Conquering Generals: an NP-Hard Proof of Useful Work

Angelique Faye Loe
Royal Holloway, University of London
Information Security Group
angelique.loe.2016@rhul.ac.uk

Elizabeth A. Quaglia
Royal Holloway, University of London
Information Security Group
elizabeth.quaglia@rhul.ac.uk

ABSTRACT

Proof of Work systems are used in cryptocurrencies to obtain consensus in distributed peer-to-peer systems that share no trust. Miners of cryptocurrency compete by engaging in the Proof of Work to solve a cryptographic challenge. The first to successfully provide a solution to the challenge wins by minting new currency. The process of mining also simultaneously prevents double-spending through the creation of an append-only distributed database known as the blockchain. The most widely adopted Proof of Work is the *Hashcash* scheme and the most widely deployed miners are ASIC-based. Despite the popularity of Hashcash, two issues are commonly identified its use. Firstly, the high energy consumption of the scheme is perceived as wasteful because the solutions found provide no useful output, and secondly, the computational complexity class of the scheme is not formally known. Based on these deficiencies, we propose a novel Proof of Work system which achieves the following goals:

- to provide a fiscally incentivized platform for algorithm research that aims to optimize an NP-Hard computational problem. This provides indirect insight into the P Versus NP Clay Institute Millennium problem, thus providing useful output.
- to construct a challenge within a known hard computational complexity class.
- to ensure the Proof of Work created is inclusive of ASIC hardware.

Our proposal is a hybrid Proof of Work system that initially uses the Hashcash scheme and which subsequently constructs an instance of the NP-Hard Travelling Salesman Problem. We build on the ambitions of others to develop Proofs of *Useful* Work. We differentiate our paper from related work as the first to consider the current capital investment into ASIC hardware, thus including them in our proposal.

1 INTRODUCTION

The Hashcash-based [2] Proof of Work exists in Bitcoin, and several other cryptocurrencies, to serve the two main purposes of minting new currency and providing consensus in a peer-to-peer network. Peers on the network mint new currency through a process known as mining. Mining can be described as expending electric energy by iteratively running the SHA256 hashing algorithm on a *candidate block header*, consisting of *unconfirmed transactions* taken from a *memory pool* and concatenating a nonce until the desired output of several leading zero's is obtained [1].

Mining is a competitive process in which miners race each other to be the first to find a solution to the computational challenge. The process of mining also simultaneously ensures consensus by generating the immutable distributed ledger data structure, commonly referred to as the blockchain. The blockchain is append-only and it contains the transaction history between the peers on the network. It provides cryptographic assurance that every entry is irrevocably scribed in its publicly viewable data structure. It is precisely because the blockchain provides a non-repudiatable, publicly distributed database of all historical transactions that fraudulent transactions known as double-spending are broadly mitigated.

Since the *Genesis Block* of Bitcoin was mined, the aggregated hash rate of the participating miners has grown exponentially [5]. As a result of this growth, cryptocurrency mining has been the focus of debate in relation to the energy consumption requirements to perform the underlying Proof of Work. The *"Bitcoin Energy Consumption Index"* estimates the energy consumption of Bitcoin miners and compares this to various nation states and performs a comparison on a per-transaction-basis to other payment systems, such as VISA [14]. In Table 1 we present a sample of cryptocurrencies and notable academic proposals based on the classification of their underlying Proof of Work schemes, and their current market capitalization in BTC. Rows 3 - 7 outline systems which address the energy waste issue by establishing Proofs of Useful Work, whilst rows 6 - 7 additionally address the issue of heuristic hardness associated with the Hashcash Proof of Work [3]. Enumerating each altcoin goes beyond the scope of this paper and [7] provides an exhaustive list of all currently traded cryptocurrencies, including those with Proofs of Useful Work.

The largest difference our proposal incorporates when compared to other Proof of Useful Work systems is the inclusion of ASIC mining hardware. If we consider one of the top end ASIC miners on the market as of March 2018, it claims a hash rate of 14 TH/s and a hardware cost of 3000 USD [6]. Based on this data we develop the following formula $(\frac{d \text{ TH/s}}{14 \text{ TH/s}})(3000 \text{ USD}) = \lambda \text{ USD}$, where λ is the estimated capital expenditure on ASIC hardware and d is the average daily hash rate. Table 2 and Figure 1 summarize our findings [2].

Clearly, the capital investment into ASIC hardware runs into the billions of USD. If we consider the estimated 4.3 billion USD currently invested in ASIC hardware, this is equivalent to the Arkansas 2018 state budget for education and transportation [8].

