is caused by the following circumstances.

To achieve the reliability and scalability for the re-ordering of parties at regular and predictable intervals in the presence of adversarial behavior, and without any trusted dealer for the initial setup, the local sources of randomness can be used in the following way.

The sufficiently small group of parties $\mathcal{P}_2$ independently generate their randomness on the base of some threshold random scheme which we describe below. The other parties generate their randomness with using the following commit-delay-reveal scheme:

Step 1. Each party from the set $\mathcal{P}\setminus\mathcal{P}_2$ generates 32 bit random data and publish hash(data).

Step 2. Each party from the set $\mathcal{P}\setminus\mathcal{P}_2$ is forced to wait for the prescribed period of time.

Step 3. Each party from the set $\mathcal{P}\setminus\mathcal{P}_2$ provides its data.

Such approach gives the chance to implement the interactions in the commitment scheme as follows: during the commit phase the values of randomness are chosen and specified, while during reveal phase which starts with some admissible delay these values are revealed and checked.

Proposed threshold random scheme consists of the following three phases.

In the first phase, called the Public Key Phase, each validator from the ZK validators subset $\mathcal{P}_1$ provides its public key (PubKey), epoch hash (EH) and the signed hash H(PubKey||EH). In the role of the hash function H can be used the Cryptographic hash function SHA256, or any other Cryptographic hash function, similar to SHA256.

In the second phase, called the Threshold Random Encrypted Part Phase each validator from the RP validators subset $\mathcal{P}_2$ generates some random string, which is 32 byte data (thus, at our assumptions $|\mathcal{P}_2| \ll N \ll 2^{256}$, where N is the number of parties that take the part in the implementation of the current epoch), split this random string in accordance to Shamir's secret scheme, and encrypts each secret by the PubKey, chosen by him from Public Key Phase.

When this process is completed, each validator from the subset $\mathcal{P}_2$ provides its list of encrypted secrets, epoch hash, and the signed hash H(the root of merkle tree||EH).

In the third phase, called Private Key Publishing Phase, each validator from the subset of the ZK validators $\mathcal{P}_1$ provides its private key (PrKey). Each party from the subset $\mathcal{P} \setminus \mathcal{P}_2$ reveals its random, and each party from the subset of the RP validators $\mathcal{P}_2$ reveals its random using corresponding PrKey.

When this process is completed, each party which will participate in the implementation of the next epoch computes the random re-ordering of parties for the next epoch.

It is evident that the above described round is intended for achievement of the following purposes:

1. We should know the result of the random, i.e. that parties from the set $\mathcal{P}_1$ have provided not less then $50\% + 1$ of all private keys, and that parties from the set $\mathcal{P}_2$ have presented their ciphered data.

2. If the set $\mathcal{P}_1$ consists of less then 50% of corrupted parties, and the set $\mathcal{P}_2$ consists of less then 100% of corrupted parties, then corrupted parties could not prevent to receive random, or to learn it in advance.

It should be noted that every time we construct new partition of the set of parties. For correctness of the proposed protocol it is necessary that the majority of parties in the set $\mathcal{P}_1$ is honest, and at least one of the parties in the set $\mathcal{P}_2$ is also honest. Because we constantly mix validators, these assumptions are quite realistic. It should be noted, however, that the probability that there is an honest majority in each round depends on the number of honest parties in $\mathcal{P}$, as well as on the shares $\frac{|\mathcal{P}_1|}{|\mathcal{P}|}$ and $\frac{|\mathcal{P}_2|}{|\mathcal{P}|}$.

2 The random re-ordering of the parties

When the commit-delay-reveal scheme is completed, the set of the parties which will participate in the implementation of the next epoch can be formed. This set can be considered as the array consisting of the same ordering of all parties that haven't been disqualified during the current epoch, and perhaps some new parties are added to its tail.

