Journal of New Economics and Finance

ISSN: 2789-5890(print)

Shuangqing Academic Publishing House

Contents lists available at www.qingpress.com

Journal homepage: qingpress.com/en-us/journals/10

Enhancing Blockchain Stability with LSTM-Based PoW Difficulty Adjustment: A Deep Learning Approach

Sixu Di*

Mitchell E. Daniels, Jr. School of Business, Purdue University West Lafayette, 47901, USA.

Abstract: This article introduces an advanced difficulty adjustment algorithm for the Ethereum blockchain built on the Proof-of-Work (PoW) mechanism, utilizing deep learning to maintain low volatility in block difficulty and generation time. Simulations with actual data demonstrate that the algorithm based on Long Short-Term Memory (LSTM) networks outperforms other baseline models in maintaining this low volatility. The study indicates that LSTM is more effective in controlling data volatility and can capture the trends in the original test dataset. Despite the limitations in runtime associated with deep learning methods, the research also presents potential approaches to reduce training time through incremental learning and explores the prospects of implementing this method on the Bitcoin chain.

Keywords: Blockchain, Proof-of-Work (PoW), Difficulty Adjustment Algorithm, Bitcoin

1. INTRODUCTION

Blockchain technology has surged across various sectors, including manufacturing, academic institutions, and financial organizations. At its core, blockchain aims to establish a decentralized ledger comprised of a series of blocks, each containing transaction information. The methodology

* *Sixu Di , Mitchell E. Daniels, Jr. School of Business, Purdue University West Lafayette, 47901, USA, sdi@purdue.edu*

Article history: Received November 30, 2023 Accepted January 18, 2024 Available online January 19, 2024
To cite this paper: Sixu Di (2024). Enhancing Blockchain Stability with LSTM-Based PoW Difficulty Adjustment: A Deep Learning Approach. Journal of New Economics and Finance, Vol4, Issue1, Pages 1-12.
Doi: https://doi.org/10.55375/jnef.2024.4.1

for generating these blocks hinges on consensus algorithms, with Proof-of-Work (PoW) [1], [2] currently serving as the widely adopted consensus mechanism, notably in Bitcoin [3] and Ethereum [4].

In PoW-based blockchains, miners, representing users who engage in the mining process, employ their computational power to solve intricate mathematical puzzles associated with each block. The expected time to solve such puzzles is directly proportional to their difficulty, influenced by the collective computing power, or hash rate, contributed by network participants. The miner who successfully finds a solution enjoys the privilege of propagating the newly created block to all users. The average time span between consecutive block timestamps defines the block production time. Given the direct link between block difficulty and block production time, effective difficulty control is paramount to the stability of PoW-based blockchains. An ideal difficulty control algorithm must ensure consistent block generation and prompt convergence of block production time to the desired value, regardless of fluctuations in network hash rate, thus facilitating efficient and reliable transactions.

Numerous studies have addressed Bitcoin's difficulty control, proposing enhanced algorithms for scenarios such as exponentially increasing hash rates [5], coin-hopping attack mitigation [6], and securing hash rate commitments through bonds [7]. Additionally, research efforts have delved into stochastic models for block production time and analyses of marginal distribution within Bitcoin [8]. Nonetheless, the development of a universal difficulty control algorithm applicable to all PoW-based blockchains remains a pressing challenge [9].

In general, well-designed difficulty control algorithms should adhere to specific requirements, as explicitly outlined by Ethereum [9]. These can be consolidated into two main categories:

- Simplicity & Low Memory: The algorithm should be straightforward, not relying on an extensive historical block record, and should be easy to implement. To reach consensus, block difficulty calculations should solely consider information within block headers.
- Fast Updating & Low Volatility: The algorithm should swiftly readjust block production times in response to network hash rate changes, with block difficulty exhibiting minimal fluctuations in cases of constant network hash rates. Furthermore, the difficulty control algorithm should not incentivize miners to manipulate timestamps.

