An Implementation of Lattice-based Proof-of-Work on Blockchain

A. Krivulčík[1] and S. Ricci[1]

[1]Department of Telecommunications, Brno University of Technology, Technicka 12, Brno, 616 00, Czech Republic

E-mail: xkrivu00@vut.cz, ricci@vut.cz

Abstract—Cryptocurrencies and blockchain are skyrocketing in recent years. They rely on Proof-of-Work (PoW) mechanisms for generating a new transaction and turn this process into ”work” (i.e., puzzles) where miners are paid for. With the advent of quantum computers, also PoW starts to migrate to post-quantum cryptographic alternatives. To the best of our knowledge, we present the first implementation of a lattice-based PoW based on the Shortest Vector Problem (SVP). By implementing in Python 3 and with the use of the NumPy library, we wrote a software that uses this concept on an artificial blockchain and demonstrates its real-world use. Even if this proposal has drawbacks on GPU optimisation and storage requirements, it shows its potential in use. The experimental results show that by balancing the size and generated range of a given matrix and vector, we can easily manipulate the time required to solve SVP challenge.

Keywords—Proof-of-Work, Lattice-based Cryptography, Blockchain, SVP problem, Post-quantum Cryptography, Python

1. INTRODUCTION

Cryptocurrency usage in the global market has grown significantly over the past decade. MarketWatch [1] estimates that by 2030, cryptocurrency usage will have an annual growth rate of 30%. One of cryptocurrency security primitives is the PoW and it is used to provide a certain amount of computational effort that has been spent. A PoW consists of solving a computational puzzle. The puzzle is very hard to compute but its correctness is easy to verify. PoW that is used in Bitcoin, Ethereum, and many other cryptocurrencies, consists of computing the hash function of a given block with the addition of a random nonce to achieve a hash with a required structure. Anyone can read and verify the PoW, but the data cannot be changed or tampered with after it is written to the blockchain. Hash functions such as SHA-256 for Bitcoin and Keccak-256 for Ethereum are proven to be secure and reliable.

However, a quantum era has been unfolding and Grover’s search algorithm simplifies the collision and symmetric key brute force search to sub-linear complexity with an increase of keys and parameters sizes [2]. In this context, cryptocurrencies also start to migrate to post-quantum algorithms and PoW can be based on post-quantum problems such as the SVP problem on a lattice. Lattice can be imaged as a finite number of vectors in a given dimension. The problem called SVP is about finding the shortest vector in a given lattice, or matrix, so that its Euclidean norm (in other words length) is the shortest.

The lattice vectors are randomly generated with very big numbers, thus making computations harder. Therefore, the PoW hash function challenge can be replaced with an SVP challenge on a randomly generated lattice. To the best of our knowledge, we present the first implementation of the algorithm proposed in [3]. Moreover, experimental results on the lattice-based PoW efficiency and a comparison with currently deployed Bitcoin PoW are shown.

2. IMPLEMENTATION

The implementation is made in the Python 3 programming language and is publicly available in the repository[1]. A small change in the NumPy library is made to make computations over such high numbers as $1.5 * 10^{540}$ possible. In particular, the Python `object` needed to be replaced with an object from library Decimal.

Algorithm 1 depicts the steps needed to run the PoW. In particular, the first step is to generate a random prime of a given size. By utilising native Python library Crypto.Utils and its function `getPrime(n)`, we

[1]https://github.com/Mysfrit/Lattice-PoW

Algorithm 1 PoW challenge structure

- $n \leftarrow 150$
- $\alpha \leftarrow$Target magnitude from already solved vectors (SVP challenge).
- $Gen(p, B, \alpha)$, where p is prime of length $10n$, create Base matrix B with uniform distribution $U(0 \in [p-1])$. Return $c = (\alpha, n, B, p)$
- $Solve(c)$

while $||V|| \leq \alpha * p^{1/n}$ **do**

- Generate a non–zero vector v of size n.
- $V = B * v$
- Calculate norm $||V||$

end while

- Return (c, v, V, α, B), Write values to blockchain
- $Verify(c, v, V, \alpha, B)$

receive a random prime number of bit size n. The second step is to create a matrix basis B. For this step, we are using the library NumPy. First with function `numpy.zeros(n)`, we create matrix of size $n \times n$ with zeros. After that, we add ones on diagonals. Again by utilising library NumPy, we generate uniformly distributed random values from range $\langle 0, p-1 \rangle$ and add them to the first row of the matrix. Then we add the generated prime on the first row and first column.

