Trapdoor proof of work

PeerJ Computer Science

Trapdoor proof of work

Vittorio Capocasale

Polytechnic Institute of Turin, Turin, Italy

ABSTRACT

Consensus algorithms play a crucial role in facilitating decision-making among a group of entities. In certain scenarios, some entities may attempt to hinder the consensus process, necessitating the use of Byzantine fault-tolerant consensus algorithms. Conversely, in scenarios where entities trust each other, more efficient crash fault-tolerant consensus algorithms can be employed. This study proposes an efficient consensus algorithm for an intermediate scenario that is both frequent and underexplored, involving a combination of non-trusting entities and a trusted entity. In particular, this study introduces a novel mining algorithm, based on chameleon hash functions, for the Nakamoto consensus. The resulting algorithm enables the trusted entity to generate tens of thousands blocks per second even on devices with low energy consumption, like personal laptops. This algorithm holds promise for use in centralized systems that require temporary decentralization, such as the creation of central bank digital currencies where service availability is of utmost importance.

Subjects Algorithms and Analysis of Algorithms, Cryptography, Distributed and Parallel Computing, Cryptocurrency, Blockchain

Keywords Blockchain, Nakamoto consensus, Proof of work, Chameleon hash function

INTRODUCTION

Peer-to-peer protocols, such as the blockchain (*Nakamoto, 2008*) and the interplanetary file system (*Capocasale, Musso & Perboli, 2022*), constitute the core technologies of the Web 3.0 revolution (*Alabdulwahhab, 2018*). By leveraging blockchain technology, a group of peers can manage a tamper-resistant and decentralized ledger, ensuring transparent and secure data sharing (*Hellani et al., 2021*). These properties are particularly valuable for decision-makers, as fairness and optimality are common requirements in decision processes (*Aringhieri et al., 2022*; *Diglio et al., 2021*; *Sale et al., 2018*), making blockchain applicable in various sectors (*Elia et al., 2022*; *Khan & Anjum, 2022*; *Hasan et al., 2022*). However, according to the scalability trilemma conjecture, a blockchain cannot simultaneously achieve security, decentralization, and scalability (*Perboli, Musso & Rosano, 2018*). The intuitive reason for such a constraint lies in the fact that each peer in a blockchain network does not trust the others, keeps a full copy of the ledger, and autonomously verifies each transaction. Thus, instead of distributing workloads, blockchain systems replicate them on each peer.

Consensus algorithms play a fundamental role in blockchain systems by enabling the replication of the ledger's state across multiple peers (*Antoniadis et al., 2018*). Consensus algorithms are one of the main causes of the scalability trilemma (*Altarawneh et al., 2020*), as the replication process involves dealing with various variables, such as network delays and the presence of selfish or malicious peers.

Submitted 26 June 2023
Accepted 19 December 2023
Published 19 January 2024

Corresponding author
Vittorio Capocasale,
vittorio.capocasale@polito.it

Academic editor
Rahul Shah

Additional Information and Declarations can be found on page 16

DOI **10.7717/peerj-cs.1815**

In general, consensus algorithms can efficiently guarantee state synchronization in the presence of crashing peers (Crash Fault-Tolerant (CFT) consensus algorithms). However, when malicious peers need to be taken into account more resilient algorithms must be used (Byzantine Fault-Tolerant (BFT) consensus algorithms). However, such algorithms sacrifice efficiency for robustness (*Du et al., 2017*; *Capocasale, Gotta & Perboli, 2023*). Consequently, in environments with high-efficiency requirements, only CFT algorithms are viable (*Gatteschi et al., 2018*).

In a decentralized scenario, it is not possible to guarantee the absence of malicious peers. As a result, numerous studies in the literature have attempted to improve the efficiency of existing BFT consensus algorithms (*Pass & Shi, 2017*; *Wu, Song & Wang, 2020*; *Xu et al., 2019*; *Zhou, Hua & Jin, 2020*; *Abuidris et al., 2021*; *Liu, Tan & Zhuo, 2022*).

However, in certain scenarios, systems consist of both honest parties and potentially malicious peers. In such situations, CFT algorithms are unsuitable as they cannot tolerate malicious peers. Conversely, BFT algorithms sacrifice efficiency unnecessarily by protecting the system from honest parties. Therefore, even efficient BFT algorithms are sub-optimal. We propose a CFT consensus algorithm that allows peers to delegate decisions to honest parties and switches to BFT when honest parties crash. This dynamic approach ensures efficiency is only sacrificed when necessary without compromising the security of the system.

Given that the existing literature does not consider the possibility of an algorithm that dynamically and automatically switches between CFT and BFT, and that hybrid approaches between CFT and BFT algorithms have not been explored, we offer the following contributions:

- We introduce Trapdoor Proof of Work (TPoW), which combines Proof of Work (PoW) (*Nakamoto, 2008*) and chameleon hash functions (*Ateniese et al., 2017*). TPoW is a mining algorithm that can replace PoW in Nakamoto consensus protocols, allowing a group of entities to make decisions efficiently by relying on a trusted party: the trusted party can quickly solve the TPoW challenge and exploit a trapdoor to minimize energy waste. Moreover, when the trusted party cannot participate in the consensus, the remaining entities can still make decisions by relying on a fully decentralized but less efficient protocol.
- We conduct the first experimental validation of TPoW-based Nakamoto consensus.
- We describe potential applications of TPoW-based Nakamoto consensus.

The remaining part of this article is structured as follows: "Background" introduces the main concepts related to blockchain, consensus algorithms, and chameleon hash functions; "Trapdoor Proof of Work" describes TPoW; "Protocol Analysis" analyzes TPoW; "Experimental Validation and Discussion" presents the experimental validation of TPoW and the related discussion; "Conclusion" concludes the article.

BACKGROUND

This section provides a brief overview of the main concepts relevant to this study. Additionally, it introduces the problem under investigation and summarizes the key solutions proposed in the literature.

Chameleon hash functions

A chameleon hash function is a cryptographic hash function that incorporates a trapdoor, allowing for the discovery of arbitrary collisions (*Krawczyk & Rabin, 1998*). Without knowledge of the trapdoor, chameleon hash functions are equivalent to regular cryptographic hash functions (*Khalili, Dakhilalian & Susilo, 2020*). This study utilizes private-coin chameleon hash functions (*Ateniese et al., 2017*). A private-coin chameleon hash function consists of the following algorithms (*Ateniese et al., 2017*):

- $(hk, tk) \leftarrow \text{CHGEN}(1^\kappa)$ is a probabilistic algorithm that generates the hash key (hk) and the trapdoor key (tk).
- $(h, \xi) \leftarrow \text{CHASH}(hk, m; r)$ is a probabilistic algorithm that hashes the message m using private-coin randomness $r \in \mathcal{R}_{hash}$. CHASH produces the chameleon hash (h) and the check value (ξ).
- $d \leftarrow \text{CHVER}(hk, h, m, \xi)$ is a deterministic algorithm that outputs true if the hash h with the check value ξ is a valid chameleon hash for the message m. CHVER returns false otherwise.
- $\xi' \leftarrow \text{CHCOL}(tk, (h, m, \xi), m')$ is a probabilistic algorithm that enables the discovery of hash collisions. Specifically, given a message m' and a valid tuple (h, m, ξ), CHCOL outputs ξ' such that (h, m', ξ') remains a valid tuple. In other words, $\text{CHVER}(hk, h, m', \xi')$ returns true.