Beyond difficulty control, blockchain technology has found applications beyond cryptocurrencies. Bitcoin [3], as the pioneering blockchain application, introduced decentralized currency systems and spearheaded blockchain technology. Blockchain's tamper-proof and immutable nature has extended its utility to timestamp proof [10], [11] and evidence preservation [12]. Additionally, the advent of smart contracts has catalyzed the decentralization of centralized services, most notably in supply chain management [13].

The quest for more stable blockchain services has drawn attention to various technical aspects. This comprehensive literature review particularly concentrates on enhancing block time stability, an essential requirement in blockchain design. Block time stability in PoW-based blockchains hinges on two critical variables: the network's hash rate and the block's difficulty target. Maintaining block time stability mandates precise matching between the difficulty target and the total hash rate. Consequently, the design of a proficient difficulty adjustment algorithm is pivotal to blockchain design [3].

Bitcoin's difficulty adjustment algorithm, which recalibrates the block's difficulty target every two weeks based on the generation time of the latest 2016 blocks, has enabled Bitcoin's stable operation for over a decade. The rapidly changing network hash rate is due to the volatility of crypto economic cycles. Consequently, researchers have embarked on endeavors to enhance it. Notable contributions encompass Fullmer's (Daniel Fullmer & A Stephen Morse，2017) exploration of expected value and variance in block time, along with modeling block time [14]. Beyond adjusting difficulty targets, addressing vulnerabilities stemming from malicious nodes, such as coin-hopping attacks, has become crucial [15]. To accommodate exponential hash rate increases, Kraft (D. Kraft，2016) proposed a novel difficulty adjustment algorithm exhibiting superior performance under such circumstances [5]. Additionally, Zhang (Shulai Zhang&Xiaoli Ma，2020) introduced a difficulty adjustment algorithm reliant on a two-layer neural network, using past block times as real-time input for adjustment decisions [16]. Nevertheless, assessing whether these algorithms can ensure block time stability necessitates further experimentation.

As a hard fork of Bitcoin, Bitcoin Cash boasts a superior difficulty adjustment algorithm [17]. It updates the difficulty based on the timestamps of the latest 144 blocks, providing a timelier adjustment compared to Bitcoin. Consequently, the system must possess the capability to accurately forecast the network's overall hash rate, making precise hash rate prediction a central concern. Zheng (Kaiwen Zheng et al.2020) introduced a linear predictor to forecast the entire network's hash rate and subsequently proposed a difficulty adjustment algorithm based on this methodology [18]. Bonded mining, which ensures miners' commitments to specific hash rates through bonded pledges [19], allows for real-time adjustments to the difficulty target. This approach places real-time control over the network's hash rate. While bonded mining admirably fulfills this requirement, it does introduce certain constraints for participating miners. It mandates a minimum miner hash rate of 1%, which could potentially limit the participation of smaller miners, thereby possibly encouraging miner conglomeration.

In summary, blockchain's difficulty control and stability present complex and critical challenges, spanning consensus algorithms, hash rate prediction, difficulty adjustments, and more. Future research endeavors will continue to explore innovative solutions to ensure the stable operation of

blockchains, aligning with the ever-expanding array of applications.

2. PROBLEM FORMULATION

Similar to [16], blocks are generated at the time instances $t_0 \le t_1 \le t_2 \le \cdots$ with $t_0 = 0$. The time between two instances $X_n = t_n - t_{n-1}$ is called the block production time (BPT) of the n_{th} block. Assume X_n is exponentially distributed with a rate λ, which changes according to the hash rate and difficulty ratio. By observing the previous BPT and adjusting the difficulty, the goal is to reach a constant target BPT X_{target}.

Based on [4], the difficulty is controlled to trace the hash rate in an effort to maintain a stable BPT. In Ethereum, a general difficulty control algorithm can be written as:

$$d_n = \left(1 + \mu f(X_{n-1})\right) \times d_{n-1} \tag{1}$$

$$f(X) = \begin{cases} 1 - \alpha X, \text{if } 1 - \alpha X \ge -99 \\ -99. \text{if } 1 - \alpha X \le -99 \end{cases} \tag{2}$$

where $\mu = \frac{1}{2048}$ is the step size, and $\alpha = \frac{1}{9}$ is the control factor to achieve $X_{target} = 13s$.

