CONSTRUCTING AND COMPRESSING FRAMES IN BLOCKCHAIN-BASED VERIFIABLE MULTI-PARTY COMPUTATION

Ravi Kiran Raman,[1,3] *Kush R. Varshney,*[1] *Roman Vaculin,*[1]
Nelson Kibichii Bore,[2] *Sekou L. Remy,*[2] *Eleftheria K. Pissadaki,*[1] *and Michael Hind*[1]

[1]IBM Research, Yorktown Heights, NY, USA
[2]IBM Research, Nairobi, Kenya
[3]University of Illinois at Urbana-Champaign, Urbana, IL, USA

ABSTRACT

In previous work, we proposed a scalable multi-party verification scheme for expensive iterative computations on a Blockchain substrate by appropriate storage and endorsement of frames of iterates. In this work, we extend the framework to verify sets of complete computations with different unordered hyperparameters and develop frame ordering and compression algorithms to enable scalability in the system. We illustrate the efficacy of the proposed approach by verifying the OpenMalaria epidemiological simulation.

Index Terms— distributed trust, compression, endorsement

1. INTRODUCTION

Blockchain technologies, although initially developed in the context of cryptocurrencies such as Bitcoin, are now being used in many different contexts including data processing [1] and trusted and secure data sharing in healthcare and genomics [2, 3]. An underappreciated limiting factor to using Blockchain in data-intensive applications is scalability [4]. Whereas “2017 was the year of Blockchain hype,” “2018 is the year of Blockchain scaling” [5, 6, 7].

Machine learning and computational simulation in low-resource environments are emerging applications of Blockchain in which scalability is a very important concern [8]. In our previous work [9], we developed an approach for verifying long-running iterative computations such as training deep neural networks and running epidemiological simulations. The data that must be stored on a Blockchain amounts to iterates of the state of the computation, neural network weights for example. Because of the relative smoothness and eventual convergence of such computations, we can efficiently compress a sequence (or frame) of iterates in the order they are generated and handle many of the scalability issues in that way.

However, a single run of a training job or simulation is not the only computation that must be verified. In this paper, we consider the verification of computations involving multiple runs in applications such as hyperparameter tuning of neural networks and doing ‘what-if analyses’ with various choices of interventions that affect disease spread. In this paper, we propose a system for trusted multi-party collaboration through the sharing of verified models in the case of several complete runs with special attention given to ordering and composing frames to be as efficient as possible in compression (a problem we did not encounter in [9]). We evaluate the system on the Openmalaria framework for evaluating efficacy of malaria control and intervention policies as an example, even though the mechanism can be easily extended to other distributed computational platforms directly as well.

2. SIMULATION AND TRUST MODEL

Consider an enumerative computational experiment, where we evaluate outputs of a black-box function $f : \mathbb{R}^{d_i} \times \mathbb{R}^{d'} \to \mathbb{R}^{d_o}$ over a set of inputs $\left\{X_i \in \mathbb{R}^{d_i} : i \in [n]\right\}$. Let $Y_i = f(X_i, \theta_i)$, for $i \in [n]$, where $Y_i \in \mathbb{R}^{d_o}$ is the output, and $\theta_i \in \mathbb{R}^{d'}$ is an external source of randomness (noise) used in the computation. We adopt a function model similar to [9], and assume $f(\cdot)$ is L-Lipschitz continuous in the input, for all $\theta \in \mathbb{R}^{d'}$ i.e.,

$$\left\|f(X_1, \theta) - f(X_2, \theta)\right\| \leq L \left\|X_1 - X_2\right\|. \tag{1}$$

The input-output pairs, (X_i, Y_i), are referred to as *states*. We aim to ensure the computational validity of the outputs.