Blockchain

A blockchain is a distributed ledger managed by a group of peers, where data is organized into blocks. Each block consists of a body that contains ledger entries (transactions) and a header that contains metadata (*Nakamoto, 2008*). Notably, a block's header includes the hash of the previous block's header.

In this study, the notation for blocks is as follows:

$$B := (s_B, x_B, h_B, \xi_B),$$

where $s_B \in 0, 1^\kappa$ represents the cryptographic hash of the previous block, $x_B \in 0, 1^*$ denotes the list of transactions included in the current block (*i.e.*, the body), and h_B and ξ_B denote the chameleon hash of the block and its check value. Figure 1 visually illustrates the blockchain structure employed in this study.

It is worth noting that, within the context of this study, the term *centralized* is the opposite of *decentralized*, not *distributed*. A system is considered distributed if it consists of physically separated modules, while it is deemed decentralized if it is managed by multiple entities (*Capocasale, Gotta & Perboli, 2023*).

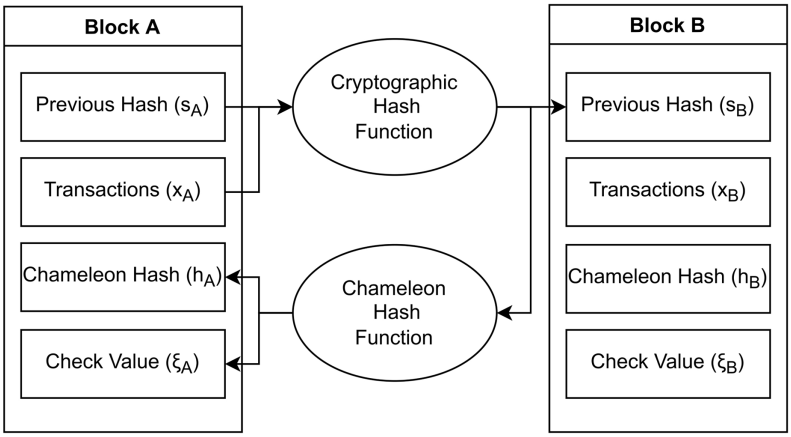


Figure 1 Structure of a TPoW blockchain. Each block stores the cryptographic hash of its predecessor, preserving immutability. However, the nodes utilize the chameleon hash of the block for mining.
Full-size DOI: 10.7717/peerj-cs.1815/fig-1

Consensus algorithms

A consensus algorithm is a protocol that enables a set of nodes to reach a mutual agreement on a decision, where all possible outcomes of the decision are equally beneficial. In a typical scenario, the nodes are geographically dispersed and can only communicate through message exchange. However, messages may experience delays, loss, or arrive out of order. In extreme cases, certain nodes may even send misleading messages to hinder the establishment of a common agreement. The objective of a consensus algorithm is to ensure that the non-faulty nodes reach the same decision. Therefore, consensus algorithms must consider a fault-tolerance model and a network model.

In terms of the fault-tolerance model, consensus algorithms that tolerate the failure of some nodes are referred to as Crash Fault-Tolerant (CFT) algorithms. Those that can also handle the presence of malicious nodes are known as Byzantine Fault-Tolerant (BFT) algorithms (*Baliga, 2017*).

Regarding the network model, consensus algorithms assume the existence of an invisible adversary who controls network delays, including message ordering.

- Synchronous model: Messages experience delays of at most a finite and known Δ.
- Asynchronous model: Messages experience delays of at most a finite but unknown Δ.
- Partially-synchronous model: The invisible adversary triggers a Global Stabilization Time (GST) event at an unknown moment. The network operates asynchronously before GST (during a transitory phase) and synchronously after GST (during the normal phase).

Consensus algorithms possess two fundamental properties (*Baliga, 2017*).

- Safety: All (honest) participants reach the same decision.
- Liveness: A decision is reached within a finite amount of time.

Proof of Work (PoW) (*Back, 2002*) is a protocol designed to prevent denial-of-service attacks. It was later adapted as a Sybil protection technique for Nakamoto consensus (*Nakamoto, 2008*). PoW involves a computationally intensive challenge that can only be solved through brute-force computation. In the context of blockchain, a block can be added to the chain if the hash of its header falls within a predefined threshold. Specifically, a valid block can be generated by iteratively modifying a portion of the header known as the nonce.

When multiple peers solve the challenge almost simultaneously, they may add different blocks to their respective chains. This situation leads to a fork, where multiple branches coexist. As one branch grows longer than the others, peers switch to it. A branch can outgrow others in the same period only if more peers support it. Thus, this protocol incorporates a majority-based voting scheme, enabling peers to reach a consensus. This consensus algorithm is known as Nakamoto consensus and combines PoW with the longest chain rule. In the context of Nakamoto consensus, safety and liveness are defined by the following properties (*Pass, Seeman & Shelat, 2017*).

- Consistency: The chains of two honest peers can differ only in the last T blocks with overwhelming probability in T.
- Future self-consistence: At any two points r and s, the chains of any honest player at r and s differ only within the last T blocks with overwhelming probability in T.
- g-chain growth: At any point in the execution, the chain of honest players grows by at least T transactions in the last $\frac{T}{g}$ rounds with overwhelming probability in T.
- μ-chain quality: at least μ of any T consecutive transactions in any chain held by some honest player were submitted by honest players with overwhelming probability in T.

Erdős–rényi random graphs

In the field of graph theory, a graph is considered connected if there exists a path between any two nodes. An Erdős–Rényi random graph, denoted as $G(n, p)$, is an undirected graph with n nodes, where each edge has a probability p of being present. An Erdős–Rényi random graph is said to be almost surely connected for some $\varepsilon > 0$ if the following condition holds (*Erdős & Rényi, 1960*):

$$p > \frac{(1+\varepsilon)\ln n}{n}. \tag{1}$$

Problem statement

Currently, consensus algorithms can be used in the following scenarios: if there are no malicious peers, both Crash Fault-Tolerant (CFT) and Byzantine Fault-Tolerant (BFT) algorithms are applicable. However, CFT algorithms tend to be more efficient. If peers are potentially malicious, only BFT algorithms can be used, but they sacrifice efficiency for robustness.

However, real-world applications may present scenarios that fall in between. These scenarios involve a mix of trustworthy parties and potentially malicious peers within the system. Therefore, the problem under consideration is finding consensus within a group of nodes that includes both peers, which are nodes that may try to disrupt the consensus process, and trusted parties, which we define as nodes that are guaranteed to follow the consensus protocol's rules and do not indulge in selfish, colluding, or disruptive behaviors. For instance, parties with economic, regulatory, or reputation-driven interests might be considered trustworthy. CFT algorithms are not suitable for solving this problem since they cannot tolerate malicious peers. On the other hand, BFT algorithms sacrifice efficiency unnecessarily because they protect the system from trusted parties that would never behave maliciously. Therefore, even efficient BFT algorithms are suboptimal. To address this, we have designed a CFT consensus algorithm that allows peers to delegate decisions to trusted parties and automatically switches to a BFT algorithm when those trusted parties are offline. This dynamic approach ensures efficiency is sacrificed only when necessary, without compromising the security of the system. We believe such an approach could find adoption in centralized systems that need to transition to a decentralized state for short periods.

