IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 5, 2019, accepted January 3, 2020, date of publication March 3, 2020, date of current version March 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2977884

A Framework for Privacy Preserving, Distributed Search Engine Using Topology of DLT and Onion Routing

ALI RAZA[1], KYUNGHYUN HAN[1], AND SEONG OUN HWANG[2], (Senior Member, IEEE)
[1]Department of Electronics and Computer Engineering, Hongik University, Sejong 30016, South Korea
[2]Department of Computer Engineering, Gachon University, Seongnam 13120, South Korea

Corresponding author: Seong Oun Hwang (sohwang@gachon.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) under Grant 2020R1A2B5B01002145, and in part by the Gachon University Research Fund of 2019 under Grant GCU-2019-0838.

ABSTRACT Risks regarding Internet privacy and security have been identified as issues for both new and experienced users of Internet technology. This paper explores the privacy and security threats in centralized search engines and proposes a decentralized search system. The proposed decentralized search system provides privacy and security for its users. The decentralized nature also provides transparency, distributed search and community-driven decisions in the proposed framework. We compare the proposed framework with the existing centralized approach. Our analysis shows that (1) there is no significant difference in search time between the centralized and proposed framework, and (2) the proposed framework is more efficient by providing search results with fewer search resources compared to the centralized approach.

INDEX TERMS Privacy, security, blockchain, onion routing, web-searching.

I. INTRODUCTION

Google is responsible for 77 percent of searches worldwide. The way Google displays information, and the way Google's search algorithm works, have a powerful impact on the way we view the world. Google uses thousands of factors in its search algorithm. Even the world's leading reverse engineers have not determined all of these factors, or figured out how Google's constantly changing search algorithm works. This lack of transparency is frustrating for content creators. This creates a scenario where Google is a god where only those who are willing to play by its rules are allowed in. Our aim is to change that by making its ranking and indexing mechanisms publicly available to anyone. Australian regulators had set their sights on big tech companies for a long time. After the 18-month investigation into the impacts of Google and Facebook on the country's economy, the Australian Government released a new report, which suggested that Google and Facebook need to reveal their secretive algorithms to operate in Australia. A distributed ledger technology-based search engine can theoretically win here. Despite spending years to become more transparent and having almost unlimited resources to invest, search engines have become increasingly secretive. They offer a murky dispute resolution process. Specifically, it is almost impossible to contact anyone directly and obtain answers, and their ranking factors are virtually unknown and ever-changing. Google can also track our privacy [1]. What we search for is most likely related to our daily lives.

The associate editor coordinating the review of this manuscript and approving it for publication was Yan Huo.

A. PROBLEM STATEMENT

Search engines including Google are widely used in our daily life. Its search results may greatly affect us in the sense that the content we see may have much influence in our life. Since our search results are heavily dependent on search engines run by central authorities, they can be monopolized and manipulated by them. For example, a centralized search engine can do the following:

1) They can be biased for a particular website or community [2] and may prompt forged results for our queries.
2) Google can also track our privacy - what we search for is most likely related to our daily lives.
3) On a political-point-of-view, if a specific search engine has a great impact, biased results may lead to harmful democratic politics [3].

4) Single point of failure - although there are many search engines, 77% of our search results depend on Google. However, what happens when Google crashes?

B. CONTRIBUTIONS

The main contributions of this paper are as follows:

1) We propose the first framework to decentralize a search engine using distributed ledger technology and provide anonymity to its users using onion routing.
2) We propose a novel consensus algorithm called Proof of Randomness, which is more secure and efficient than existing ones.
3) The proposed framework additionally boasts desirable features: transparency by removing the monopoly and biasness of centralized search engines; resistance against single point of failure.
4) We provide an in-depth analysis of the proposed framework in terms of performance as well as security. We consider various attack scenarios and theoretically prove that it is secure against collusion and Sybil attacks. By conducting simulations, we experimentally show that it is more efficient and practical than Google.