Consider a distributed network of agents, where one, referred to as the *client*, computes the outputs for the set of inputs. All agents, called *peers*, have access to $f(\cdot)$ and are informed of the evaluations of the client. Non-overlapping subsets of peers, called *endorsers*, validate individual client computations by (approximate) recomputation. Assume the client reports a state $(\tilde{X}_i, \tilde{Y}_i)$ and let the output recomputed by an endorser j be $\hat{Y}_i^{(j)} = f(\tilde{X}_i, \theta_j)$. The reported state is *valid* if and only if

$$\left\|\tilde{Y}_i - \tfrac{1}{|\mathcal{E}_i|} \textstyle\sum_{j \in \mathcal{E}_i} \hat{Y}_i^{(j)}\right\| \leq \Delta_{\text{val}}, \tag{2}$$

where $\mathcal{E}_i$ is the set of endorsers validating state i, and Δ_{val} is the validation tolerance. Here, we consider Euclidean distance for deviations although this can easily be generalized.

Verification of the experiment concerns ensuring computational integrity through recorded audits of validated states. If the audits record the states $\{\tilde{Z}_i : i \in [n]\}$, then verifying state i, is to guarantee

$$\mathbb{P}\left[\left\|\tilde{Y}_i - \mathbb{E}\left[f(\tilde{X}_i, \theta)\right]\right\| \geq \Delta_{\text{ver}}\right] \leq \rho, \tag{3}$$

for some desired $\rho > 0$, without explicit recomputation. Similar to [9], this is done by recording validated states on a Blockchain and verifying by checking for consistency of the hash chain of the ledger.

3. MULTI-AGENT BLOCKCHAIN FRAMEWORK

We now extend the multi-agent Blockchain framework of [9] and consider a similar functional categorization of agents. Clients run the computations, multiple independent frames of states are validated in parallel by non-overlapping subsets of endorsers, orderers check for consistency in endorsements and append valid frames to the Blockchain ledger as shown in Fig. 1. Note that validation can be massively parallelized over large networks, reducing overheads.

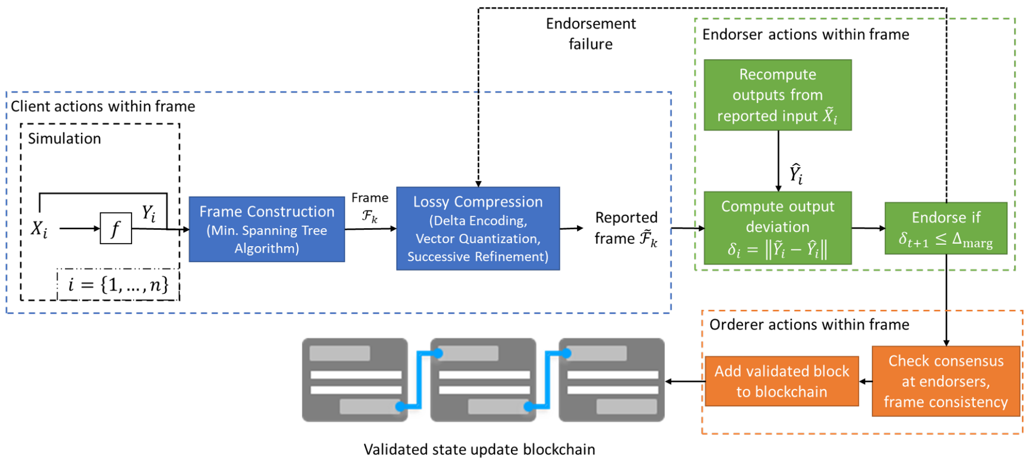
Endorsement failure
Endorser actions within frame
Recompute outputs from reported input $\tilde{X}_i$
$\hat{Y}_i$
Compute output deviation $\delta_i = \|\tilde{Y}_i - \hat{Y}_i\|$
Endorse if $\delta_{t+1} \leq \Delta_{\text{marg}}$
Client actions within frame
Simulation
X_i
f
Y_i
$i = \{1, \dots, n\}$
Frame Construction (Min. Spanning Tree Algorithm)
Frame $\mathcal{F}_k$
Lossy Compression (Delta Encoding, Vector Quantization, Successive Refinement)
Reported frame $\tilde{\mathcal{F}}_k$
Orderer actions within frame
Check consensus at endorsers, frame consistency
Add validated block to blockchain
Validated state update blockchain

Fig. 1. Block diagram of computational trust policy.