For simplicity we denote this array $\mathcal{P} = \langle P_1, \ldots, P_N \rangle$, and, also, we denote $\mathcal{R} = \langle r_1, \ldots, r_N \rangle$ the array of 32 byte random data that have been produced by the parties from the array $\mathcal{P}$ in the current epoch. Assume that for each party P_j that has been unsuccessful in the commit-delay-reveal scheme, as well as for each new party P_j, its 32 byte random data r_j is the zero sequence.

To implement the random re-ordering of the elements of the array $\mathcal{P}$ we have used the following refinement of the Fisher—Yates shuffle scheme [27, 28, 29], based on the use of some random positive integer M ($M \gg N$), generated on the base of the array $\mathcal{R}$.

RanReOrd($\mathcal{P}$, N, M)
  begin
    for $i = 1$ to $N - 1$
      do
        $j := (M - i + 1)(\mathrm{mod}(N - i + 1)) + i$;

```
        swap P[i] and P[j];
      end do
    end
```

It is reasonable to make the following remark concerning the proposed above algorithm *RanReOrd*($\mathcal{P}$, N, M).

Since M ($M \gg N$) is some random positive integer, then for any fixed integer $i = 1, \ldots, N-1$ the integer

$$j = (M - i + 1)(\mathrm{mod}(N - i + 1)) + i$$

is some random positive integer, such that $i \leq j \leq N$. This factor implies that

$$j = (M - i + 1)(\mathrm{mod}(N - i + 1)) + i \quad (i = 1, \ldots, N)$$

is some sequence of random integers. For any fixed integer $i = 1, \ldots, N-1$ the elements $\mathcal{P}[i]$ and $\mathcal{P}[j]$ ($i \leq j \leq N$) are swapped. Therefore at the completion of the algorithm *RanReOrd*($\mathcal{P}$, N, M) the elements of the array $\mathcal{P}$ are reordered in a random way.

The following two approaches intended to generate some random positive integer M ($M \gg N$) on the base of the list $\mathcal{R}$ has been checked.

The first approach is based on the computing of the binary string

$$r = \bigoplus_{i=1}^{N} r_i, \tag{1}$$

where $\bigoplus$ is bit-wise XOR operation. Afterwards, the random positive integer M can be defined as the result of the transformation of the binary string r into the corresponding positive integer.

The justification that M is some random integer follows from the fact that there is the honest majority among the parties that participates in the implementation of the current epoch.

The advantage of this approach consists in the fast computing of the random positive integer M.

From our point of view, at least, the following two shortcomings are inherent into this approach.

Firstly, there can be some groups of parties, for each of which the result of the bit-wise XOR operation is the zero sequence, and, thus, these groups of parties, as a matter of fact, are eliminated from the formation of the re-ordering of parties for the next epoch.

Secondly, the corrupted parties, using these or the others unforeseen shortcomings of the delay function, can try to influence on the computation of the binary string r (see [32], for example), and, thus, on the computation of the integer M.

The second approach is based on the idea to use some well known sufficiently easily computable function $f(x_1, \ldots, x_N)$ of non-negative discrete independent random variables x_i ($i = 1, \ldots, N$) with the known distribution laws.

In this case, each random binary string r_i $(i = 1, \ldots, N)$ can be transformed independently into some random non-negative integer m_i, and then we can set

$$M = \lceil f(m_1, \ldots, m_N) \rceil. \tag{2}$$

Proceeding from the probabilistic reasons, the most expedient choice is the function

$$f(x_1, \ldots, x_N) = N^{-1} \sum_{i=1}^{N} x_i, \tag{3}$$

where x_i $(i = 1, \ldots, N)$ are non-negative discrete independent random variables with the known distributions. Due to this function, the integer M can be computed as follows.