Although the current Ethereum difficulty control algorithm can maintain a relatively stable performance, there are still existing challenges. First, the current difficulty control algorithm is hard to be adjusted to reach a specific target BPT value. Second, because the difference between adjacent difficulties is limited, the difficulty cannot be adjusted with the hashrate synchronously and adaptively.

To mitigate such problems, as aforementioned, several approaches have been proposed to address the challenges. Zheng (Kaiwen Zheng et al.2020) introduced the use of a linear predictor to forecast the entire network's hashrate with both smoothed BPT and integrated BPT and subsequently proposed a difficulty adjustment algorithm based on this methodology [18]. However, while the prediction was capable of generating a value that was close to the actual value, in some cases, such as sudden change in hash rate, the predicted value was considerably higher or lower than the actual value.

Zhang (Shulai Zhang&Xiaoli Ma，2020) further introduced a difficulty adjustment algorithm reliant on a two-layer neural network, using past block times as real-time input for adjustment decisions [16]. This algorithm provided quite an improvement in the abnormal changes control in the process of difficulty adjustment algorithm, shedding lights on the application of deep-learning methods on difficulty prediction.

In this research, we propose an advanced difficulty control algorithm from signal processing point of view. We point out that the adjustment of difficulty with linear predictor can perform much better

when replacing integrated BPT and smoothed BPT with LSTM algorithm to capture past information of X_n, thereby improving the stability of BPT.

3. ALGORITHM DESIGN

3.1 Design of the indicator

In Zheng's (Kaiwen Zheng et al.2020) work, they used smoothed BPT and integrated BPT as indicator to calculate PoW term PT_n^{pred} in the following formula [18]:

$$d_n^p = \frac{X_{target}}{PT_n^{pred}}\left(1 + \mu f(X_{n-1})\right) \times d_{n-1}$$

The smoothed BPT is represented as followed:

$$S_n = \frac{X_n + X_{n-1} + \cdots + X_{n-ps+1}}{ps}$$

The integrated BPT is expressed as:

$$I_i = \frac{X_{(i-1)p_I+1} + X_{(i-1)p_I+2} + \cdots + X_{ip_I}}{p_I}$$

However, since both of these methods are using linear indicators, it will be dramatically affected by the fluctuation, or some extreme situation caused by X_n. To address such problem, we construct a LSTM model to better capture the information in the past.

The LSTM (Long Short Term Memory) model is a well-established deep-learning algorithm specifically designed to address long-term memory problems. It was firstly introduced to deal with natural language processing problems such as machine translation and article generation, but it also excels in time series data prediction problem in Malhotra's (Malhotra et al. 2015) [20] work for anomaly detection in time series. In our case, to overcome certain extreme possibilities, we apply LSTM to calculate the proper indicator of the PoW term.

LSTM networks consist of an input layer, one or more hidden layers, and an output layer. The number of neurons in the input layer corresponds to the number of explanatory variables (feature space), while the number of neurons in the output layer represents the output space. The distinguishing feature of LSTM networks lies in the hidden layer(s), which are comprised of memory cells. Each memory cell is equipped with three gates — the forget gate, the input gate, and the output gate — that manage and adjust its cell state.

For every step in the prediction of our indicators, we set $\{X_n\}$ as our input series, the hidden state is h, the output state is y_n, which is the indicator that we need. We modified every state in the following equation based on the structure of LSTM:

1. Input gate:

$$i_t = \sigma(W_i X_t + U_i h_{t-1} + b_i)$$

2. Forget gate:

$$f_t = \sigma(W_f X_t + U_f h_{t-1} + b_f)$$

3. Output gate:

$$o_t = \sigma(W_o X_t + U_o h_{t-1} + b_o)$$

4. Cell state updates:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c X_t + U_c h_{t-1} + b_c)$$

5. Hidden state updates:

$$h_t = o_t \odot \tanh(c_t)$$

After the renovation of LSTM, the final outcome of y_n learning result of all the information in the past by X_n.

3.2 Baseline Model

In order to further examine and test the capability of our model, we construct baseline models of RNN (Recurrent Neural Networks) and NAR (Neural Autoregressive Networks) to examine whether LSTM excels in the assignment of volatility control.