3.1. Compression and Validation Schema

The multi-agent Blockchain framework (MBF) [9] constructs frames of states, compresses them using delta encoding and successive refinement lattice vector quantization, reducing the communication and storage overheads and making the system scalable. For iterative computations, the closeness in the Euclidean space of successive iterates can be exploited in easily constructing frames of successive iterates that can be compressed using delta encoding [10] and successive refinement lattice vector quantization [11, 12, 13]. However, in the enumerative computational experiment, such spatial relationships are not available *a priori* and so the construction of frames and their compression is not outrightly evident.

We now formulate a frame construction algorithm that sequentially organizes the states into frames that can be compressed efficiently. From the Lipschitz continuity of $f(\cdot)$, we know that any two inputs X_1, X_2 that are close result in spatially close outputs, Y_1, Y_2. This property is leveraged to construct frames of spatially close states, that can be compressed efficiently using a similar lattice-based compression of state differences.

Each frame comprises an ordering of different states and a compressed representation of the differences according to this order. Consider a set of states $\{Z_1, \dots, Z_n\}$ and let the distance function be $d(\cdot, \cdot)$. The algorithm for frame construction is described in Alg. 1. The pairwise distances between the states are used to construct a forest of trees with the minimum cumulative weight (distances). Here the coding parameter Δ_q represents the maximum magnitude of the state difference incurred in the delta encoding phase, which controls the number of bits allocated per state by the compression scheme. The algorithm uses a modified version of Prim's algorithm for minimum spanning tree construction.

Lemma 1 *The frame construction algorithm Φ minimizes the cumulative weight of the spanning forest for a given Δ_q.*

The result follows directly from the matroidal structure of trees. This also naturally implies the following result.

Corollary 1 *For a given set of states $\{Z_1, \dots, Z_n\}$, minimizes the number of frames, subject to the distance constraint imposed by Δ_q.*

The compression, with approximation error ϵ, is then performed according to Alg. 2. Each tree is compressed into a frame by first choosing a root as checkpoint at random, compressing the checkpoint using the lossy Lempel-Ziv compressor of [9], $\text{ENC}_{LZ}(\cdot)$. The algorithm then scans edges of the tree using depth first search, delta encodes the states along the edges, and compresses the differences using a successive refinement lattice vector quantizer, according to the lattice $\mathcal{L}$, $\text{ENC}_{\mathcal{L}}(\cdot)$. The compressed frames are then composed of the tree structure and the set of compressed differences.

Algorithm 1 Frame construction, $\mathcal{T} = \Phi\left(\{Z_1, \dots, Z_n\}, \Delta_q\right)$

```
$W_{i,j} \leftarrow d(Z_i, Z_j)$, for $i \neq j \in [n]$
Construct weighted complete graph $G = ([n], W)$
Choose $v_1 \in [n]$; $T_1 \leftarrow (\{v_1\}, \emptyset)$, $\mathcal{T} \leftarrow \{T_1\}$, $\mathcal{V} \leftarrow \{v_1\}$
for $r = 2$ to $n$ do
    $d_{\min} \leftarrow \min\{W_{j,k} : j \in \mathcal{V}, k \in [n]\backslash\mathcal{V}\}$
    $(\tilde{u}, \tilde{v}, \tilde{T}) \leftarrow \arg\min\{W_{j,k} : j \in \mathcal{V} \cap T_i, k \in [n]\backslash\mathcal{V}\}$
    if $d_{\min} \leq \Delta_q$ then
        Add vertex $\tilde{v}$ and edge $(\tilde{u}, \tilde{v})$ to tree $\tilde{T}$; $\mathcal{V} \leftarrow \mathcal{V} \cup \{\tilde{v}\}$
    else
        Construct tree $T_{|\mathcal{T}|+1} = (\{\tilde{v}\}, \emptyset)$; $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_{|\mathcal{T}|+1}\}$
    end if
end for
return $\mathcal{T}$
```

In Alg. 2, the lattice for the quantization is chosen according to the dimension of the states. Further information on the input sampling distribution or the joint distribution of the states, if available, can be used to choose optimal lattices. Also, encoding and decoding according to a lattice requires a search for the closest vector on the lattice, which may be implemented using efficient heuristics. We omit the details of the lattice vector coding owing to space constraints and refer the reader to prior work in the area [14, 15].