Each binary string r_i $(i = 1, \ldots, N)$ can be transformed into the corresponding non-negative integer m_i. Thus, formulae (2) and (3) imply that

$$M = \lceil N^{-1} \sum_{i=1}^{N} m_i \rceil. \tag{4}$$

Since r_i $(i = 1, \ldots, N)$ are random values that are uniformly chosen from the set $\{0, 1, 2^{256} - 1\}$, we deal with the function (3) under the supposition that each x_i $(i = 1, \ldots, N)$ is the uniformly distributed random variable on the consecutive integers $0, 1, 2^{256} - 1$.

Thus, for each random variable x_i $(i = 1, \ldots, N)$ the mean equals to

$$\mathbf{E}(x_i) = 0.5(2^{256} - 1), \tag{5}$$

and the variance equals to

$$\mathbf{Var}(x_i) = \frac{2^{512} - 1}{12}. \tag{6}$$

Formulae (5) and (6), taking into account the properties of the mean and the variance imply that for the random variable

$$X = N^{-1} \sum_{i=1}^{N} x_i \tag{7}$$

we get

$$\mathbf{E}(X) = \mathbf{E}\left(N^{-1} \sum_{i=1}^{N} x_i\right) = N^{-1} \sum_{i=1}^{N} \mathbf{E}(x_i) = N^{-1} \sum_{i=1}^{N} 0.5(2^{256} - 1) =$$

$$= 0.5(2^{256} - 1) N^{-1} \sum_{i=1}^{N} 1 = 0.5(2^{256} - 1) N^{-1} N = 0.5(2^{256} - 1), \tag{8}$$

$$\mathbf{Var}(X) = \mathbf{Var}\left(N^{-1} \sum_{i=1}^{N} x_i\right) = N^{-2} \sum_{i=1}^{N} \mathbf{Var}(x_i) = N^{-2} \sum_{i=1}^{N} \frac{2^{512} - 1}{12} =$$

$$= N^{-2} \cdot \frac{2^{512}-1}{12} \cdot \sum_{i=1}^{N} 1 = N^{-2} \cdot \frac{2^{512}-1}{12} \cdot N = \frac{2^{512}-1}{12N}. \quad (9)$$

Formula (9), in its turn, implies that for the standard deviation of the random variable $X = N^{-1} \sum\limits_{i=1}^{N} x_i$ the following formula is true

$$\sigma_X = \sqrt{\frac{2^{512}-1}{12N}}. \quad (10)$$

It should be noted that formulae (8)-(10) represent probability-theoretic characteristics of the random variable X, defined by formula (7). Unfortunately, no probability-theoretic characteristics of the random variable constructed according to the formula (1) are known to us.

3 Conclusion

In the given paper we have proposed some approaches for the solution of the problem of random re-ordering for parties which will participate in the implementation of the next epoch in the completely distributed blockchain operating on a decentralized network topology on the assumption that there are neither a trusted third party nor a dealer. The results of experiments has shown that time needed for computing the re-ordering is acceptable for both proposed approaches.

Comparative analysis of efficiency for different well known sufficiently easily computable functions $f(x_1, \ldots, x_N)$ of non-negative discrete integer-valued independent random variables x_i $(i = 1, \ldots, N)$ form some trend for future research.

Another trend for future research consists of comparable analysis for characteristics of these re-orderings for parties to resist to these or other actions of the corrupted parties. For the solution of this problem it is supposed to use the System of insertion modeling and symbolic verification of large systems [33].

Acknowledgements. We thank an anonymous referee for carefully reading a preliminary version of this paper, for pointing out some slips made in it, and for posing questions that have led to improvements of the presentation.