3.2.1 RNN Model

RNNs are a class of supervised machine-learning models made of artificial neurons with one or more feedback loops [21]. The feedback loops are recurrent cycles over time or sequence [22]. RNNs are extremely good at modeling sequential data for sequence recognition and prediction [23].

A simple RNN has three layers: input, recurrent hidden, and output. The input to this layer is a sequence of vectors through time. The input units in a fully connected RNN are connected to the hidden units in the hidden layer.

The basic idea of the RNN network is to use the internal multi-hop loop to ensure the continuous transmission of data, and the network updates the weight through backward propagation. However, the activation function (a) used in this network model is a sigmoid function with saturation, which causes the gradient to become very large or very small under the propagation of the activation function, so there is a problem of gradient mutation or gradient attenuation, which affects the predictability. This cannot be corrected. In order to solve the long-term data dependence of RRNs, Hochreiter and Schmidhuber (Wang Yumeng， 2014) put forward the LSTM neural network model [24] in 1997, which introduced gates to form neural units with special memory, so as to better solve the problems of gradient attenuation and gradient explosion caused by time series data in the learning process. There are three kinds of gate structures in each neural network layer of the LSTM structure as shown in Figure 1, which are output gate o_t, forgetting gate f_t and input gate i_t. These gate structures use recursive equations to constantly update the cell state C_t and activate the mapping from the input gate to the output gate.

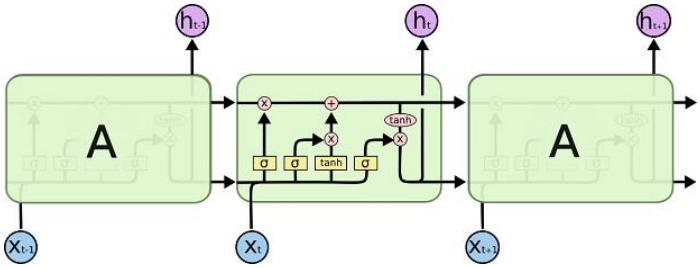


Figure 1 LSTM schematic diagram

The value of the forgetting gate is determined by the input x_t at time t and the output h_{t-1} at time t - 1, and the activation function adopted is the sigmoid function, which is expressed as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

When the value of the forgetting gate is obtained, the input gate is used to add the obtained new information to the state, so as to replace the old information in the past, and its expression is:

$$i_t = \sigma\ （W_i \cdot [h_{ti1}, X_t] + b_i）$$

Multiply the above two and add the information of forgetting gate to get a new C_t, which is expressed as:

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, X_t] + b_c)$$

The result of the output gate is the latest state C_t, the output value H_{t-1} at time T-1 and the input value X_t at current time T, which are processed by the forgetting gate and the input gate. At this time, the activation functionαis no longer a sigmoid function, but a tanh function, so that the required information can be output from the output gate.

$$o_t = \sigma(W_o \cdot [h_{t-1}, X_t] + b_o)$$

Forgetting gate f_t determines what information the neural unit abandons. By reading the states of h_{t-1} and x_t, this gate layer can output a value between 0 and 1 from the forgetting gate, where 0 represents complete abandonment and 1 represents complete retention. The value to be updated by the neural unit is determined by the input gate it, which will forget the information filtered by the gate and use the lanh function to update the state of the neural unit. Finally, the output gate ot determines the output state of the neural unit. The state of the neural unit to be output is first determined by the general sigmoid layer, and according to these states, it is compressed by tanh function between -1 and 1 [25].