A direct extension of [9, Thm. 1] indicates that the optimal choice of approximation error for deterministic, L-Lipschitz continuous computations is $\epsilon \leq \frac{\Delta_{val}}{L+1}$. The representation of inter state dependency as trees is optimal as it minimizes the number of bits required to communicate the relationship.

Lemma 2 *If the difference vectors are uniform in $\mathcal{B}(\Delta_q) = \{x : \|x\| \leq \Delta_q\}$, then the maximum expected cost of represent-*

Algorithm 2 Compressor, $\mathcal{F} = \text{ENC}\left(\{Z_1, \ldots, Z_n\}, \mathcal{T}, \epsilon\right)$

```
for all $i \in |\mathcal{T}|$ do
    Choose a root $r \in T_i$, $\tilde{\Delta}_r^{(i)} = \text{ENC}_{LZ}(Z_r, \epsilon)$
    while Depth First Scan($T_i$) do
        Let scanned edge be $(u, v)$ where $u$ is the parent node
        Delta encode along edge, $\Delta Z_v \leftarrow Z_v - Z_u$
        Lattice code difference $\tilde{\Delta}_v^{(i)} = \text{ENC}_{\mathcal{L}}(\Delta Z_v, \epsilon)$
    end while
    Compressed frame, $F_i \leftarrow \left\{T_i, \left\{\tilde{\Delta}_v^{(i)} : v \in T_i\right\}\right\}$
end for
return $\mathcal{F} \leftarrow \{F_i : i \in [|\mathcal{T}|]\}$
```

ing a single frame of size M is

$$C_{comm} = O\left(d \log \frac{B}{\epsilon} + (M-1) d \log \frac{\Delta_q}{\epsilon} + (M-1)\log M\right), \tag{4}$$

where $d = d_i + d_o$ and for any state Z, $\|Z\| \leq B$.

The result follows analogous to that of [9]. We note that representation the cost of frames as trees incurs a cost of $(M-1)\log M$ bits as the number of rooted trees on M nodes is given by Cayley's formula as M^{M-1} [16]. This, along with Lem. 1 and Cor. 1 highlights the efficiency, need, and choice of the frame construction algorithm.

3.2. Endorser and Orderer Operations

We now briefly highlight the role of the endorser and orderer. As depicted in Fig. 1, the endorsers receive compressed frames from the computing client, decode the state vectors by parsing the frame using depth first search, and using the lattice vector decoding to recover the states from the compressed differences vectors. The endorsers then recompute the output for the approximate input and compare the reported output with the recomputed average. If the deviation is within the accepted validation tolerance, then the state is validated and dispatched to the orderer to be added to the Blockchain. If a state is invalidated, the endorsers report to the client.

Clients attempt to validate invalidated states by successively refining the reported estimates using the lattice vector quantizer and resend the updates to the endorser for validation. When sufficiently refined versions of the state are invalidated by the endorsers, the client recomputes the output from scratch.

Upon receiving the endorsements for frames, the orderer checks for consistency of the endorsements, broadcasts validated frames and appends it to the Blockchain ledger, as depicted in Fig. 1. Since validated states are stored as a hash chain, any alteration of the stored audits results in an inconsistency in the hash chain with high probability. It is also computationally infeasible to compute alterations to the audits that preserve hash consistency owing to the collision resistance of hash functions. Thus even if some adversaries in the network collude to corrupt stored audits, it is reflected as an inconsistency, both across peers, and in the hash chain.