Related work

The main ideas behind this study draw from Proof of Work (PoW) (*Back, 2002*) and private-coin chameleon hash functions (*Ateniese et al., 2017*). The combination of blockchain and chameleon hash functions was proposed in Ref. (*Ateniese et al., 2017*) and further explored in various studies (*Ashritha, Sindhu & Lakshmy, 2019*; *Huang et al., 2020*; *Precht & Marx Gómez, 2020*; *Wu, Ke & Du, 2021*; *Jia et al., 2022*). However, the proposed solution aimed to create a redactable blockchain rather than a consensus algorithm.

Nakamoto consensus has high energy consumption (*Mardiansyah & Sari, 2021*), which pushed several authors to propose modifications to reduce it. Proof of Elapsed Time (PoET) (*Chen et al., 2017*) utilizes hardware components to guarantee Byzantine fault tolerance. However, the security of such components has been questioned (*Schwarz, Weiser & Gruss, 2019*). Proof of Stake (PoS) algorithms (*e.g.*, Ouroboros (*Kiayias et al., 2017*)) assume that nodes are rational agents acting to maximize their economic return. Therefore, PoS algorithms are not usable without a cryptocurrency with real economic value. Some authors have proposed combining multiple mining algorithms. Proof of Contribution (*Xue et al., 2018*) is an example of such an approach, which combines PoW and PoS.

Modifications to the longest chain rule have also been explored. IOTA (*Popov, 2018*) utilizes a directed acyclic graph (DAG) instead of a linear sequence of blocks to represent the ledger. The Tangle 2.0 Leaderless Nakamoto Consensus (*Müller et al., 2022*) employs the heaviest DAG rule and a stake- or reputation-based weight function to reach an agreement.

Liu et al. (2016) introduces XPaxos, the first Cross Fault Tolerance (XFT) consensus algorithm. XFT algorithms offer stronger consistency and availability guarantees than CFT algorithms. Moreover, XFT algorithms provide stronger availability guarantees than BFT

Table 1 Contextualization of this study within the existing literature.

Algorithm	Key ideas	Advantages	Disadvantages
Proof of work (Nakamoto)	Permissionless (no identities), eventual safety	Secure, scalable	Not efficient, energy-hungry
Proof of elapsed time (Nakamoto)	Usage of trusted computing devices	Low energy consumption	Vulnerabilities in trusted computing components
Proof of stake (Nakamoto)	Game theory, rational agents	Energy savings, improved efficiency	Requires currency
Proof of contribution (Nakamoto)	PoW + PoS	Inherited from PoW and PoS	Inherited from PoW and PoS
Leaderless Nakamoto consensus	No leader election, optimisitc and concurrent block creation	Extension to DAG	Requires currency or reputation
xPaxos	All-mighty adversaries are unlikely	Consistency and availability guarantees	Cannot tolerate all-mighty adversaries
Hybrid algorithms	Two BFT algorithms combined	Compromises between the original algorithms	Compromises between the original algorithms
This study (Nakamoto)	Trusted parties are common	Enhanced efficiency, potential energy savings	Trusted party needed to have advantages

algorithms. However, XFT algorithms are applicable only when Byzantine nodes are not coordinated.

Hybrid consensus algorithms, where existing BFT protocols are combined to offer different compromises between efficiency and decentralization, have been explored in many studies (*Pass & Shi, 2017*; *Wu, Song & Wang, 2020*; *Xu et al., 2019*; *Zhou, Hua & Jin, 2020*; *Abuidris et al., 2021*; *Liu, Tan & Zhuo, 2022*). However, none of these proposals specifically focuses on a consensus algorithm that dynamically and automatically switches between CFT and BFT protocols. This study addresses that gap by describing the general idea and providing possible applications. Table 1 provides a summary of the literature.

TRAPDOOR PROOF OF WORK

This section provides an overview of Trapdoor Proof of Work (TPoW).

Protocol overview

TPoW is based on the original Proof of Work (PoW) implementation introduced by Bitcoin (*Lánskỳ, 2017*). However, in TPoW, the pre-image challenge is built upon chameleon hash functions instead of cryptographic hash functions.

Our protocol uses a cryptographic hash function to establish the linkage between each block and its predecessor, while a chameleon hash function is employed for the PoW computation (Fig. 1). The security and resilience of TPoW heavily rely on the strength of the underlying chameleon hash functions. It's important to note that certain implementations of chameleon hash functions have encountered key-exposure issues (*Ateniese & Medeiros, 2004*). However, for the purpose of this study, theoretical chameleon hash functions are assumed to exist.

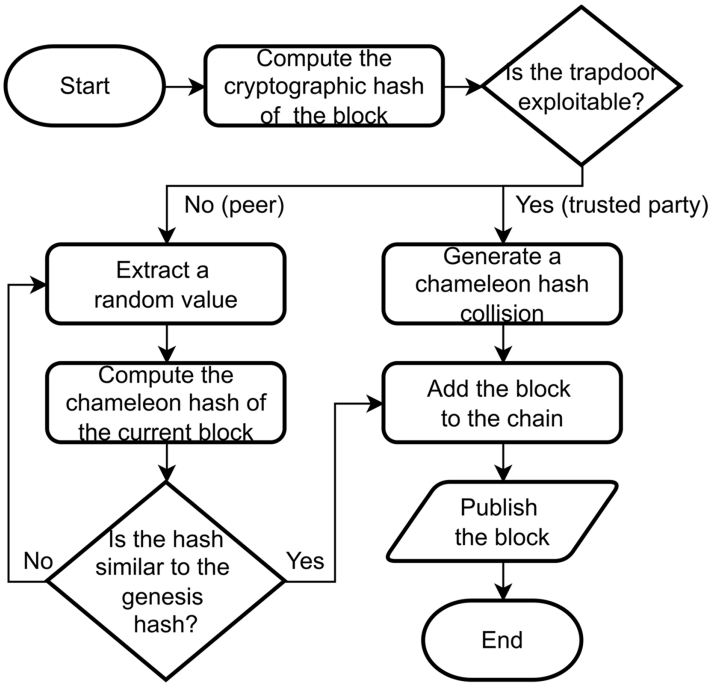


Figure 2 **A flowchart describing the generation of a block through the TPoW protocol.**
Full-size DOI: 10.7717/peerj-cs.1815/fig-2

In TPoW, a block is considered valid if its chameleon hash is sufficiently close to a predetermined value, which corresponds to the chameleon hash of the genesis block. The nodes within the network can approach the pre-image challenge through two methods (Fig. 2): brute-force computation or by utilizing the trapdoor knowledge. The brute-force method involves an inefficient trial-and-error approach, whereas the trapdoor method enables an efficient and direct computation. Consequently, any node possessing the trapdoor can effectively solve the TPoW challenge almost instantaneously. However, for nodes without the trapdoor knowledge, the challenge remains as difficult as the original PoW.