The rest of the paper is organized as follows: Section II presents the related work. Section III gives the preliminaries of the proposed framework, and is followed by a detailed description of our proposed model in Section IV. Section V presents the smart contract and the consensus algorithm. Sections VI and VII present the security analysis and the performance evaluation, respectively. Section VIII concludes the paper.

II. RELATED WORK

In this section, we describe some of the existing distributed search engines. Faroo implements a modified Kademlia [4] distributed hash table (DHT). The ID of a node is generated from the node's MAC address and some other local information. After a Web page is crawled, the client program generates relevant reverse word index (RWI) entries and computes the target ID for each entry, which is the hash value of the search term. An entry is then stored on 20 index nodes, whose IDs are closest to the target ID. To perform a search, the initiator of a search query computes the target ID of the search term and queries the relevant index nodes. The IP address of the initiator is sent along with the query, so that the responsible index nodes can reply in a direct connection. Only the top 100 results are returned as replies. Seeks fundamentally utilizes the network differently from the approach of YaCy [5] and Faroo. A search query is not mapped to the relevant RWI entries, but mapped to users who issued similar queries. Users can also ''push'' entries into the network, so that they will show up in other users' search results. Seeks P2P network is built on Chord [6]. It implements a locally sensitive hash function that maps similar queries to the same ID and thus the same group of index nodes. Users are able to register themselves at the index nodes for their search query and to retrieve the IP addresses of other peers that issued similar queries. This enables the user to communicate and share interesting search results. The goal of Seeks is the direct opposite of providing anonymity: Attackers can get a list of peers who are interested in a search term simply by issuing the query themselves. In terms of censorship resistance, while it is difficult to censor search results from centralized search engines. Seeks does not serve as an alternative for users seeking to mitigate the risk of being censored by centralized search engines. Meanwhile, an attacker can also easily deploy the 'eclipse' attack [7], to occupy index nodes for a search query; In an eclipse attack, a set of malicious, colluding overlay nodes arranges for a correct node to peer only with members of the coalition. If successful, the attacker can mediate most or all communication to and from the victim. Furthermore, by supplying biased neighbor information during normal overlay maintenance, a modest number of malicious nodes can eclipse a large number of correct victim nodes, therefore preventing users from communicating with each other. An index poisoning attack of injecting fake URLs into the network is also possible. YaCy is another distributed search engine. The YaCy network of YaCy peers (such as Freeworld) maintains a single, shared reverse-word-index for all of the crawled pages (i.e., a database of matching URLs, ordered on the words that would make up likely search terms). The index is sharded among the peers in a distributed hash table (DHT). Whenever a peer indexes a new set of pages, it writes updates to the DHT. Shards are replicated between multiple peers to bolster lookup speed and availability. YaCy does not provide any consensus between the peers for indexing so the indexing results can be manipulated by the peers. There is an in-depth description of YaCy available in [5]. In [8] researchers analyze all of these distributed search engines, with respect to their censorship resistance, resistance against malicious peers, and privacy protection. They show that none of them provides an adequate level of protection against an adversary with modest resources. Presearch [9] is an open, decentralized search engine that rewards community members with Presearch Tokens for their usage, contribution to, and promotion of the platform. DuckDuckGo (DDG) is an internet search engine that emphasizes protecting users' privacy and avoiding the filter bubble of personalized search results [10]. It returns the best results, rather than the most results, generating those results from over 400 individual sources, including crowdsourced sites such as Wikipedia, and other search engines like Google, Bing and Yahoo. But both [9] and [10] do not provide proper privacy to its users, and depend on the search engines like Google and Yahoo at the back-end. For this reason, their results can be manipulated by the centralized organizations.

In relation to privacy issues, previous research provides a different understanding of concerns about privacy [11]. There are a variety of privacy protection techniques proposed such as sender identity protection, sender k-anonymity, sender untraceability, blender anonymity, controllable anonymity in [12] and [13], used in different application environments.

In [14] the researchers used blockchain based architecture to enhance the privacy in online ecosystems. To address the challenges of online security and privacy, [15] presented a survey highlighting that research is required to address all the challenges regarding security and privacy. Reference [16] provides the studies of the European Union Blockchain Observatory & Forum on DLT and General Data Protection Regulation (GDPR) compatibility. Nevertheless, the privacy issue still remains as an open challenge in search engines.