When sufficiently large number of endorsers are chosen to validate each state, as described in [9], this allows for a simple verification mechanism in which the peer only needs to check for hash consistency across a sampled subset of peers to guarantee (3). Since the states have already been validated according to a tolerance of Δ_{val}, it might suffice in certain applications to only store a further compressed approximate of some states. That is, depending on the importance of the computation to the overall experiment, input-output pairs with lesser significance can be compressed further, using a coarser compressor to reduce the their storage cost on the blockchain. This however is both application and context-specific.

4. EXPERIMENTAL RESULTS

We study the computational trust system through the example of the OpenMalaria simulation framework [17, 18], which is an open source simulation environment used to study malaria epidemiology and efficacy of control mechanisms. In particular, we consider the experiment of identifying the optimal disease intervention policy in terms of the proportion of insecticide treated nets distributed and the fraction of geographical area covered by indoor residual spraying. For various candidate policies, the client evaluates the efficiency of the policy in terms of cost-normalized disability adjusted life years. Let us refer to the output as the *reward* corresponding to the policy. We evaluate the costs involved with different endorsement set sizes and the resulting improvements in computational accuracy.

Consider a set of policies $\{X_1, \ldots, X_n\}$ sampled i.i.d. uniformly at random from $[0,1]^2$ and let the corresponding mean rewards be $\{\bar{Y}_1, \ldots, \bar{Y}_n\}$. The compressed frames are validated, according to (2), by subsets of m endorsers operating in parallel. For these experiments we adopt an approximation error of 30% of the maximum deviation, and a maximum frame size of 100.

To understand the importance of frame construction toward efficient compression, we consider the communication and compression costs for varying validation tolerance and compression accuracy in Fig. 2. For the analysis of costs of trust, we consider validations across $m = 10$ endorsers for each state. First, we note that the baseline is set by the uncompressed communication that incurs significant number bits that effectively amount to communicating an unsigned float per dimension.

We also consider fine and coarse compression as determined by the approximation error using the MST-based frame construction and just sequential framing of states. We observe that for fine compression, the frame construction algorithm significantly reduces the the cost of compression as the sequential framing results in more checkpoints and smaller frames. Since the approximation error is low, the average number of bits per state, per dimension, per instance of communication also increases substantially with each new checkpoint being stored as is. On the other hand, the frame construction algorithm groups states more efficiently, and the delta encoding and vector quantization functions more efficiently. This however is much less pronounced under coarse compression. When the approximation error is large, the Lempel-Ziv compression [19] for the checkpoints suffices to represent frames in terms of prior encounters of sufficiently close states, and so the need for frame construction is much less pronounced.

The cost of communication is benchmarked against the corresponding cost of computation expressed in terms of the average number of computations across the client and endorsers for validation of a state in Fig. 3. The baseline for the computational cost is established by the uncompressed communication case, and as observed in the figure, decreases with increasing validation tolerance magnitude. We note that a fine compression, both with and without frame construction results in comparable computation costs, indicating that the communication costs can be reduced without an increase in the number of computations.

On the other hand, in the case of coarse compression, whereas the communication costs are significantly reduced, the number of computations per state also increases, especially for smaller validation tolerance magnitudes. This is understandable as the coarse compression results in an increased number of invalidations result-