CRYBLOCK '18, June 15, 2018, Munich, Germany

ACM ISBN 978-1-4503-5838-5/18/06…$15.00
https://doi.org/10.1145/3211933.3211943

[1] For a detailed view of this process and clarification of definitions refer to [1]. History of Proof of Work algorithm development prior to the landmark paper [23] can be found most notably in [2, 15, 19]. History of digital cash and "*pre-Nakamoto*" cryptocurrency papers can be found in [9, 12, 16, 25].

[2] We export the "Hash Rate" data from [5] for our calculations of average daily hash rate.

Table 1: Current Proof of Work Variants

Row	Proof of Work Variant	Proof of Work Sub-Variant	Use Case	Perceived Benefits	Perceived Limitations	Market Cap (BTC) [7]
1	Hash Based	SHA256 output less than target value	Bitcoin [23]	original cryptocurrency Proof of Work, widely used, matured	computational complexity hardness heuristic, energy consumption	17,067,359
2	Hash Based	Scrypt key derivation	Litecoin [27]	widely used, matured	computational complexity hardness heuristic only, energy consumption	1,144,753
3	Number Theoretic	Cunningham Chain, Bi-Twin Chain search	Primecoin [21]	academic application of solutions	low market cap, alienates ASIC miners	1,367
4	Number Theoretic	Prime gap search	Gapcoin [17]	academic application of solutions	very low market cap, alienates ASIC miners	55
5	Resource Efficient Mining	trustworthy reporting on inherently useful CPU workloads	academic only [28]	fractional percentage Hash Based energy waste	currently only academic, possible implications of Spectre bug, alienates ASIC miners	0
6	Graph Theoretic	Cycle and structure search in large graphs	academic only [26]	academic application, constructed problems have strongly conjectured computational complexity classes	currently only academic, alienates ASIC miners	0
7	Linear Algebra Based	Orthogonal Vector Problem	academic only [3]	academic application, verbose fine-grained computational complexity class conjecture provided	currently only academic, alienates ASIC miners	0

We thus reason that Proof of Useful Work systems that do not consider the use of ASIC hardware to be quixotic, and likely to face opposition in adoption. They additionally risk industrial scale waste of purpose-built physical hardware, that by design, cannot be repurposed. It is specifically for this reason that we do not consider Proof of Stake or Proof of Space systems in our proposal as these schemes purposely exclude ASIC miners.

We therefore propose a hybrid two-round Proof of Useful Work scheme with a deterministic time interval which continues the use of ASIC hardware in the first round and subsequently constructs an instance of the NP-Hard Travelling Salesman Problem. Our hybrid approach ensures the current use of ASIC miners that have already been purchased and deployed, thus avoiding the latter noted waste scenario. Additionally, a hybrid approach addresses ASIC mining energy usage by limiting the Hashcash-stage to a single round, bound by a deterministic time interval. Furthermore, by including an NP-Hard computational problem we build upon the ideas presented in [3] by offering a challenge within a well-known computational complexity class whilst presenting an incentive to enhance research into algorithm design, rather than the current sole ambition of increasing brute force hashing power. Also, we argue that a hybrid scheme could shift the dependence from ASIC hardware in a gradual manner thereby making adoption of this Proof of Useful Work scheme less contentious. We finally recall that, should anyone find an algorithm to "*solve*" the TSP the implications would have far reaching consequences because: "*NP-Complete problems have the intriguing property that if any NP-Complete problem can be solved in polynomial time, then every problem in NP has polynomial time solution, that is P = NP* [11]."

2 BACKGROUND, ASSUMPTIONS AND PREREQUISITES

2.1 Background

As we aim to construct instances of the Euclidean TSP, the subsequent claims attest to the computational hardness associated with seeking optimal solutions:

Claim 1: There is a polynomial time reduction from the Hamiltonian Cycle Problem to the TSP, Figure 2 [11, 20].

Claim 2: The Travelling Salesman Decision Problem is NP-Complete in the Strong Sense [18].

Claim 3: It is possible to construct the NP-Hard Travelling Salesman Optimization Problem from the Travelling Salesman Decision Problem [18] [3].

2.2 Assumptions

Our construction of the TSP is on a Euclidean plane, therefore we recall the following:

Definition 1 *The TSP (Optimization)* [18]. Input: Set Ω of n cities $\{\omega_1, \ldots, \omega_n\}$, distance $d(\omega_i, \omega_j) \in \mathbb{R}^+$, where $1 \leq i, j \leq n$ and $i \neq j$, for each pair of cities $\omega_i, \omega_j \in \Omega$. Output: The minimal tour length of a permutation $< \omega_{\pi(1)}, \omega_{\pi(2)}, \ldots, \omega_{\pi(n)} >$ of Ω, i.e. find a permutation that gives the following:

$$\min\left\{ \left(\sum_{i=1}^{n-1} d(\omega_{\pi(i)}, \omega_{\pi(j)}) \right) + d(\omega_{\pi(n)}, \omega_{\pi(1)}) \right\}$$

Property 1 *The Triangle Inequality.* For Cities $\omega_i, \omega_j, \omega_k$, where $1 \leq i, j, k \leq n$, the following property holds:
$d(\omega_i, \omega_j) \leq d(\omega_i, \omega_k) + d(\omega_k, \omega_j)$.