III. PRELIMINARIES

In this section, we describe the system model, attack model, design goals, assumptions, asymmetric cryptography, onion routing and distributed ledger technology (DLT) concepts that we used in this research article.

A. SYSTEM MODEL

The system model of our proposed framework consists of four types of participants: User, Tor nodes, search nodes and Tor block nodes. The overall system model is conceptually divided into three layers: layer 1 consists of users; layer 2 consists of Tor nodes; layer 3 consists of search nodes and Tor block nodes. The role of each participant in the model is defined below:

1) *Users:* A user can be anyone who wants to search content in the distributed search engine. Users are provided with specially designed Web browsers, through which they can search for content in the framework. The users are connected to the layer two (onion network) through the specially designed Web browsers.
2) *Tor Nodes:* The main purpose of Tor nodes is the onion routing. The query of a user is routed by the Tor nodes to the mempool in the third layer. The Tor nodes carry back the Search Engine Results Pages (SERP) from the third layer to the user in the first layer. SERP are the pages displayed by search engines in response to a query by a user. The main component of the SERP is the listing of results that are returned by the search engine in response to a keyword query.
3) *Search Nodes:* Search nodes perform indexing like the data centers in centralized systems. They search for the results related to the query in a transaction in the mempool. Search nodes can use searching and indexing algorithms decided by the community, and those algorithms are publicly available. They give their search results for the consensus.
4) *Tor Block Nodes:* Tor block nodes are involved in Proof of Randomness (PoR), which will be explained later and has two parts. The first part is minimum randomness and the second part is Practical Byzantine Fault Tolerance (PBFT) [17] consensus. A Tor block node in the first part of PoR selects a group of n random Tor block nodes. This group of Tor block nodes then performs the PBFT consensus to achieve the SERP ranking results.

B. ATTACK MODEL

The attack model consists of different types of adversaries. First are the malicious Tor block nodes who try to select the biased group of nodes for the PBFT consensus in the first part of PoR. Second are the malicious nodes who try to give forged results for the PBFT consensus. Third are the malicious Tor nodes who try to eavesdrop into the search query and master indexing results. We consider that majority of the Tor nodes, search nodes and Tor block nodes are honest, but some of them are malicious. We also consider collusion and Sybil attacks, along with a new attack vector, called 'Pre-compute' attack.

C. DESIGN GOALS

The monopoly of centralized organizations can be removed by decentralizing master indexing and SERP ranking. Since the SERP is achieved by Tor block nodes by the PBFT consensus and the master indexing is done by many decentralized search nodes, the results in SERP become non-monopolized. Because everyone can see which algorithms are used for master indexing and SERP ranking, biasness can be removed. On the other hand, centralized systems do not reveal their secretive search algorithms to the public. In centralized search systems, only the centralized organization can decide the search algorithms related to the search engine and search accuracy. In the proposed framework, a community can decide the search algorithms related to the search engine and search accuracy. Note that the proposed framework does not specify any search algorithm which will be determined by the operator of the framework. For readers interested in the accuracy of search engines, we recommend reviewing [18] which investigates the accuracy of search engine hit counts for search queries of the major search engines like Google, Yahoo and Bing. In centralized systems, only the centralized organizations can earn millions of dollars based on searching, while in the proposed framework, everyone can participate in the network and get benefits. The key points of the proposed framework are as follows:

1) *Privacy Preserving*
   One of the most important objectives of the proposed framework is to protect the privacy of the users while allowing them to search for Web content. The participants in the framework should be able to search for Web content anonymously, and without compromising privacy.
2) *Distributed Search*
   The user should be able to search for Web content without trusting any centralized authority.
3) *Transparency*
   The user can see how and which algorithms are used for master indexing and SERP ranking. The user can see if their search results are biased or not.
4) *Community-driven Decision-making*
   A community should be able to decide which algorithms should be used for unbiased master indexing and SERP ranking.