The third and last step is to start generating random vectors. For this, we are again using NumPy function `numpy.random.randomint(low, high,n)`, where low and high are thresholds for values in the vector and n is the size of a given vector. Matrix multiplication with the generated vector and matrix basis is applied and the magnitude of the output vector is calculated. The final step is to compare the magnitude with a required value and loop this process until we find a satisfactory vector[2].

2.1. Speed and memory usage

In this section, we introduce the speed optimisation achieved and compare the memory usage of the proposed PoW with Bitcoins PoW. In fact, our code is optimised to do so with parallel computing, utilising all possible CPU resources. For the time being, the biggest bottleneck is generating random numbers for the vector. With the benchmark CPU, we could achieve around 2500 guessed vectors per second (i.e., 25000 generated numbers per second with $n = 100$).

Bitcoin block size is around 1 megabyte including transactions and other variables. At first, it appears to be manageable, but if this network is scaled into five years, every single kilobyte can make a difference. Referring to the paper [3], there are several variables:

- n - recommended around 150
- prime p - Recommended size is $10n$ bits
- Base matrix B - consisting of prime p and uniformly generated numbers $x_i \in U(0 - [p-1]$, with size $n \times n$

By taking these variables into account, the required storage space for a verification pass from a few bytes (in Bitcoins) to around tens or hundreds of kilobytes. Therefore, just a verification part of one block with the matrix of size 150×150 would take 99.3 KBytes. For example, a Bitcoin blockchain of size 379 GB with 725000 blocks would be around 48% bigger with the use of a lattice PoW. Of course, if this PoW would be used on a real network, the parameters would need to be even bigger to compensate for larger computational power. That could lead to a significant increase in size.

3. EXPERIMENTAL RESULTS

The complexity of computation, i.e., finding a vector with required properties, is linked to the range of a generated vector. For example, taking into account a matrix basis of size 120×120 with range of generated vector $\langle -470, 470 \rangle$, the challenge is solved in around 16.96 seconds[3], with around 2500 guessed vectors per second. But a vector with a range of $\langle -480, 480 \rangle$ needs on average 2.2 times longer. These values are plotted in Figure 1. As with Bitcoin, we can artificially increase the difficulty of finding the solution

[2] https://www.latticechallenge.org/svp-challenge/halloffhp

[3] Calculated on a CPU Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz

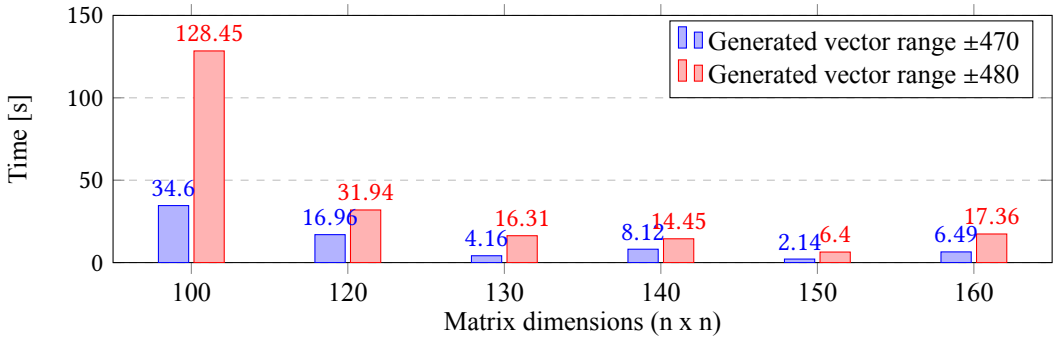

Figure 1: Comparison of time taken to find a vector with sufficient norm and different vector ranges.

to slow down the network computations. With this knowledge, we can estimate the behaviour of the network. Note that increasing the size n of a given basis and vector does not necessarily increase the computational time, so it should be stable at 150. For instance, if the network uses 10–minute intervals as in Bitcoin, the network would need to adjust according to the time required to solve the latest block and adjust the vector generation range accordingly. However, it has be noted that finding a vector is purely a game of chance. There could be a faster or slower solution for the same vector at two different times.