In our case, we construct a simple RNN model to predict our indicator and renovate in the formula as follows:

Similarly, we assume $\{X_n\}$ to be our input series, h_t to be our output at time t, then we have:

$$h_t = f_w(h_{t-1}, X_t)$$

where f_w is a nonlinear transformation function, here we apply a sigmoid function to activate the network. The output of the model h_t is mapped to the corresponding output value $\tilde{y}_t$ through a fully connected layer f_{f_c}:

$$\widetilde{y_t} = f_{f_c}(h_t)$$

After training, the trained model is used to make predictions. The entire process can be represented using the following equation:

$$\widetilde{y_{t+1}} = f_{f_c}(f_w(h_t, X_t))$$

3.2.2 NAR Model

The Nonlinear Autoregressive (NAR) model is a class of nonlinear time series models that aims to capture the complex nonlinear relationships among variables in a time series. It is typically used for forecasting and predictive modeling in various fields, such as finance, economics, and engineering. The NAR model extends the traditional autoregressive (AR) model by introducing nonlinear transformation functions of past observations. In other words, it models the dependency of the current observation on not only the past observations but also on nonlinear transformations of those observations. This allows the model to capture more complex patterns and dependencies within the time series.

In our case, we continually apply NAR model to construct our baseline model. Similarly, we assume $\{X_n\}$ to be our input series. We construct the network as follows:

1. Firstly, transform the input sequence into a form that can be input into a neural network:

$$X_i = \frac{1}{\omega} \sum_{j=i-\omega+1}^{i} X_j$$

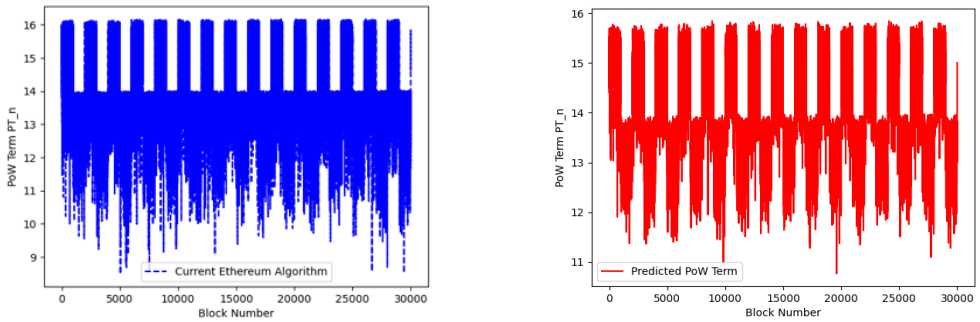
2. The input sequence is transformed linearly and non-linearly to obtain the hidden state:

$$h_i = \max(0, W_h X_i + b_h)$$

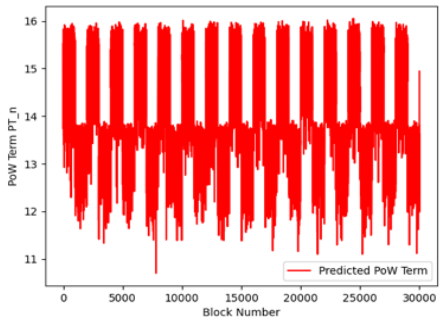
3. The output is obtained by linear transformation of the hidden state:

$$y_i = W_o h_i + b_o$$

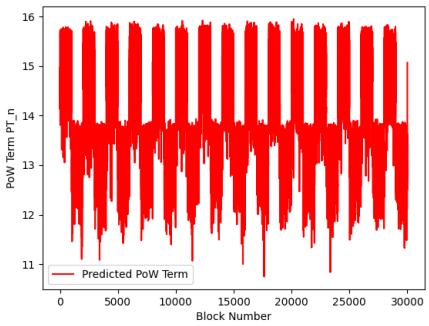
4. RESULTS



(a) (b)



(c)



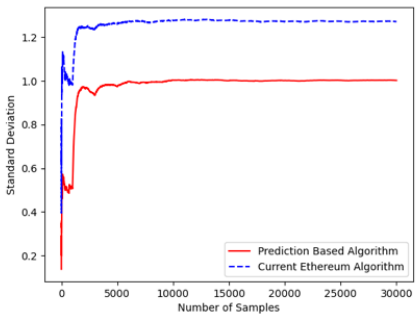
(d)

Figure 1: Real BPT (a) and Predicted PoW Term of LSTM (b), NAR (c), and RNN (d)