Algorithms 1–4 summarize the primary steps involved in the TPoW protocol.

Safety and liveness

We provide simple proof of the safety and liveness of TPoW by leveraging well-known PoW-related results. The following theorem is proven in a previous work (*Pass, Seeman & Shelat, 2017*).

Theorem 1. *"Assume $\rho < \frac{1}{2}$. Then for every n, Δ, there exists some sufficiently small $p_0 = \Theta\left(\frac{1}{\Delta n}\right)$ such that Nakamoto's protocol with mining parameter $p \leq p_0$ satisfies consistency, future self consistency, $1 - \frac{\rho}{1-\rho}$-chain quality, and $\frac{pn}{2}$-growth".*

In particular, let ρ represent the fraction of computational power controlled by malicious nodes, n denote the total number of nodes, Δ be the time delay parameter defining the network model, and p_0 indicate the mining hardness parameter, which represents the probability of finding a valid nonce in a single attempt.

Algorithm 1 **Genesis generation algorithm.**

function GENESIS(κ, D) ▷ GENESIS receives the security parameter (κ) and the difficulty paremeter (D)
  $(hk, tk) \leftarrow \text{CHGEN}(1^{\kappa})$ ▷ Generation of the chameleon hash keys
  $s_T \leftarrow 0$ ▷ The hash of the previous block is zero by convention
  $x_T \leftarrow (hk, D, \kappa)$ ▷ For simplicity, the TPoW difficulty (D), hk, and κ are published in the body of the genesis
  $m_T \leftarrow \text{HASH}\ (s_T, x_T)$ ▷ HASH is a cryptographic hash function
  $r_T \leftarrow \text{RNDEXTRACT}()$ ▷ Private randomness r_T is randomly extracted from its domain
  $(h_T, \xi_T) \leftarrow \text{CHASH}(hk, m_T, r_T)$ ▷ Generation of the chameleon hash and the check value of the genesis
  $T \leftarrow (s_T, x_T, h_T, \xi_T)$ ▷ Generation of the genesis block
  return T
end function

Algorithm 2 **Block validity verification algorithm.**

function VERIFY(A, B, T) ▷ VERIFY receives the current chain head (A), the block to append to it (B), and the genesis (T)
  $m_A \leftarrow \text{HASH}(A.s, A.x)$ ▷ The cryptographic hash of the current chain head
  $m_B \leftarrow \text{HASH}(B.s, B.x)$ ▷ The cryptographic hash of the new block
  $p_1 \leftarrow \text{EQUAL}(B.s, m_A)$ ▷ A valid block contains the cryptographic hash of its predecessor
  $p_2 \leftarrow \text{CHVER}(hk, B.h, m_B, B.\xi)$ ▷ A valid block must have a valid chameleon hash
  $p_3 \leftarrow \text{DIFF}(T.h, B.h)\ 2^{\kappa} < \frac{2^{\kappa}}{D}$ ▷ The chameleon hash of a valid block must be close enough to the chameleon hash of the genesis. Diff performs the binary subtraction between $T.h$ and $B.h$ and converts the result into an integer. D and κ can be retrieved from the genesis block
  return p_1 **AND** p_2 **AND** p_3 ▷ A valid block satisfies all the previous conditions
end function

From a practical perspective, the theorem establishes the consistency and liveness properties of the Nakamoto consensus if the computational power controlled by malicious nodes is less than half of the total computational power of the network.

We can prove the following corollary:

Corollary 1. *Assuming $\rho < \frac{1}{2}$ and the existence of a Nakamoto protocol with a mining parameter p satisfying Theorem 1, there exists a TPoW-based Nakamoto protocol with a mining parameter p satisfying Theorem 1.*

Proof. We begin with a Nakamoto protocol that satisfies Theorem 1. We replace the cryptographic hash function used in PoW with a chameleon hash function that has the same mining parameter. The trapdoor knowledge is not provided to any of the peers. Under these conditions, cryptographic hash functions and chameleon hash functions are equivalent by definition (*Khalili, Dakhilalian & Susilo, 2020*). Therefore, the modified protocol still satisfies Theorem 1. Next, we introduce a trusted party into the system and provide the trapdoor knowledge exclusively to this trusted party. Since the trusted party is

Algorithm 3 **Slow block mining algorithm (it does not require knowledge of *tk*).**

function MINE _SLOW(A, T) ▷ MINE _SLOW receives the current chain head (A) and the genesis (T)
  $x_B \leftarrow$ FILL_TRANSACTIONS() ▷ FILL_TRANSACTION() includes some transactions in the block
  $m_A \leftarrow \text{HASH}(A.s, A.x)$ ▷ The cryptographic hash of A
  $m_B \leftarrow \text{HASH}(m_A, x_B)$ ▷ The cryptographic hash of B
  repeat
    $r_B \leftarrow$ RNDEXTRACT() ▷ The private randomness r_B is randomly extracted from its domain
    $(h_B, \xi_B) \leftarrow \text{CHASH}(hk, m_B, r_B)$ ▷ The chameleon hash is computed with the current randomness
    $B \leftarrow (m_A, x_B, h_B, \xi_B)$ ▷ The candidate block B is generated}
  until $\text{VERIFY}(A, B, T)$ ▷ The validity of B must be checked
  return B
end Function

Algorithm 4 **Fast block mining algorithm (it requires knowledge of *tk*)**

function MINE _FAST(A, T, tk) ▷ MINE _FAST receives the current chain head (A), the genesis (T), and tk
  $x_B \leftarrow$FILL_TRANSACTIONS () ▷ FILL_TRANSACTION includes some transactions in the block
  $m_A \leftarrow \text{HASH}(A.s, A.x)$ ▷ The cryptographic hash of A
  $m_B \leftarrow \text{HASH}(m_A, x_B)$ ▷ The cryptographic hash of B
  $m_T \leftarrow \text{HASH}(T.s, T.x)$ ▷ The cryptographic hash of T
  $\xi_B \leftarrow \text{CHCOL}(tk, (T.h, m_T, T.\xi), m_B)$ ▷ Hash collision: $(T.h, m_T, T.\xi)$ and $(T.h, m_B, \xi_B)$ are both valid tuples
  $B \leftarrow (m_A, x_B, T.h, \xi_B)$ ▷ A valid block is generated
  return B
end function

honest by definition, we are only increasing the mining power of the honest peers. Thus, the assumption $\rho < \frac{1}{2}$ is not violated, and the protocol continues to satisfy Theorem 1. Moreover, the protocol we have constructed is precisely a TPoW-based Nakamoto consensus. Therefore, the thesis is proven by construction.

PROTOCOL ANALYSIS

The analysis of TPoW covers various aspects, taking into account a conservative uptime ratio of 90% for the trusted party. It is worth noting that typical production systems achieve even higher uptime ratios of 99% or more (*Cérin et al., 2014*). However, in markets with high volatility or applications with critical missions, even brief periods of service downtime can have severe consequences, as discussed in "Possible Applications".