Property 2 *Symmetric TSP.* We also assume that the distances between any pair of cities is symmetric, that is: $d(\omega_i, \omega_j) = d(\omega_j, \omega_i)$.

[3] The formal proofs for these claims can be found in the respective references.

Table 2: Estimated Capex on ASIC Hardware

Time Interval	d average daily hash rate (TH/s)	λ est. capex on ASIC hardware (USD)
Jan. 1 - Jun. 30, 2017	3,778,265	809,628,108
Jul. 1 - Dec. 31, 2017	8,826,086	1,891,304,038
Jan. 1 - Mar. 4, 2018	20,146,175	4,317,037,509

Table 3: Algorithm Classification for the TSP

Algorithm Classification	Example Algorithm	Big O run time
Exhaustive Search	Näive Algorithm	$O(n!)$
Exact	Held-Karp Algorithm	$O(2^n n^2)$
Approximation	Christofides Algorithm	$O(n^4)$
Heuristic	LKH Algorithm	$O(n^{2.2})$

2.3 Prerequisites

Our proposal relies on the following prerequisites that miners must have:

- synchronized clocks obtained via reputable NTP Servers, (see section 3.4 for further discussion)
- the computational resource to run:
  (i) the SHA256 cryptographic hash function
  (ii) an algorithm of their choice to optimize the TSP
  (iii) a sorting algorithm
  (iv) an IEEE 754-2008 [24] floating-point arithmetic operation to find square roots using the Pythagorean theorem to calculate ℓ_2 norms.

3 PROPOSED PROOF OF USEFUL WORK

A Proof of Useful Work [3] typically consists of three algorithms (Gen, Solve, Verify), where Gen(x) takes an input x and generates a challenge c, Solve(c) solves the challenge c, outputting a solution s, and Verify(c, s) is a (possibly randomized) algorithm that verifies the solution s to c. The detailed formal requirements of Proofs of Useful Work, namely, *Efficiency*, *Completeness*, *Soundness*, *Hardness* and *Usefulness* can be found in [3].

Our proposed Proof of Useful Work relies on the efficiency of the Gen and Verify algorithms running in polynomial time. The Solve algorithm satisfies the notion of associated hardness, with a running time of $O(t(n))$, where the values of $t(n)$ are dependent on algorithm selection as outlined in Table 3. Currently the only algorithm (excluding the naïve) to guarantee finding an optimal solution to the TSP is the Held-Karp, which has an intractable run time. Formal security proofs for our proposed construction are presented in [22].

3.1 Notation

ω_i denotes a city, and x is the input of the problem. In Round 1 $|x|$ = 1MB (as per the bitcoin maximum *candidate block* size), in Round 2 n will indicate the number of cities used to construct the Euclidean TSP, and m is the number of miners online, where $n, m \in \mathbb{Z}^+$. The deterministic time interval for each blocks Proof of Useful Work is $[t_s, t_c]$, where $t_c - t_s$ = 10 minutes. H is the keyless hash function SHA256, η is a cryptographic nonce and ψ is a SHA256 cryptographic hash output. The subscripts are defined as 1 or 2 to identify the round, (i) to represent the identification of the miner, (j) to represent the block number in the blockchain and (k) to represent the counter number on the nonces. The superscripts are: (ℓ), in Round 1 represents the lowest value in the set of all attempted hashes, (ℓ) in Round 2 represents the permutation of n cities giving the lowest sum, and $(\mathcal{L})$ represents the lowest of the (ℓ) solutions.

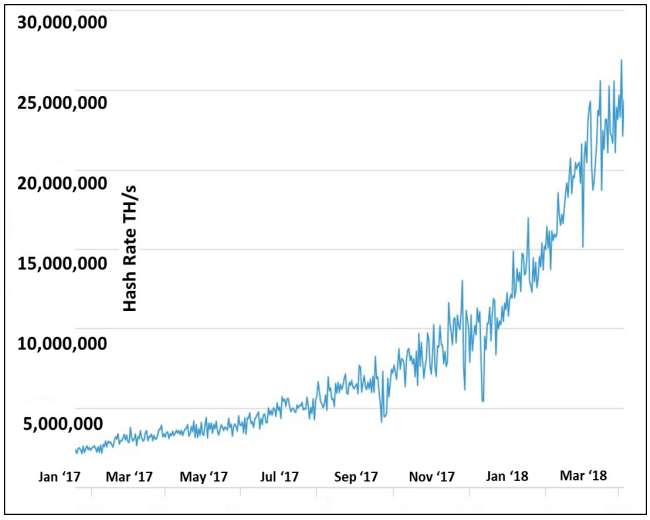

Figure 1: Jan. '17 - Mar. '18 Hash Rate Chart TH/s vs. date [5]