References

1. Kosba A., Miller A., Shi E., Wen Z., and Papamanthou C.: Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, pp. 839–858 (2016). https://doi.org/10.1109/SP.2016.55
2. Wu H., and Wang F.: A Survey of noninteractive zero knowledge proof system and its applications. The Scientific World Journal, vol. 2014, Article ID 560484, 7 pages (2014) https://doi.org/10.1155/2014/560484

3. Wang W., Hoang D.T., Xiong Z., Niyato D., Wang P., Hu P., and Wen Y.: A Survey on consensus mechanisms and mining management in blockchain networks. https://arxiv.org/pdf/1805.02707.pdf
4. Rabin M.O.: Transaction protection by beacons. J. Comput. Syst. Sci., **2**(27), pp. 256–267 (1983)
5. Bonneau J., Clark J., Goldfeder S.: On bitcoin as a public randomness source. Cryptology ePrint Archive, Report 2015/1015 (2015), http://eprint.iacr.orrg/2015/1015
6. Cascudo I., David B.: SCRAPE: Scalable randomness attested by public entities. In: Gollmann D., Miyaji A., and Kikuchi H. (eds.) ACNS 2017, LNCS, vol. 10355, pp. 537-556. Springer, Heidelberg (2017). https://doi.org/10.1007/978 − 3 − 319 − 61204 − 1_27
7. Shamir A.: How to share a secret. Communications of the ACM, **22**(11), pp. 612–613 (1979)
8. Chor B., Goldwasser S., Micali S.: Verifiable secret sharing and achieving simultaneti in the presence of faults (extended abstract). In: Proc. 26^{th} IEEE Symposium on Foundations of Computer Science (FOCS), pp. 383–395. IEEE Computer Society Press (1985)
9. Rabin T., Ben-Or M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In 21^{st} ACM STOC, pp. 3-85. ACM Press (1989)
10. Stadler M.: Publicly verifiable secret sharing. In: Maurer, U. M. (eds.) EUROCRYPT'96, LNCS, vol. 1070, pp. 190–199. Springer, Heidelberg (1996). https://doi.org/10.1007/3 − 540 − 68339 − 9_17
11. Schoenmakers B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Wiener M. (eds) CRYPTO' 99, LNCS, vol. 1666, pp. 148–164. Springer, Berlin, Heidelberg (1999). https://doi.org/10.1007/3 − 540 − 48405 − 1_10
12. Boudot F., Traoré J.: Efficient publlicly verifiable secret sharing schemes with fast or delayed recovery. In: Varadharajan V., and Mi Y. (eds) ICICS 99, LNCS, vol. 1726, pp. 87–102. Springer, Heidelberg (1999). https://doi.org/10.1007/978 − 3 − 540 − 47942 − 0_8
13. Heidarvand S., Voillar J.L.: Public verifiability from parings in secret sharing schemes. In: Avanzi R.M., Keliher L., and Sica F. (eds) SAC 2008, LNCS, vol. 5381, pp. 294–308, Springer, Heidelberg (2009). https://doi.org/10.1007/978 − 3 − 642 − 04159 − 4_19
14. Jhanvar M.P.: A practical (non-interactive) publicly verifiable secret sharing scheme. In: Bao F., and Weng J. (eds) ISPEC 2011, LNCS, vol. 6672, pp. 273–287, Springer, Berlin, Heidelberg (2011). https://doi.org/ 10.1007/978 − 3 − 642 − 21031 − 0_21
15. Peng K.: Critical survey of existing publicly verifiable secret sharing schemes. IET Information Security **6**(4), pp. 249–257 (2012) https://doi.org/10.1049/iet-ifs.2011.0201
16. Lenstra A.K., and Wesolowski B.: A random zoo: sloth, unicorn, and trx. IACR Cryptology ePrint Archive, Report 2015/366 (2015) https://eprint.iacr.org/2015/366
17. Bünz B., Goldfeder S., and Bonneau J.: Proofs-of-delay and randomness beacons in Ethereum (2017). http://www.jbonneau.com/doc/BGB17-IEEESB-proof_of_delay_ethereum.pdf
18. Rivest R.L., Shamir A., and Adleman L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, **21**(2), pp. 120-–126, (1978)