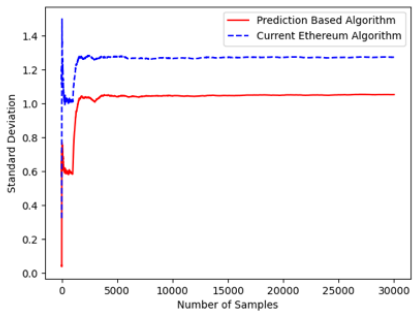
Figure 1 displays the predicted PoW Term of LSTM (Long Short Term Memory), NAR (Nonlinear Autoregressive), and RNN (Recurrent Neural Networks) in red lines. It shows that their predicted results are roughly the same. However, compared with the real BPT (the blue line), the predictions of the three algorithms have small fluctuations, with a larger lower bound, smaller upper bound, and more symmetrical structures. Moreover, focusing on the shape of their lower bound, LSTM is the most similar one to the original test data.

Table 1: Standard Deviation of prediction of LSTM, NAR, and RNN

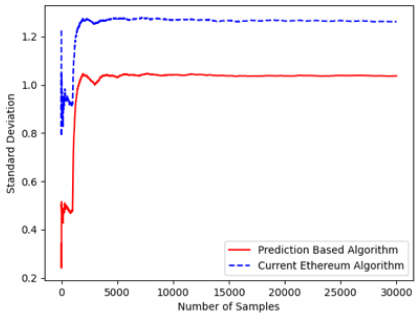
	LSTM	**NAR**	**RNN**
After prediction	0.8700	1.0411	1.0390



(a)



(b)



(c)

Figure 2: Standard Deviation of LSTM (a), NAR (b), and RNN (c)

The standard deviation of test sets before prediction is 1.2713. As Table 1 shows, the three algorithms can reduce the volatility of the predicted data, especially for LSTM. Figure 2 demonstrates the details of the changes in standard deviation as the amount of sample data increases. The red lines are the predicted results of LSTM, NAR, and RNN, respectively. The blue lines are the results of corresponding test datasets before prediction. Figure 2 has the additional information that as the amount of data increases, the standard deviation converges. Among them, the convergence value of LSTM is slightly smaller than 1, which is the opposite of NAR and RNN, indicating less volatility of BPT as we attempt to apply LSTM models. Such results are meaningful when dealing with tasks like prediction of BPT as LSTM are more capable of absorbing information in the past.

5. DISCUSSION

The objective of the report is to explore the volatility control capabilities of LSTM, using the other two machine learning algorithms, RNN and NAR, as the baseline. In the results section, we compare their prediction results and prediction standard deviation with the original test set. Among the algorithms, LSTM has the best performance since it is more effective in controlling the volatility of data; meanwhile, it can capture the changing trend of the original test dataset.

Our work also provides a solution for current difficulty adjustment algorithms. The suggested technique aimed to be reliable enough to minimize the deviation of difficulty variations, resulting in more stable generation of blocks. Our method maximized the probability to produce equal and consistent difficulty outputs from chains in network.

However, more issues could also be investigated. Our deep learning approach has its natural limitation in its running time, and it's of significance to introduce incremental learning to reduce training time. Also, it will be more popular if this method can be implemented on the chains of Bitcoin, in which we believe our strategy will also work well.

6. CONCLUSION

In this paper, we propose an advanced difficulty control algorithm for PoW-based Ethereum using a deep-learning method. Simulations based on real data reveal that our LSTM-based algorithm preserves the low volatility of the block difficulty as well as its producing time compared with other baseline models.

REFERENCES

[1] C. Dwork and M. Naor(1992). "Pricing via processing or com-batting junk mail," in Annual International Cryptology Conference. Springer, pp. 139–147.

[2] A. Back et al.(2002). "Hashcash-a denial of service countermeasure," [Online]. Available: http://www. hashcash.org/papers/amortizable.pdf

[3] Nakamoto, S. (n.d.). Bitcoin: A Peer-to-Peer Electronic Cash System. Bitcoin. https://bitcoin.org/en/bitcoin-paper

[4] G. Wood et al.(2014). "Ethereum: A secure decentralised generalised transaction ledger," Ethereum project yellow paper, vol. 151, no. 2014, pp. 1–32, 2014.

[5] D. Kraft(2016), "Difficulty control for blockchain-based consensus systems," Peer-to-Peer Networking and Applications, vol. 9, no. 2, pp. 397–413.