3.2 Example

For instance, $s_{x_1(i=33)(j=512253)(k=67987)}$ is a solution output by the miner with identification (33), and where x_1 indicates the input of x relevant to Round 1. (512253) indicates the block number in the blockchain, and (67987) indicates the counter number of the attempted nonce.

3.3 Conquering Generals: an NP-Hard Proof of Useful Work

Conquering Generals is a two round Proof of Useful Work. In the first round the Hashcash scheme is used and subsequently in the second round an instance of the Travelling Salesman Problem is constructed. The set of $\{1, \ldots, m\}$ miners online will perform the following:

Round 1

1.1 $\text{Gen}_1(x_1)$ each miner will generate a unique challenge $c_{x_1(i)(j)}$ by selecting $|x|$ = 1MB of unconfirmed transactions from the memory pool to create a candidate block, Figure 5.

1.2 $\text{Solve}_1(c_{x_1(i)(j)})$, will operate the double iterated SHA256 algorithm as follows:

Attempt 1: $H(H(c_{x_1(i)(j)} || \eta_{x_1(i)(j)(1)})) = s_{x_1(i)(j)(1)}$

Attempt 2: $H(H(c_{x_1(i)(j)} || \eta_{x_1(i)(j)(2)})) = s_{x_1(i)(j)(2)}$

$\vdots$

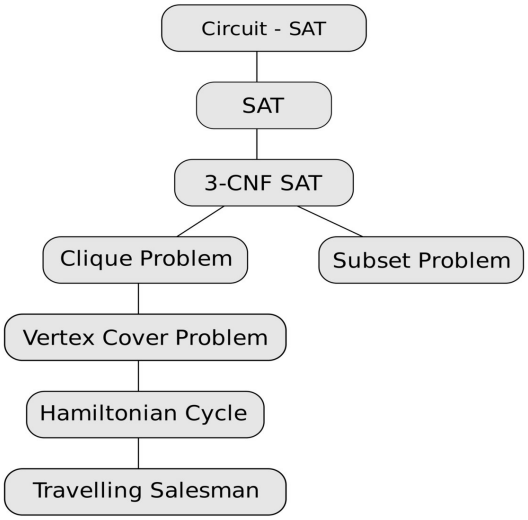

Figure 2: Polynomial reductions from SAT to the TSP [20] [11]

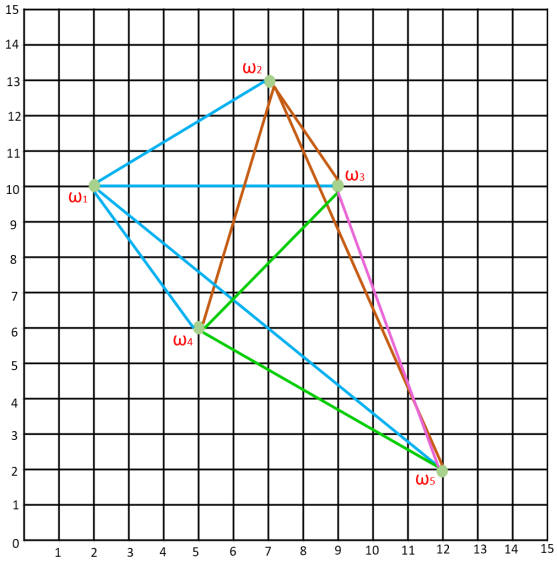

Figure 3: Visual of toy example of Gen$_2(x_2)$

Attempt @t_1: $H(H(c_{x_1(i)(j)}||\eta_{x_1(i)(j)(@t_1)})) = s_{x_1(i)(j)(@t_1)}$

1.3 Propagate solution $s^{(\ell)}_{x_1(i)(j)} = \min\{s_{x_1(i)(j)(1)}, s_{x_1(i)(j)(2)}, \ldots, s_{x_1(i)(j)(@t_1)}\}$ and corresponding $c_{x_1(i)(j)}||\eta_{x_1(i)(j)(k)}$ to other miners.