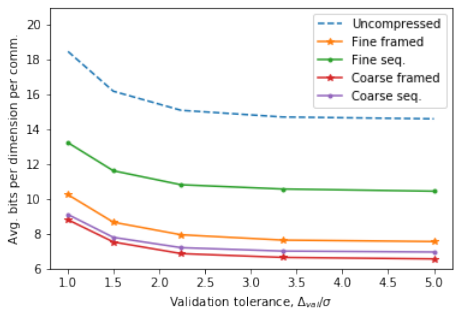

Fig. 2. Communication costs of framing

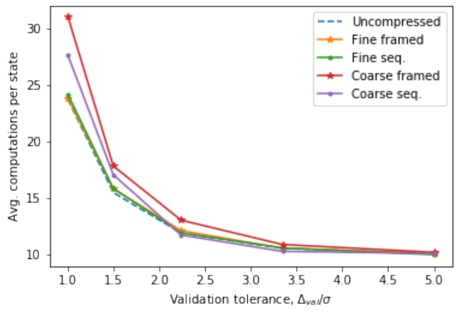

Fig. 3. Computation costs with framing

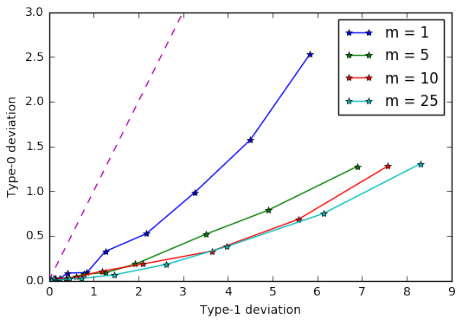

Fig. 4. Type errors by deviation: plot of T_0 vs T_1

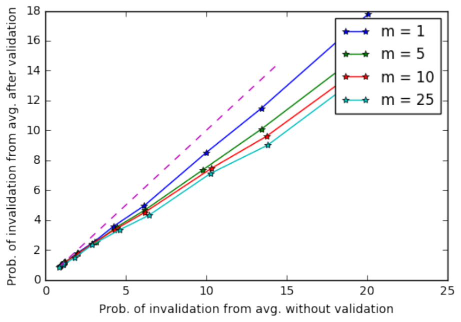

Fig. 5. Validation gains

ing from compression error. Naturally, when the validation tolerance is comparable to the approximation error, the error from the compression is effectively mitigated, resulting in similar computational cost as that of fine compression. Comparing the compression with and without frame construction, we note that the computational cost with the frame construction is larger as the coarse compression error accounts for coarse errors in the deviations across states, therein implying that the deviation in output is comparatively higher, especially when the standard deviation across outputs is small.

To study the effect of the validation scheme on computational accuracy, let us define $\delta_i^0 = \left\|Y_i - \bar{Y}_i\right\|$ and $\delta_i^1 = \left\|\tilde{Y}_i - \bar{Y}_i\right\|$, where $\bar{Y}_i$ is the expected output, $\tilde{Y}_i$ is the validated output, and Y_i is the output generated by the client prior to validation. Define $T_0 = \mathbb{P}\left[\delta_i^1 > \delta_i^0\right]$ and $T_1 = \mathbb{P}\left[\delta_i^1 < \delta_i^0\right]$, i.e., T_0 is the probability that the validation results in larger deviation from the mean reward, and T_1 is the probability with which this deviation from the mean reward is reduced.

In Fig. 4 we plot the variation of T_0 with T_1 for different sizes of endorser sets and differing validation tolerance. As the tolerance reduces, the validation mechanism reduces the deviation from the expectation more often. This is observed in Fig. 4 and we note that $T_1 > T_0$ implies that the validation improves the computation more often than not. When the tolerance Δ_{val} is large, the number of instances of invalidation is far fewer and so the fraction of computations that are altered are also much fewer.

As the number of number of endorsers increases, the average of recomputed outputs is a robust estimator of the mean of the computation and thus an alignment with these averages for validation also implies reduced deviation from the mean. Thus $T_1 \gg T_0$ when m is large, as seen in Fig. 4, highlighting the returns from the investment in more endorsers for validation.

Another metric to study the computational gains from validation is the probability that the output, as compared against the expected output would be invalidated, with and without the validation framework. To this end, let us define $\rho_0 = \mathbb{P}\left[\delta_i^0 > \Delta_{\text{val}}\right]$ and $\rho_1 = \mathbb{P}\left[\delta_i^1 > \Delta_{\text{val}}\right]$. Naturally we would like to ensure that $\rho_1 < \rho_0$, and in fact make it as small as possible.

Fig. 5 compares ρ_1 with ρ_0 as achieved by the system for various numbers of endorsers and various tolerance levels. In each case, the validation mechanism ensures that $\rho_1 < \rho_0$. Furthermore, the gains from the validation are far more pronounced upon the use of more endorsers. This is again expected as the average of more independent endorsers is a more robust estimate of the expected rewards. The gains in terms of $\frac{\rho_0}{\rho_1}$ reduce with increasing tolerance Δ_{val} as the system is more accommodating to deviations from the mean and from the endorser averages.