[6] D. Meshkov, A. Chepurnoy, and M. Jansen(2017), "Short paper: Revisiting difficulty control for blockchain systems," in Data Privacy Management, Cryptocurrencies and Blockchain Technology. Springer, pp. 429– 436.

[7] G. Bissias, D. Thibodeau, and B. N. Levine (2019) . "Bonded mining: Difficulty adjustment by miner commitment,"in Data Privacy Management, Cryptocurrencies and Blockchain Technology. Springer, pp. 372–390.

[8] D. Fullmer and A. S. Morse(2018), "Analysis of difficulty control in bitcoin and proof-of-work blockchains," in 2018 IEEE Conference on Decision and Control (CDC). IEEE, pp. 5988–5992.

[9] Design rationale of Ethereum. [Online]. Available: https://github.com/Ethereum/wiki/wiki/Design-Rationale#difficulty-update-algorithm

[10] Thomas Hepp, Alexander Schoenhals, Christopher Gondek, and Bela Gipp(2018). Originstamp: A blockchain-backed system for decentralized trusted timestamping. it-Information Technology, 60(5-6):273–281.

[11] Yuan Zhang, Chunxiang Xu, Nan Cheng, Hongwei Li, Haomiao Yang, and Xuemin Shen(2019). Chronos+: An accurate blockchain-based timestamping scheme for cloud storage. IEEE Transactions on Services Computing, 13(2):216–229.

[12] Wenqi Yan, Jiachen Shen, Zhenfu Cao, and Xiaolei Dong(2020). Blockchain based digital evidence chain of custody. In Proceedings of the 2020 The 2nd International Conference on

Blockchain Technology, pages 19–23.

[13] Menghui Lou, Xiaolei Dong, Zhenfu Cao, and Jiachen Shen(2021). Sescf: A secure and efficient supply chain framework via blockchain-based smart contracts. Security and Communication Networks.

[14] Daniel Fullmer and A Stephen Morse(2018). Analysis of difficulty control in bitcoin and proof-of-work blockchains. In 2018 IEEE Conference on Decision and Control (CDC), pages 5988–5992. IEEE.

[15] Dmitry Meshkov, Alexander Chepurnoy, and Marc Jansen(2017). Short paper: Revisiting difficulty control for blockchain systems. In Data Privacy Management, Cryptocurrencies and Blockchain Technology, pages 429– 436. Springer.

[16] Shulai Zhang and Xiaoli Ma(2020). A general difficulty control algorithm for proof-of-work based blockchains. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3077–3081. IEEE, 2020.

[17] Amaury Séchet(2017). Implement simple moving average over work difficulty adjustment algorithm. D601, Accepted Proposal for Bitcoin Cash. https://reviews.bitcoinabc.org.

[18] Kaiwen Zheng, Shulai Zhang, and Xiaoli Ma(2020). Difficulty prediction for proof-of-work based blockchains. In 2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pages 1–5. IEEE.

[19] George Bissias, David Thibodeau, and Brian N Levine(2019). Bonded mining: Difficulty adjustment by miner commitment. In Data Privacy Management, Cryptocurrencies and Blockchain Technology, pages 372– 390. Springer.

[20] Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015, April). Long Short Term Memory Networks for Anomaly Detection in Time Series. In Esann (Vol. 2015, p. 89).

[21] S. Haykin, Neural networks: a comprehensive foundation. Prentice Hall PTR, 1994.

[22] T. Mikolov, M. Karafi´at, L. Burget, J. Cernock `y, and S. Khudanpur(2010), “Recurrent neural network based language model.” in Interspeech,vol. 2, p. 3

[23] Y. Bengio, P. Simard, and P. Frasconi(1994).“Learning long-term dependencies with gradient descent is difficult,” IEEE transactions on neural networks, vol. 5, no. 2, pp. 157–166.

[24]Wang Yumeng(2014). Financial volatility analysis and prediction based on symbolic time series analysis [D]. Tianjin University.

[25]Huang Min Hao(2019). Analysis of stock price trend based on time series [J]. Modern Marketing: Academic Edition, (12):2.