1.4 Verify$_1(c_{x_1(i)(j)}||\eta_{x_1(i)(j)(k)}, s^{(\ell)}_{x_1(i)(j)})$ each *solution* $\{s^{(\ell)}_{x_1(1)(j)}, \ldots, s^{(\ell)}_{x_1(m-1)(j)}\}$ as follows:

If $H(H(c_{x_1(i)(j)}||\eta_{x_1(i)(j)(k)})) \neq s^{(\ell)}_{x_1(i)(j)}$, Then omit from the next step, Else include.

1.5 Sort and store solutions into an ascending ordered sequence. Output solution $s^{(\mathcal{L})}_{x_1(i)(j)} = \min\{s^{(\ell)}_{x_1(1)(j)}, \ldots, s^{(\ell)}_{x_1(m))(j)}\}$.

Round 2

2.1.1 Gen$_2(x_2)$ will take $x_2 = s^{(\mathcal{L})}_{x_1(i)(j)}$ and recursively perform a double iterated hash n times as follows: Initialize: $H(H(x_2)) = \psi_{\omega_1}$, $H(H(\psi_{\omega_1})) = \psi_{\omega_2}, \ldots, H(H(\psi_{\omega_{n-1}})) = \psi_{\omega_n}$.

2.1.2 Gen$_2$ will extract the 128LSB's from each hash to overlay the cities $\{\omega_1, \ldots, \omega_n\}$ on a 2^{64}x2^{64} Euclidean plane as follows: If $\psi_{\omega_1} = 00000000000000B\ldots E360568F29EF54005F2AE8A787A28759E$, Then $\underbrace{360568F29EF54005}_{x-coordinate}\underbrace{F2AE8A787A28759E}_{y-coordinate}$, Repeat for $\{\psi_{\omega_2}, \ldots, \psi_{\omega_n}\}$.

2.1.3 Gen$_2$ will calculate ℓ_2 norms for each of the $\frac{n(n-1)}{2}$ edges on the complete graph and construct the graph distance matrix as follows:

$$c_{x_2(j)} = \begin{bmatrix} 0 & d(\omega_1,\omega_2) & d(\omega_1,\omega_3) & \ldots & d(\omega_1,\omega_n) \\ d(\omega_2,\omega_1) & 0 & d(\omega_2,\omega_3) & \ldots & d(\omega_2,\omega_n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d(\omega_n,\omega_1) & d(\omega_n,\omega_2) & d(\omega_n,\omega_3) & \ldots & 0 \end{bmatrix}.$$

2.2 Solve$_2(c_{x_2(j)})$ will be run and output solution [4] :

$$s^{(\ell)(1)}_{x_2(i)(j)} = \underbrace{\min^{[t_4]}\left\{\left(\sum_{i=1}^{n-1} d(\omega_{\pi(i)}, \omega_{\pi(j)})\right) + d(\omega_{\pi(n)}, \omega_{\pi(1)})\right\}}_{lowest\ sum},$$

$$s^{(\ell)(2)}_{x_2(i)(j)} = \underbrace{\left\langle \omega_{\pi(1)}, \omega_{\pi(2)}, \ldots, \omega_{\pi(n)} \right\rangle}_{permutation\ giving\ lowest\ sum}.$$

2.3 Propagate solutions $s^{(\ell)(1)}_{x_2(i)(j)}$ and $s^{(\ell)(2)}_{x_2(i)(j)}$ to other miners.

2.4 Sort and store solutions in ascending order. Output solution $s^{(\mathcal{L})}_{x_2(i)(j)} = \min\{s^{(\ell)(1)}_{x_2(1)(j)}, \ldots, s^{(\ell)(1)}_{x_2(m))(j)}\}$. If there is a tie for min $\{s^{(\ell)(1)}_{x_2(1)(j)}, \ldots, s^{(\ell)(1)}_{x_2(m))(j)}\}$, then the solution belonging to the miner with the lowest hash in step **1.5** will be the tie-breaking criteria.

2.5 Verify$_2(c_{x_2(j)}, s^{(\mathcal{L})}_{x_2(i)(j)})$ will run as follows: Look up associated distances from $s^{(\ell)(2)}_{x_2(i)(j)}$, namely $d(\omega_{\pi(1)}, \omega_{\pi(2)}), d(\omega_{\pi(2)}, \omega_{\pi(3)}), \ldots, d(\omega_{\pi(n-1)}, \omega_{\pi(n)}), d(\omega_{\pi(n)}, \omega_{\pi(1)})$, directly from $c_{x_2(j)}$. Sum over all the distances. If the preceding sum $= s^{(\mathcal{L})}_{x_2(i)(j)}$, then Output 1 (accept) and mark the miner identification as $(i||W)$, where the character W = winner, and progress to **2.6**. If the preceding sum $\neq s^{(\mathcal{L})}_{x_2(i)(j)}$, then Output 0 (reject) and remove $s^{(\mathcal{L})}_{x_2(i)(j)}$ from the sorted solutions and loop back to **2.4**.