5) *Community Benefits*
Along with a fair and unbiased Web search, the community should be able to earn benefits from its contribution and participation in the network.
6) *Resistance to a Single Point of Failure*
The network should be working, even if some of the participants are not working.

D. ASSUMPTIONS

We made the following assumptions in our framework:

1) All the nodes have synchronized clocks.
2) All nodes keep their cryptographic credentials safe.
3) The network has a majority of non-malicious nodes (search nodes, Tor nodes and Tor block nodes).

E. ASYMMETRIC CRYPTOGRAPHY, ONION ROUTING AND BLOCKCHAIN CONCEPTS

First, we will discuss asymmetric cryptography which is used in onion routing, blockchain and distributed ledger technology.

1) Asymmetric Cryptography
Asymmetric cryptography has a pair of keys for encryption and decryption. A public key is used for encryption and a private key is used for decryption. The messages encoded using public keys can only be decoded by their private keys. The secret transmission of a key for decryption is not required and every entity can generate a key pair and can publish their public keys to the people they want to interact with, as shown in Figure 1. More information on asymmetric cryptography can be seen in [19]. We use Public Key Encryption with the Equality Test (PKEET) [20] for registration of a node in the network.
2) Blockchain and Distributed Ledger Technology
A blockchain or distributed ledger technology (DLT) [21] is a decentralized digital ledger that transparently and permanently records blocks of transactions across computers, based on a consensus algorithm without modifying the subsequent blocks. A blockchain is similar to a linked list except that blocks are added according to a consensus protocol. DLT can be considered a first step towards a blockchain, but more importantly it does not necessarily construct a chain of blocks. Rather, the ledger in question will be stored across many servers, which then communicate to ensure that the most accurate and up-to-date record of transactions is maintained. The genesis or the first block defines the settings of blockchain and DLT as shown in Figure 2.

F. ONION ROUTING

Onion routing is an infrastructure for private communication over a public network. It provides anonymous connections

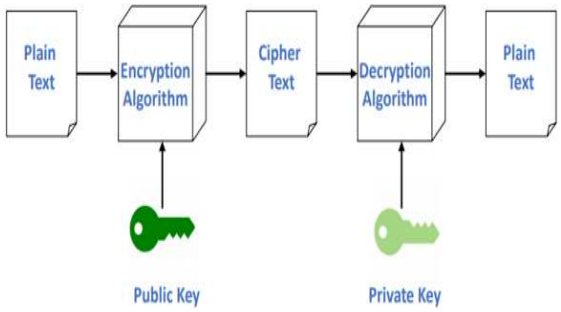

FIGURE 1. Asymmetric cryptography.

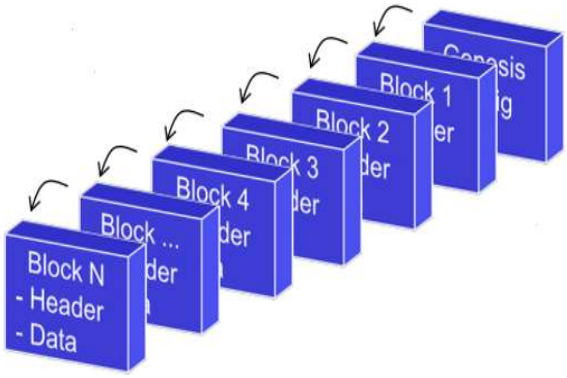

FIGURE 2. Blockchain.

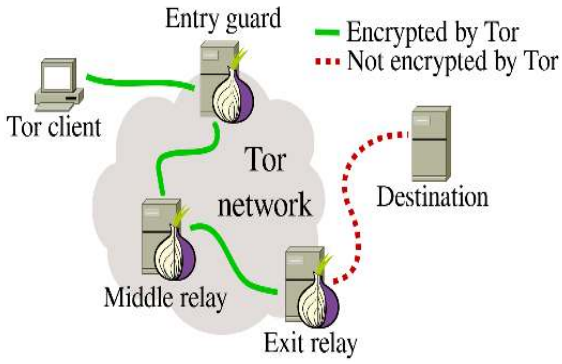

FIGURE 3. Onion network.