19. Johnson D., Menezes A., and Vanstone S.: The elliptic curve digital signature algorithm (ecdsa). International Journal of Information Security, **1**(1), pp. 36-–63 (2001)
20. Merkle R. C.: A certified digital signature. In: Brassard G. (eds) Advances in Cryptology — CRYPTO' 89, LNCS, vol. 435, pp. 218–238, Springer, New York, NY (1989). https://doi.org/10.1007/0 − 387 − 34805 − 0_21
21. Buchmann J., Garcıa L.C.C., Dahmen E., Doring M., and Klintsevich E.: CMSS – an improved merkle signature scheme. In: Rana Barua R., and Lange T. (eds) INDOCRYPT 2006, LNCS, vol. 4329, pp. 349–363, Springer-Verlag, Berlin, Heidelberg (2006). https://doi.org/10.1007/11941378_25
22. Buchmann J., Dahmen E, Klintsevich E., Okeya K., and Vuillaume C.: Merkle signatures with virtually unlimited signature capacity. In: Katz J., and Yung M. (eds) ACNS 2007, LNCS, vol. 4521, pp. 31—45, Springer-Verlag, Berlin, Heidelberg (2007). https://doi.org//10.1007/978 − 3 − 540 − 72738 − 5_3
23. Buchmann J., Dahmen E, and Hulsing A.: XMSS - a practical forward secure signature scheme based on minimal security assumptions. In: Yang B.Y. (eds) PQCrypto 2011, LNCS, vol. 7071, pp. 117–129, Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978 − 3 − 642 − 25405 − 5_8
24. Hulsing A., Rausch L., and Buchmann J.: Optimal parameters for XMSS-MT. In: Cuzzocrea A., Kittl C., Simos D.E., Weippl E., Xu L. (eds) CD-ARES 2013, LNCS, vol. 8128, pp. 194–208, Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978 − 3 − 642 − 40588 − 4_14
25. Bernstein D.J., Hopwood D.,Hulsing A., Lange T., Niederhagen R., Papachristodoulou L., Schneider M., Schwabe P., and WilcoxO'Hearn Z. SPHINCS: practical stateless hash-based signatures. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, LNCS, vol. 9056, pp. 368—397. Springer, Heidelberg (2015). https://doi.org/10.1007/978 − 3 − 662 − 46800 − 5_15
26. de Oliveira A.K., L*ó*pez J., and Cabral R.: High performance of hash-based signature schemes. International Journal of Advanced Computer Science and Applications, **8**(3), pp. 421–432 (2017)
27. Fisher R. A., and Yates F.: Statistical tables for biological, agricultural and medical research, Oliver & Boyd, London (1938)
28. Durstenfeld R.: Algorithm 235: random permutation. Communications of the ACM, **7**(7), p. 420 (1964) https://doi.org/10.1145/364520.364540
29. Knuth D.E.: The Art of computer programming, third edition, Vol. 2: Seminumerical algorithms, Addison-Wesley, Reading, MA (1997)
30. Sedgewick R.: Permutation Generation Methods. Computing Surveys, **9**(2), pp. 137–164 (1977)
31. Barcher A., Bodini O., Hwang H.K., and Tsai T.H.: Generating random permutations by coin-tossing: classical algorithms, new analysis and modern implementation. ACM Transactions on Algorithms, **13**(2), Article No. 24 (2017) https://doi.org/10.1145/3009909
32. Syta E., Jovanovic P., Kogias E.K., Gailly N., Gasser L., Khoffi I., Fischer M.J., and Ford B.: Scalable bias-resistant distributed randomness. IACR Cryptology ePrint Archive: Report 2016/1067 (2017) https://www.ieee-security.org/TC/SP2017/papers/413.pdf
33. Letichevsky A.A., Letichevskyi O.A., Peschanenko V.S., Weigert T.: Insertion modeling and symbolic verification of large systems. In: SDL 2015: Model-Driven Engineering for Smart Cities, LNCS, vol. 9369, pp. 3-18. Springer-Verlag Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-319-24912-4.1