2.6 Increment Block Number, Mint Currency, Commit Transactions and Loop as follows: Increase the counter of (j) to $(j+1)$. Miner $(i||W)$ will mint the new currency and the unconfirmed transactions chosen in **1.1** will be those committed in block $(j+1)$.

For the solutions provided by miner $(i||W)$, we compute: $H(H(c_{x_2(j)}||s^{(\ell)(1)}_{x_2(i)(j)}||s^{(\ell)(2)}_{x_2(i)(j)})) = \psi_{x_2(j)}$, take $s^{(\ell)}_{x_1(i)(j)} = \psi_{x_1(j)}$ from

[4] Note: min indicates the absolute minimum, whereas $\min^{[t_4]}$ indicates the minimum obtained by time t_4

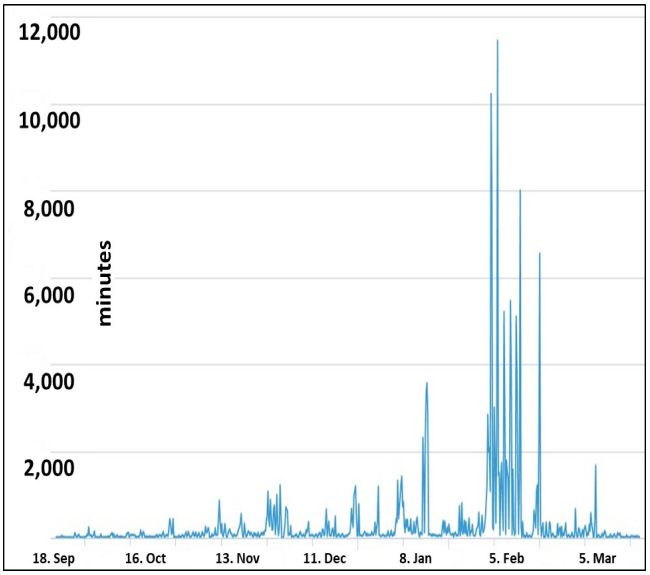

Figure 4: Average block confirmation time Sept. '17 - Mar. '18 [5]

step **1.3** and inject them into the $(j+1)$ candidate block header. We then remove the sorted values from step **1.5** and **2.4** and Loop to step **1.1**. Figure 5.

3.4 Timelines

As we propose a deterministic mining interval, the time line of this scheme will be as follows:

- step 1.1 (generating the candidate block/ challenge) occurs as a background process during steps 1.2 through 2.6,
- step 1.2 (the Hashcash stage) $[t_s, t_1 = t_s + 3]$,
- steps 1.3 through 1.5 (propagation, verification, sorting) $[t_1, t_2 = t_1 + 1]$,
- step 2.1 (generating instance of the TSP) $[t_2, t_3 = t_2 + 1]$,
- step 2.2 (solving instance of the TSP) $[t_3, t_4 = t_3 + 4]$,
- steps 2.3 through 2.6 (propagation, sorting, verification) $[t_4, t_c = t_4 + 1]$ [5].

It will also be critical to ensure protection against time shifting attacks, which can be achieved using Chronos [13].

3.5 Toy Example of Step 2.1.2

On a 2^4 x 2^4 Euclidean Plane we take the 8 LSB's from each hash output $\{\psi_{\omega_1}, \ldots, \psi_{\omega_5}\}$ as follows:

$$\psi_{\omega_1} = 36F6E32A0B7C96BF84\ldots F6\underbrace{2}_{x_1}\underbrace{A}_{y_1}. \text{ So } \omega_1 = (2, A)_{16} = (2, 10)_{10}$$

$$\psi_{\omega_2} = BEBEF3761858AC5113\ldots 8B\underbrace{7}_{x_2}\underbrace{D}_{y_2}. \text{ So } \omega_2 = (7, D)_{16} = (7, 13)_{10}$$

[5]It may also be necessary between steps 1.3 and 1.4 and between 2.3 and 2.4 to introduce another time limit for the receipt of other miners propagated solutions prior to sorting and verification.