Immutability

As depicted in Fig. 1, TPoW blocks are linked using a cryptographic hash function. Therefore, past blocks cannot be modified without altering all subsequent blocks. However, the chameleon hash and the check value can be changed as long as Algorithm 2 returns true. This flexibility is not problematic because the chameleon hash and the check value do not impact the ledger's state or the chain of hashes. Nonetheless, storing the chameleon hash and the check value in the following block can prevent their alteration.

Finality

Even though the chain is immutable, forks can still occur. When the trusted party is online, transaction consolidation is guaranteed after a single block if the difficulty parameter is appropriately configured. In fact, the trusted party can publish blocks faster than the combined peers and should have no incentive to fork the system. When the trusted party is offline, the TPoW-based Nakamoto consensus has the same level of finality as the PoW-based consensus. The blocks generated by the trusted party are identifiable since they have the same chameleon hash as the genesis block. Thus, peers are aware of the number of block confirmations necessary to finalize a given transaction.

Message complexity

If the identity of the trusted party is public, each peer can directly query the trusted party to obtain the latest block. In the absence of the trusted party, each peer should query other peers and rely on the longest chain among the responses received. If the network is connected, querying a single honest peer is sufficient to receive the longest chain since such a chain is formed by the honest majority. Hence, we model the network as an Erdős Rényi random graph $\mathrm{G}(n, p)$ where each node queries k other peers. Additionally, we assume that q out of the n nodes are faulty. Only edges connecting honest peers need to be considered since faulty nodes may never respond to queries. Thus, G is almost surely connected if:

$$\frac{k}{n}\left(\frac{n-q}{n}\right)^2 > \frac{(1+\varepsilon)\ln n}{n}. \tag{2}$$

For instance, if $n = 10^6$ and $q = \frac{n}{3}$, the honest nodes should randomly query $k = 32$ other peers to obtain the longest chain.

Therefore, the message complexity of the TPoW-based Nakamoto consensus is $O(kn)$. Specifically, $k = 1$ when peers query the trusted party directly, and $k = 32$ when peers query each other. Taking into account the assumed uptime ratio of the trusted party, the average value for k is 4. By comparison, under normal conditions, the message complexity is $O(n)$ for Raft (*Ongaro & Ousterhout, 2019*), $O(n^2)$ for PBFT (*Castro & Liskov, 1999*), $O(kn)$ with $k = 32$ for PoW-based Nakamoto consensus (as per the previous analysis), and $O(kn)$ for Avalanche (where $k = 20$ is currently used) (*Rocket et al., 2019*).

Energy efficiency

TPoW could result in energy savings since the trusted party generates most of the blocks most of the time. For example, finding a 256-bit hash collision on a personal laptop takes

about 165 ± 5 microseconds[1]. In contrast, the thousands of miners in the Bitcoin network require approximately 10 minutes to find a 46-bits hash collision, according to current statistics (*Blockchain.Com, 2022*). Therefore, attempting to solve the TPoW challenge is not advantageous for peers when the trusted party is online. Instead, the optimal strategy for peers is to wait for some time before addressing the TPoW challenge. This waiting period can be estimated based on the average chain growth rate. As a result, energy is only wasted when the trusted party is offline, potentially leading to energy savings.

[1] Laptop model: Acer Nitro AN515-44. The test was performed using the implementation provided in Ref. (*Willems, 2020*).

Horizontal scaling

Since no single peer is likely to solve the PoW challenge two or more times consecutively, PoW does not allow for parallel block production. In contrast, TPoW enables horizontal scaling as the trusted party can optimistically create blocks in parallel and chain them afterward. The only limitation lies in the possible read/write conflicts among different transactions. Transactions can declare their read/write sets to overcome these limitations, and a parallel scheduler can order them. Hyperledger Sawtooth (*Olson et al., 2018*) adopts this approach.

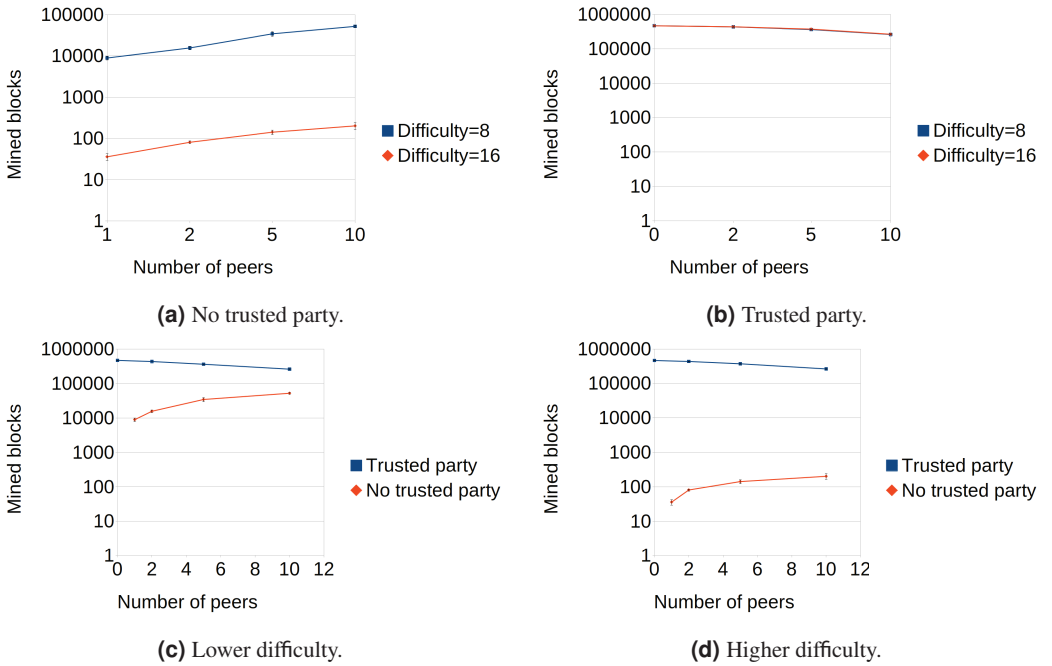
Properties

Based on the previous discussion, the TPoW protocol possesses several features:

- It does not rely on public key infrastructure or assumptions regarding clock synchronization.
- It is efficient since the trusted party can leverage the trapdoor to quickly solve the TPoW challenge without wasting energy.
- It is resilient as it does not depend on the active participation of the trusted party. The TPoW challenge remains solvable even without knowledge of the trapdoor. In such cases, TPoW behaves like a standard PoW.
- It is dynamic and automatic, with each node operating independently of others. The presence of the trusted party does not need to be notified to peers to take advantage of improved efficiency, and no action is required when the trusted party ceases to participate in the protocol. The transition between BFT and CFT is automatic, unnoticeable, and does not introduce additional risks.
- The trusted party has complete control over the blockchain. The trusted party can solve the TPoW challenge faster than all other parties combined. Consequently, the trusted party can fork the system at any time by producing a branch longer than the currently active one (similar to a 51% attack). However, the trusted party is honest by definition and should only fork the system in exceptional situations such as stolen assets. To prevent deep forks, finality gadgets could be added to the TPoW protocol (*Buterin & Griffith, 2017*).