5. CONCLUSION

We have developed a multi-party Blockchain-based framework for verifying expensive computations, including a novel method for constructing and ordering frames of outputs that compress well. Our study is informative in practical system design for trusted distributed computing as it quantifies the cost-benefit tradeoffs allowing for appropriate system design and parameter selection. The OpenMalaria experiment conducted herein provides an example of a computational experiment where the distributed trust mechanism significantly enhances the computations. The platform can naturally be adapted to other settings such as hyperparameter tuning in machine learning that not only allows for better learning mechanisms, but also provides a pipeline to facilitate scientific reproducibility.

6. REFERENCES

[1] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, and B. C. Ooi, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1366–1385, Jul. 2018.

[2] J. Tsai, "Transform blockchain into distributed parallel computing architecture for precision medicine," in *2018 IEEE 38th Int. Conf. Distrib. Comput. Systems (ICDCS)*, Jul. 2018, pp. 1290–1299.

[3] H. I. Ozercan, A. M. Ileri, E. Ayday, and C. Alkan, "Realizing the potential of blockchain technologies in genomics," *Genome Research*, 2018.

[4] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, J. Clark, S. Meiklejohn, P. Y. A. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds. Berlin: Springer, 2016, vol. 9604, pp. 106–125.

[5] Z. Koticha, "2018: Blockchain scaling > all else," https://medium.com/thunderofficial/2018-blockchain-scaling-all-else-7937b660c08, Jun. 2018.

[6] R. K. Raman and L. R. Varshney, "Distributed storage meets secret sharing on the blockchain," in *Proc. Inf. Theory Appl. Workshop*, Feb. 2018.

[7] ——, "Dynamic distributed storage for blockchains," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2018, pp. 2619–2623.

[8] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "On-device federated learning via blockchain and its latency analysis," arXiv:1808.03949, Aug. 2018.

[9] R. K. Raman, R. Vaculin, M. Hind, S. L. Remy, E. K. Pissadaki, N. K. Bore, R. Daneshvar, B. Srivastava, and K. R. Varshney, "Trusted multi-party computation and verifiable simulations: A scalable blockchain approach," arXiv:1809.08438, Sep. 2018.

[10] C. W. Granger and R. Joyeux, "An introduction to long-memory time series models and fractional differencing," *J. Time Ser. Anal.*, vol. 1, no. 1, pp. 15–29, Jan. 1980.

[11] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Springer Science & Business Media, 2012, vol. 159.

[12] D. Mukherjee and S. K. Mitra, "Successive refinement lattice vector quantization," *IEEE Trans. Image Process.*, vol. 11, no. 12, pp. 1337–1348, Dec. 2002.

[13] Y. Liu and W. A. Pearlman, "Multistage lattice vector quantization for hyperspectral image compression," in *Conf. Rec. 41st Asilomar Conf. Signals, Syst. Comput.*, Nov. 2007, pp. 930–934.

[14] J. Conway and N. Sloane, "Fast quantizing and decoding and algorithms for lattice quantizers and codes," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 227–232, Mar. 1982.

[15] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[16] R. Durrett, *Random Graph Dynamics*. Cambridge, UK: Cambridge University Press, 2007.

[17] T. Smith, N. Maire, A. Ross, M. Penny, N. Chitnis, A. Schapira, A. Studer, B. Genton, C. Lengeler, F. Tediosi, D. de Savigny, and M. Tanner, "Towards a comprehensive simulation model of malaria epidemiology and control," *Parasitology*, vol. 135, no. 13, p. 15071516, Aug. 2008.

[18] O. Bent, S. L. Remy, S. Roberts, and A. Walcott-Bryant, "Novel exploration techniques (NETs) for malaria policy interventions," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2018, pp. 7735–7740.

[19] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inf. Theory*, vol. 24, no. 5, pp. 530–536, Sep. 1978.