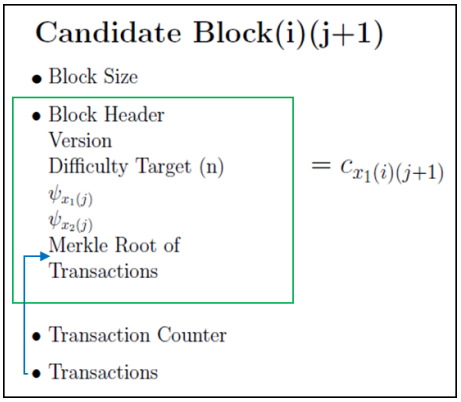

Figure 5: Block Header = $c_{x_1(i)(j)}$ of Candidate Block

$$\psi_{\omega_3} = 6E7CE26935FB65E5BA\ldots C8\underbrace{9}_{x_3}\underbrace{A}_{y_3}. \text{ So } \omega_3 = (9, A)_{16} = (9, 10)_{10}$$

$$\psi_{\omega_4} = 15F9F245212C371786\ldots 50\underbrace{5}_{x_4}\underbrace{6}_{y_4}. \text{ So } \omega_4 = (5, 6)_{16} = (5, 6)_{10}$$

$$\psi_{\omega_5} = F407951EE865345078\ldots 25\underbrace{C}_{x_5}\underbrace{2}_{y_5}. \text{ So } \omega_5 = (C, 2)_{16} = (12, 2)_{10}$$

The visual construction of this toy example can be seen in Figure 3.

3.6 Background to the Conquering Generals

The name "Conquering Generals" was the merger of ideas from the Travelling Salesman Problem and the Byzantine Generals Problem associated with the consensus achieved in a Proof of Work. We equate the work as the planning effort undertaken by a group of generals with a conquering mission. First, they perform reconnaissance to determine the cities they wish to overthrow (the Hashcash stage) then they establish the optimal order in which to conquer the cities (the TSP stage).

3.7 Implementation Considerations

Hashcash based schemes subject transaction clearing times to jitter as noted in Figure 4. Our proposal aspires to ease erratic block confirmation times by using a deterministic mining interval, requiring the clock synchronization from a legitimate time source, as noted in the prerequisites.

One of the difficulty parameters in "Conquering Generals" will be n - the number of cities. As step 2.2 is limited to 4 minutes, we determine that n be initially set to 5000. We justify our selection on the elapsed CPU years recorded by the *Concorde* code when attempting all the 110 TSPLIB instances [10].

To aid miners in determining how many propagated solutions they expect to receive in step 1.3 and 2.3, they may use a recursive algorithm such as the getaddr Python crawler to discover the other miners online [4].

In [22] we also discuss the probability and impact of the construction of geometrically congruent instances of the TSP in different blocks. In summary we argue that the combinatorial search space of all possible instances of the TSP with n cities on a 2^{64} x 2^{64} plane would make the computational resources required for storage and search infeasible. Furthermore, we recall Claims 1 - 3 justify that

each instance of the problem is NP-Hard, therefore computationally intractable to fully optimize. This final consideration is analogous to ensuring the required entropy is attained when extracting partial outputs from cryptographic hashing algorithms. We leave this consideration for further research.

4 ANALYSIS OF USEFUL WORK

Whilst there is currently a desire to create Proofs of Useful Work, the term "*Useful*" remains a qualitative description open to interpretation. Presently, this term lacks any formal and quantitative definitions when designing Proof of Work frameworks for distributed consensus. We view this an outstanding item, warranting further consideration.

We also note that our proposal presents an instance of a randomly generated computational challenge and thus lacks an explicit link to a real-life problem. Our original ambition was to design a system that would solve real-life instances of the TSP. We contemplated that a logistics company may consider outsourcing instances of the TSP to be solved by the cryptocurrency miners concerning delivery routes for their transportation fleets. However, with the latter scenario, we noted the time sensitive and confidential nature associated with commercially based real-life problems. This introduced an anticipated dependence on a Trusted Third Party to handle typical business requirements such as job queueing, prioritization and confidentiality. Reliance on a TTP would of course break the untrusted distributed consensus model inherent in cryptocurrencies. We therefore opted to focus on a framework which precluded the use of any centralized elements.

However, in [28] a framework entitled "*Resource-Efficient Mining*" addresses the issue of solving real-life problems. In this system miners are free to choose which challenges to solve (including self-submitted problems), and to decide the order in which they wish to solve them. As noted previously, real-life problems often have time restraints associated with solution submission and may also contain confidential information. Therefore, the outstanding issue of handling these properties without a TTP remains. We leave this challenge open for further research.

5 CONCLUSION

We have presented a Proof of Useful Work which avoids the potential loss of an estimated 4.3 billion USD of currently deployed ASIC hardware whilst introducing a method to subsequently construct an instance of the NP-Hard Travelling Salesman Problem. Our proposal ensures that the hardness of the scheme belongs to a formally defined computational complexity class whilst simultaneously providing a fiscally incentivized platform to perform algorithm optimization research, thereby providing useful insight into the P versus NP problem.

REFERENCES

[1] A. Antonopoulos. 2017. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies* (2 ed.). O'Reilly Media, Sebastopol, CA, USA.