EXPERIMENTAL VALIDATION AND DISCUSSION

This section describes and discusses the experimental tests performed to validate some of the theoretical results obtained in “Protocol Analysis”.



(a) No trusted party. **(b)** Trusted party. **(c)** Lower difficulty. **(d)** Higher difficulty.

Figure 3 **Performance evaluation of the TPoW-based Nakamoto consensus.**
Full-size DOI: 10.7717/peerj-cs.1815/fig-3

Test description

We created a TPoW simulation based on the implementation of chameleon hash functions provided in Ref. (*Willems, 2020*). We measured the number of blocks mined by a varying number of processes in two minutes. All processes pushed the blocks to a shared data structure protected by a mutex. A process could be a peer or a trusted party. The difference between the two lies in the fact that peers are not aware of the trapdoor and cannot efficiently find collisions, whereas trusted parties can. We examined the behavior of TPoW both when a trusted party was online and when it was not. We also examined the behavior of TPoW for different values of the difficulty parameter. We conducted the tests on an Acer Nitro AN515-44 laptop and used the Go programming language version 1.16.

Discussion

The results of the tests are summarized in Fig. 3 and confirm the theoretical analysis of TPoW. Figure 3A shows the behavior of TPoW when no trusted party is online. As expected, increasing the number of peers also increases the number of mined blocks if the difficulty is kept constant, in conformity with PoW. Moreover, when the difficulty parameter increases from 2^8 to 2^{16} and the number of peers is kept constant, the number of mined blocks decreases by approximately a factor of 2^8. This indicates that when no trusted party is online, TPoW behaves similarly to PoW, and the difficulty parameter can be adjusted to suit the network size, for example, to meet a target block period, as in the case of Bitcoin.

Conversely, when the trusted party is online (Fig. 3B), the difficulty parameter does not affect the mining process. The lines representing the number of mined blocks by the

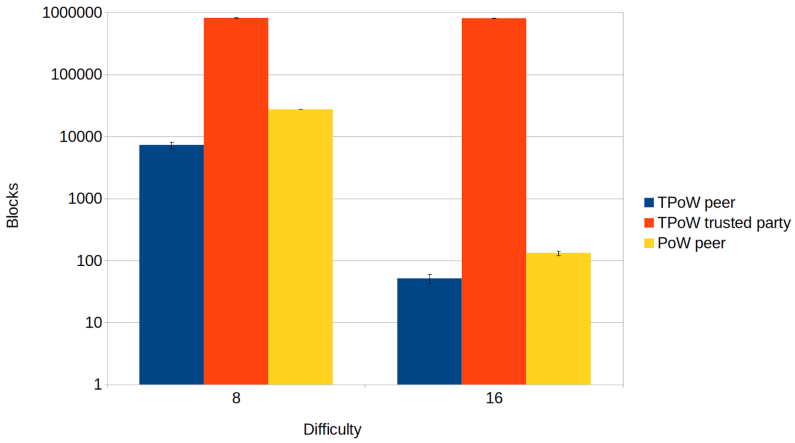


Figure 4 **Comparison between TPoW and PoW.** Full-size DOI: 10.7717/peerj-cs.1815/fig-4

trusted party with different difficulties overlap almost perfectly, suggesting that the trusted party can mine blocks at a high rate regardless of the difficulty. This result confirms that TPoW is energy-efficient when the trusted party is online, and the trusted party can generate more blocks than all the combined peers (as depicted in Figs. 3C and 3D).

An unexpected finding during the tests was that the number of mined blocks decreased as the number of peers increased. This behavior was attributed to the current testing strategy, which involves a shared data structure protected by a mutex. The acquisition of the mutex likely becomes more time-consuming as the number of contenders increases, causing a slowdown for the trusted party. Thus, improving the testing strategy could lead to obtaining the expected behavior.

Figure 4 shows a comparison of the blocks mined by a single node, either a TPoW peer, a TPoW trusted party, or a PoW peer. Depending on the difficulty, the TPoW trusted party can outperform a PoW peer by several orders of magnitude. Upon comparing the two types of peers, it is evident that the TPoW peer produces fewer blocks, possibly due to the complexity of computing chameleon hashes. However, in real-world scenarios, this does not translate to a performance loss but rather to the use of simpler challenges. In fact, the difficulty parameter is usually tuned to obtain a target block period, which must be as short as possible but also much longer than the block propagation time to guarantee safety (*Pass, Seeman & Shelat, 2017*). This restriction does not apply to the trusted party, which is assumed to be honest and should produce blocks at the highest rate possible. The results of this test further confirm that TPoW behaves like PoW when the trusted party is offline.

Possible applications

TPoW could simplify network management in certain use cases. We believe that TPoW could find adoption in systems that need to become decentralized for short periods. We propose two scenarios: enhanced availability and gradual decentralization. However, other scenarios are likely to emerge in the future.

Service availability is a key factor in mission-critical or financial applications. For example, periods of market volatility are characterized by the sudden generation of high

volumes of transactions, particularly when investors use automated trading tools (*Attanasio et al., 2019*). However, centralized exchanges cannot process such a high volume of transactions as they usually handle much lower volumes, leading to their services going offline. Consequently, investors are unable to submit transactions when they need to. One such situation occurred in May 2021, with investors losing millions of U.S. dollars in a cryptocurrency market crash (*Kowsmann & Ostroff, 2021*).

If a user relies on the services of a centralized exchange, they must trust the company managing it. Thus, TPoW could be exploited by centralized exchanges to decentralize their architecture when their systems are overwhelmed by users' requests. Such an approach would safeguard the availability of their services. Similarly, TPoW could enable the development of resilient central bank digital currencies (CBDCs) (*Benedetti et al., 2022*): users would be able to transact even when the trusted central bank is offline.

In recent years, smart contracts (*Capocasale & Perboli, 2022*) have created the opportunity to transform centralized services into decentralized ones (*Hribernik et al., 2020*; *Serrano, 2022*). However, such transformations are often experimental, as decentralized paradigms are not yet well-established. To mitigate decentralization uncertainty, projects like Olympus DAO (decentralized autonomous organization) (*Chitra et al., 2022*) are defined as futuristic or as Ponzi schemes (*Thurman, 2021*). In response to this, the Polkadot (*Burdges et al., 2020*) ecosystem launched Kusama (*Burdges et al., 2020*). Kusama is an experimental network with real economic incentives and a current market capitalization of around 300 million €. IOTA (*Popov, 2018*) was launched in 2016 as a temporary centralized protocol but has not yet switched to a decentralized one, as many features are still under development.

The aforementioned real-world examples demonstrate the importance and difficulties of migrating from well-established centralized solutions to experimental and decentralized ones. TPoW would allow for a smooth transition from a centralized to a decentralized system through a gradual approach: the system's managers could monitor the impact of the decentralized features by not participating in the consensus and could regain control of the system in case any issues arise by leveraging their trapdoor knowledge.

CONCLUSION

This article introduced Trapdoor Proof of Work (TPoW), a mining algorithm that combines Proof of Work (PoW) and chameleon hash functions. By leveraging the knowledge of the trapdoor, trustworthy peers can efficiently generate blocks. From the perspective of potentially malicious peers, TPoW is as hard as the original PoW. Thus, TPoW enables fast probabilistic finality, low message complexity, horizontal scaling, and energy savings when the trusted party is online.