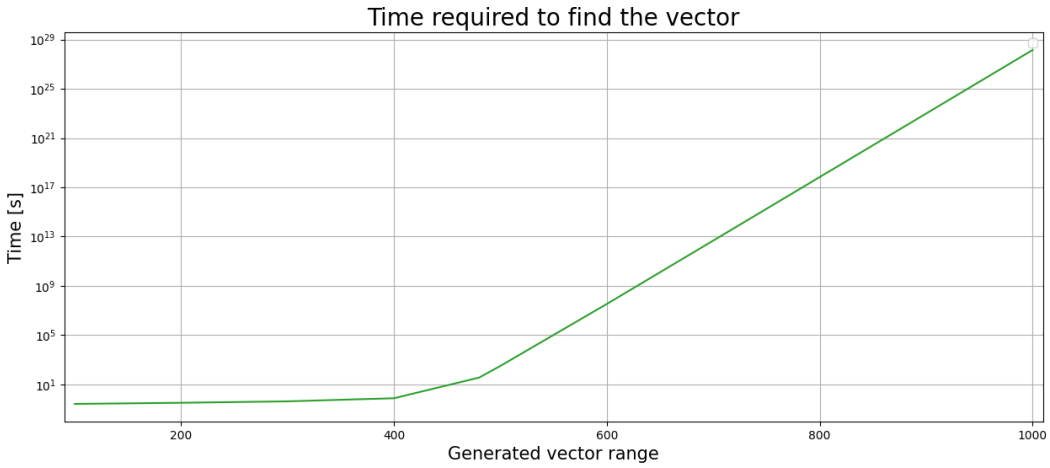

Figure 2: Time required to find the vector.

Figure 2 shows our challenge solving times for vectors ranges smaller than $\langle -500, 500 \rangle$ and our estimation from this threshold onward. In particular, comparing our results to the SVP challenge website[4], we estimated that finding a solution of a targeted generated vector of range $\langle -1000, 1000 \rangle$ would take on average $4 * 10^{20}$ years. To put this time into perspective, the universe is old $1.6 * 10^{10}$ years. At first glance, this number should be non-achievable for sole individuals, but we need to remember that the network may consist of a hundred thousand users trying to solve a given problem. Furthermore, these vectors are easier to solve on faster computers. For example, a vector of size 180 with generated range can be solved on 2 Xeon processors with 4 Nvidia GPU cards in 28 days by a single user.

4. CONCLUSION

Cryptocurrencies are being used more frequently every year. With the advent of quantum computers, their security and cryptographic primitives will start to migrate to post-quantum secure algorithms. To the best of our knowledge, we present the first implementation of the lattice-based PoW proposed in [3]. Moreover, experimental results on the lattice-based PoW efficiency and a comparison with currently deployed Bitcoin PoW are carried out.

[4] https://www.latticechallenge.org/svp-challenge/halloffame.php

As a result of our analysis, we conclude that solving the SVP problem depends more on the range of generated guessed vector rather than on the dimension of the lattice matrix. Moreover, we estimate that the computational time to get a resulting vector will need to be more than double by increasing the vector size by a factor of 10. Even if a real-life implementation of this proposal can lead to blockchain resistance to quantum computer attacks, we predict that the higher memory consumption of the whole blockchain due to storing such large numbers will become a drawback that needs to be resolved.

Furthermore, there is a possibility of deadlock when a generated lattice does not have a vector that meets the required magnitude. In this case, the network should regenerate a new matrix and restart the process after some arbitrary amount of time.

As future work, we plan to speed up the matrix generation running the computation on a graphic card from Nvidia, where guesses could achieve values over 200 million guessed numbers per second (in our case 2 million vectors per second)[5].

REFERENCES

[1] Cryptocurrency Market Size, Growth 2022 Global Opportunities, Trends, Regional Overview, Global Leading Company Analysis, And Key Country Forecast to 2030. Market Watch [online]. 2021, December 22 2021 [cit. 2022-03-06]. Available from URL:https://on.mktw.net/3hU5nz0

[2] BERNSTEIN, Daniel J. Introduction to post-quantum cryptography. Post-quantum cryptography. Springer, Berlin, Heidelberg, 2009. 1-14. Available from URL: doi:10.1007/978-3-540-88702-7_1

[3] BEHNIA, Rouzbeh, Eamonn W. POSTLETHWAITE, Muslum Ozgur OZMEN and Attila Altay YAVUZ. Lattice-Based Proof-of-Work for Post-Quantum Blockchains. Data Privacy Management, Cryptocurrencies and Blockchain Technology [online]. Cham: Springer International Publishing, 2022, 2022-01-01, 310-318 [cit. 2022-03-01]. Lecture Notes in Computer Science. ISBN 978-3-030-93943-4. Available from URL: doi:10.1007/978-3-030-93944-1_21

[4] CHEN, Lily, Stephen JORDAN, Yi-Kai LIU, Dustin MOODY, Rene PERALTA, Ray PERLNER and Daniel SMITH-TONE. Report on Post-Quantum Cryptography [online]. 2016, 1–3 [cit. 2021-11-17]. Available from URL: http://dx.doi.org/10.6028/NIST.IR.8105

[5] https://developer.nvidia.com/curand