[2] A. Back. 2002. Hashcash - A Denial of Service Counter-Measure. http://www.hashcash.org/hashcash.pdf

[3] M. Ball, A. Rosen, M. Sabin, and P. Nalini Vasudevan. 2017. Proofs of Useful Work. https://eprint.iacr.org/2017/203.pdf

[4] Bitnodes. 2018. Global Bitcoin Nodes Distribution. https://bitnodes.earn.com/

[5] Blockchain.info. 2018. Bitcoin Hash Rate. https://blockchain.info/charts/hash-rate

[6] Buybitcoinworldwide.com. 2018. Bitcoin Mining Hardware Comparison. https://www.buybitcoinworldwide.com/mining/hardware

[7] Coin Market Cap. 2018. CryptoCurrency Market Capitalizations. https://coinmarketcap.com

[8] C. Chantrill. 2018. State Spending for Arkansas. https://www.usgovernmentspending.com/year_spending_2018ARbs_19bs2n

[9] D. Chaum. 1983. Blind Signatures for Untraceable Payments. In *Advances in Cryptology: Proceedings of Crypto 82 (CRYPTO '82)*. Springer, Boston, MA, Santa Barbara, CA, USA, 199–204. https://doi.org/10.1007/978-1-4757-0602-4_18

[10] W. Cook. 2016. Concorde TSP Solver. http://www.math.uwaterloo.ca/tsp/concorde.html

[11] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. 2009. *Introduction to Algorithms, Third Edition* (3 ed.). MIT Press, Cambridge, MA, USA.

[12] W. Dai. 1998. b-money. http://www.weidai.com/bmoney.txt

[13] O. Deutsch, N.R. Schiff, D. Dolev, and M. Schapira. 2018. Preventing (Network) Time Travel with Chronos. In *Network and Distributed Systems Security Symposium (Proceedings of NDSS 2018)*. San Diego, CA, USA. http://dx.doi.org/10.14722/ndss.2018.23231

[14] Digiconomist. 2018. Bitcoin Energy Consumption Index. https://digiconomist.net/bitcoin-energy-consumption

[15] C. Dwork and M. Naor. 1993. Pricing via Processing or Combatting Junk Mail. In *Advances in Cryptology: Proceedings of Crypto 92 (CRYPTO '92)*. Springer, Berlin, Heidelberg, Santa Barbara, CA, USA, 139–147. https://doi.org/10.1007/3-540-48071-4_10

[16] H. Finney. 1993. Digital Cash and Privacy. http://fennetic.net/irc/finney.org/~hal/dig_cash_priv.html

[17] Gapcoin. 2014. What is Gapcoin? http://gapcoin.org/index.php

[18] M. Garey and D. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP Completeness*. W.H. Freeman and Company, New York, NY, USA.

[19] M. Jakobsson. 1999. Proofs of Work and Bread Pudding Protocols. In *Secure Information Networks. The IFIP, vol 23 (CMS '99)*. Springer, Boston, MA, Leuven, Belgium, 258–272. https://doi.org/10.1007/978-0-387-35568-9_18

[20] R. Karp. 1972. Reducibility among Combinatorial Problems. In *IBM Research Symposia Series, Complexity of Computer Computations (CCS '72)*. Springer, Boston, MA, Yorkton Heights, NY, USA, 85–103. https://doi.org/10.1007/978-1-4684-2001-2_9

[21] S. King. 2013. Primecoin: Cryptocurrency with prime number proof of work. http://primecoin.io/bin/primecoin-paper.pdf

[22] A. Loe. 2017. Conquering Generals NP-Hard Proof of Work for Blockchain Construction. https://yadi.sk/d/PG8kvFzP3SnTcs

[23] S. Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf

[24] Institute of Electrical and Inc Electronics Engineers. 2008. IEEE Standard for Floating-Point Arithmetic. https://doi.org/10.1109/IEEESTD.2008.4610935

[25] N. Szabo. 2005. Bit Gold. http://nakamotoinstitute.org/bit-gold/

[26] J. Tromp. 2015. Cuckoo Cycle: A Memory Bound Graph-Theoretic Proof-of-Work. In *Lecture Notes in Computer Science, vol 8976 (Financial Cryptography and Data Security)*. Springer, Berlin, Heidelberg, San Juan, Puerto Rico, 49–62. https://doi.org/10.1007/978-3-662-48051-9_4

[27] Litecoin Wikipage. 2015. Litecoin Scrypt Hashing. https://litecoin.info/index.php/Scrypt

[28] F. Zhang, I. Eyal, R. Escriva, A. Juels, and R. van Renesse. 2017. REM: Resource-Efficient Mining for Blockchains. In *26th USENIX Security Symposium*. Springer, Boston, MA, Santa Barbara, CA, USA, 1427–1444. https://eprint.iacr.org/2017/179.pdf