TPoW is applicable in systems composed of a trustworthy party and potentially malicious peers. TPoW-based consensus dynamically and automatically adapts to the evolving state of the system: decisions are delegated to the trustworthy party while it is online, whereas Byzantine fault tolerance is guaranteed when no trustworthy party is online. This behavior can be exploited to enhance service availability or facilitate the transition from a centralized to a decentralized protocol.

The main limitation of this study lies in its lack of testing in real-world scenarios with thousands of geographically distributed peers and real workloads. Future developments will aim to conduct additional tests to analyze TPoW's performance in a more realistic environment, potentially through integration into Bitcoin's codebase. Furthermore, a quantitative comparison of the energy wasted by PoW and TPoW could extend the analysis conducted in this study. Research efforts could also explore application-specific consensus algorithms or propose protocols for achieving consensus under alternative assumptions.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

The authors received no funding for this work.

Competing Interests

The authors declare that they have no competing interests.

Author Contributions

- Vittorio Capocasale conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The code is available in the Supplemental File.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.1815#supplemental-information.

REFERENCES

Abuidris Y, Kumar R, Yang T, Onginjo J. 2021. Secure large-scale e-voting system based on blockchain contract using a hybrid consensus model combined with sharding. *ETRI Journal* **43(2)**:357–370 DOI 10.4218/etrij.2019-0362.

Alabdulwahhab FA. 2018. Web 3.0: the decentralized web blockchain networks and protocol innovation. In: *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*. 1–4.

Altarawneh A, Herschberg T, Medury S, Kandah F, Skjellum A. 2020. Buterin's scalability trilemma viewed through a state-change-based classification for common consensus algorithms. In: *CCWC*. 727–736.

Antoniadis K, Guerraoui R, Malkhi D, Seredinschi D-A. 2018. State machine replication is more expensive than consensus. In: *Leibniz International Proceedings in Informatics, LIPIcs*. Vol. 121.

Aringhieri R, Bigharaz S, Duma D, Guastalla A. 2022. Fairness in ambulance routing for post disaster management. *Central European Journal of Operations Research* **30(1)**:189–211 DOI 10.1007/s10100-021-00785-y.

Ashritha K, Sindhu M, Lakshmy K. 2019. Redactable blockchain using enhanced chameleon hash function. In: *ICACCS*. 323–328.

Ateniese G, Magri B, Venturi D, Andrade E. 2017. Redactable blockchain-or-rewriting history in bitcoin and friends. In: *EuroS&P*. 111–126.

Ateniese G, Medeiros Bd. 2004. On the key exposure problem in chameleon hashes. In: *International Conference on Security in Communication Networks*. Cham: Springer, 165–179.

Attanasio G, Cagliero L, Garza P, Baralis E. 2019. Quantitative cryptocurrency trading: exploring the use of machine learning techniques. In: *Proceedings of the 5th Workshop on Data Science for Macro-Modeling with Financial and Economic Datasets, DSMM'19*. 1–6.

Back A. 2002. Hashcash-a denial of service counter-measure. *Available at http://www.hashcash.org/papers/hashcash.pdf*.

Baliga A. 2017. Understanding blockchain consensus models. *Available at https://www.persistent.com/wp-content/uploads/2017/04/WP-Understanding-Blockchain-Consensus-Models.pdf*.

Benedetti M, De Sclavis F, Favorito M, Galano G, Giammusso S, Muci A, Nardelli M. 2022. A pow-less bitcoin with certified byzantine consensus. ArXiv preprint DOI 10.48550/arXiv.2207.06870.

Blockchain.Com. 2022. Blockchain charts. *Available at https://www.blockchain.com/charts#mining*.

Burdges J, Cevallos A, Czaban P, Habermeier R, Hosseini S, Lama F, Alper HK, Luo X, Shirazi F, Stewart A, Wood G. 2020. Overview of polkadot and its design considerations. ArXiv preprint DOI 10.48550/arXiv.2005.13456.

Buterin V, Griffith V. 2017. Casper the friendly finality gadget. ArXiv preprint DOI 10.48550/arXiv.1710.09437.

Capocasale V, Gotta D, Perboli G. 2023. Comparative analysis of permissioned blockchain frameworks for industrial applications. *Blockchain: Research and Applications* **4(1)**:100113 DOI 10.1016/j.bcra.2022.100113.

Capocasale V, Musso S, Perboli G. 2022. Interplanetary file system in logistic networks: a review. In: *Proceedings-2022 IEEE 46th Annual Computers, Software, and Applications Conference, COMPSAC 2022*. 1684–1689.

Capocasale V, Perboli G. 2022. Standardizing smart contracts. *IEEE Access* **10**:91203–91212 DOI 10.1109/ACCESS.2022.3202550.

Castro M, Liskov B. 1999. Practical byzantine fault tolerance. In: *OSDI*. Vol. 99173–186.

Cérin C, Coti C, Delort P, Diaz F, Gagnaire M, Gaumer Q, Guillaume N, Lous JL, Lubiarz S, Raffaelli J-L, Shiozaki K, Schauer H, Smets J-P, Seguin L, Ville A. 2014. Downtime statistics of current cloud solutions. *Available at https://lipn.univ-paris13.fr/~coti/papiers/iwgcr2014.pdf*.

Chen L, Xu L, Shah N, Gao Z, Lu Y, Shi W. 2017. On security analysis of proof-of-elapsed-time (POET). In: *International Symposium on Stabilization, Safety, and Security of Distributed Systems*. 282–297.

Chitra T, Kulkarni K, Angeris G, Evans A, Xu V. 2022. Defi liquidity management via optimal control: ohm as a case study. *Available at https://people.eecs.berkeley.edu/~ksk/files/Ohm_Liquidity_Management.pdf*.

Diglio A, Mancuso A, Masone A, Piccolo C, Sterle C. 2021. A milp formulation for the reorganization of the blood supply chain in italian regions. In: *Optimization and Data Science: Trends and Applications*. Cham: Springer International Publishing, 51–66.

Du M, Ma X, Zhang Z, Wang X, Chen Q. 2017. A review on consensus algorithm of blockchain. In: *SMC*. Vol. 2017-January2567–2572.

Elia N, Barchi F, Parisi E, Pompianu L, Carta S, Bartolini A, Acquaviva A. 2022. Smart contracts for certified and sustainable safety-critical continuous monitoring applications. In: Chiusano S, Cerquitelli T, Wrembel R, eds. *Advances in Databases and Information Systems*. Cham: Springer International Publishing, 377–391.

Erdős P, Rényi A. 1960. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences* **5(1)**:17–60 DOI 10.1515/9781400841356.38.

Gatteschi V, Lamberti F, Demartini C, Pranteda C, Santamaría V. 2018. Blockchain and smart contracts for insurance: is the technology mature enough? *Future Internet* **10(2)**:20 DOI 10.3390/fi10020020.

Hasan ASMT, Sabah S, Haque RU, Daria A, Rasool A, Jiang Q. 2022. Towards convergence of IoT and blockchain for secure supply chain transaction. *Symmetry* **14(1)**:64 DOI 10.3390/sym14010064.

Hellani H, Sliman L, Samhat AE, Exposito E. 2021. On blockchain integration with supply chain: Overview on data transparency. *Logistics* **5(3)**:46 DOI 10.3390/logistics5030046.

Hribernik M, Zero K, Kummer S, Herold DM. 2020. City logistics: towards a blockchain decision framework for collaborative parcel deliveries in micro-hubs. *Transportation Research Interdisciplinary Perspectives* **8(1)**:100274 DOI 10.1016/j.trip.2020.100274.

Huang K, Zhang X, Mu Y, Rezaeibagha F, Du X, Guizani N. 2020. Achieving intelligent trust-layer for internet-of-things via self-redactable blockchain. *IEEE Transactions on Industrial Informatics* **16(4)**:2677–2686 DOI 10.1109/TII.2019.2943331.

Jia M, Chen J, He K, Du R, Zheng L, Lai M, Wang D, Liu F. 2022. Redactable blockchain from decentralized chameleon hash functions. *IEEE Transactions on Information Forensics and Security* **17**:2771–2783 DOI 10.1109/TIFS.2022.3192716.

Khalili M, Dakhilalian M, Susilo W. 2020. Efficient chameleon hash functions in the enhanced collision resistant model. *Information Sciences* **510(4)**:155–164 DOI 10.1016/j.ins.2019.09.001.

Khan A, Anjum A. 2022. Blockchain-based distributed platform for accountable medical data sharing. In: *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*. Piscataway: IEEE.

Kiayias A, Russell A, David B, Oliynykov R. 2017. Ouroboros: a provably secure proof-of-stake blockchain protocol. In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10401. Springer Lecture Notes in Computer Science (LNCS), 357–388.

Kowsmann P, Ostroff C. 2021. Binance froze when bitcoin crashed. Now users want their money back. *Available at https://www.wsj.com/articles/binance-froze-when-bitcoin-crashed-now-users-want-their-money-back-11626001202*.

Krawczyk H, Rabin T. 1998. Chameleon hashing and signatures. *IACR Opens the Cryptology ePrint Archive* **1998**:10.

Lánskỳ J. 2017. Bitcoin system. *Acta Informatica Pragensia* **6(1)**:20–31 DOI 10.18267/j.aip.97.

Liu Y, Tan T, Zhuo Y. 2022. Hybrid consensus protocols and security analysis for blockchain. In: *2022 International Conference on Data Analytics, Computing and Artificial Intelligence (ICDACAI)*. Piscataway: IEEE, 191–195.

Liu S, Viotti P, Cachin C, Quéma V, Vukolić M. 2016. XFT: practical fault tolerance beyond crashes. In: *OSDI*. 485–500.

Mardiansyah V, Sari RF. 2021. Implementation of proof-of-work concept algorithm using simblock simulator. In: *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. Piscataway: IEEE, 1–6.

Müller S, Penzkofer A, Polyanskii N, Theis J, Sanders W, Moog H. 2022. Tangle 2.0 leaderless nakamoto consensus on the heaviest dag. *IEEE Access* **10**:105807–105842 DOI 10.1109/ACCESS.2022.3211422.

Nakamoto S. 2008. Bitcoin: a peer-to-peer electronic cash system. *Available at https://bitcoin.org/bitcoin.pdf*.

Olson K, Bowman M, Mitchell J, Amundson S, Middleton D, Montgomery C. 2018. Sawtooth: an introduction. *Available at https://8112310.fs1.hubspotusercontent-na1.net/hubfs/8112310/Hyperledger/Hyperledger_Sawtooth_WhitePaper.pdf*.

Ongaro D, Ousterhout J. 2019. In search of an understandable consensus algorithm. In: *USENIX ATC*. 305–319.

Pass R, Seeman L, Shelat A. 2017. Analysis of the blockchain protocol in asynchronous networks. In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10211. Springer Lecture Notes in Computer Science (LNCS), 643–673.

Pass R, Shi E. 2017. Hybrid consensus: efficient consensus in the permissionless model. In: *LIPIcs*. Vol. 91.

Perboli G, Musso S, Rosano M. 2018. Blockchain in logistics and supply chain: a lean approach for designing real-world use cases. *IEEE Access* **6**:62018–62028 DOI 10.1109/ACCESS.2018.2875782.

Popov S. 2018. The tangle. *Available at https://www.cryptoground.com/storage/files/1527489133_iota1_4_3.pdf*.

Precht H, Marx Gómez J. 2020. Redactable blockchain–leveraging chameleon hash functions for a GDPR compliant blockchain. In: *Konferenzband zum Scientific Track der Blockchain Autumn School 2020*. Vol. 166–70.

Rocket T, Yin M, Sekniqi K, van Renesse R, Sirer EG. 2019. Scalable and probabilistic leaderless bft consensus through metastability. ArXiv preprint DOI 10.48550/arXiv.1906.08936.

Sale H, Berg K, Landsverk H, Wruk J, Cibis K, Macdonald R. 2018. Prototype for estimation and forecasting of the future demand and generation from households in selected european countries. In: *Proceedings-2018 53rd International Universities Power Engineering Conference, UPEC 2018*. 1–6.

Schwarz M, Weiser S, Gruss D. 2019. Practical enclave malware with intel SGX. In: *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. 177–196.

Serrano W. 2022. Verification and validation for data marketplaces via a blockchain and smart contracts. *Blockchain: Research and Applications* **3**:100100 DOI 10.1016/j.bcra.2022.100100.

Thurman A. 2021. Olympus dao might be the future of money (or it might be a ponzi). *Available at https://www.coindesk.com/policy/2021/12/05/olympus-dao-might-be-the-future-of-money-or-it-might-be-a-ponzi/*.

Willems J. 2020. Chameleon_hash. *Available at https://github.com/julwil/chameleon_hash*.

Wu C, Ke L, Du Y. 2021. Quantum resistant key-exposure free chameleon hash and applications in redactable blockchain. *Information Sciences* **548(1)**:438–449 DOI 10.1016/j.ins.2020.10.008.

Wu Y, Song P, Wang F. 2020. Hybrid consensus algorithm optimization: a mathematical method based on pos and pbft and its application in blockchain. *Mathematical Problems in Engineering* **2020(11)**:1–13 DOI 10.1155/2020/7270624.

Xu R, Chen Y, Blasch E, Chen G. 2019. Microchain: a hybrid consensus mechanism for lightweight distributed ledger for IoT. ArXiv preprint DOI 10.48550/arXiv.1909.10948.

Xue T, Yuan Y, Ahmed Z, Moniz K, Cao G, Wang C. 2018. Proof of contribution: a modification of proof of work to increase mining efficiency. In: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. Piscataway: IEEE, 636–644.

Zhou C-X, Hua Q-S, Jin H. 2020. Hotdag: Hybrid consensus via sharding in the permissionless model. In: *Lecture Notes in Computer Science*. Vol. 12384. Springer Lecture Notes in Computer Science (LNCS), 807–821.