that are strongly resistant to both eavesdropping and traffic analysis. Onion routing's anonymous connections are bidirectional, near real-time, and can be used anywhere a socket connection can be used. Any identifying information must be in the data stream carried over an anonymous connection. An onion is a data structure that is treated as the destination address by onion routers. Thus, it is used to establish an anonymous connection. Onions themselves appear different to each onion router as well as to network observers. The same goes for data carried over the connections they establish. Proxy-aware applications, such as Web browsers and e-mail clients, require no modification to use onion routing, and do so through a series of proxies [22], [23]. Figure 3 shows the onion network.

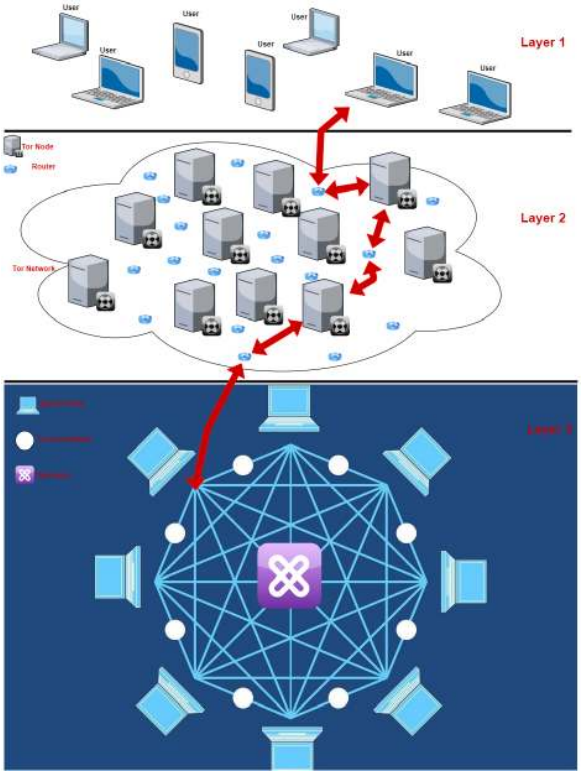

FIGURE 4. General model of the proposed framework.

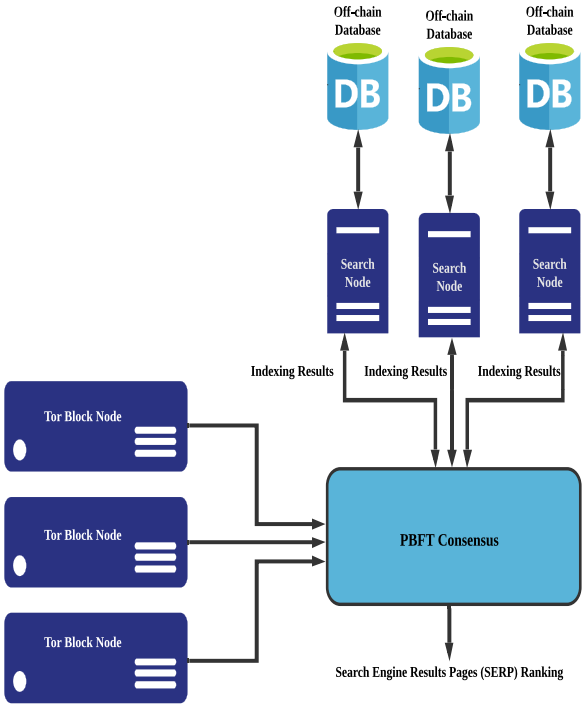

FIGURE 5. Flow chart for consensus.

IV. PROPOSED MODEL

In order to provide privacy, the proposed framework uses the topology of the onion network and the decentralization is achieved using DLT. The general model consists of the users in layer 1, Tor nodes in layer 2, and search nodes and Tor block nodes in layer 3. Figure 4 shows the general model.

Tor block nodes perform the first part of PoR by selecting groups of Tor block nodes with minimum randomness; Those groups later perform the PBFT consensus (the second part of PoR). Minimum randomness is explained in a later section. These groups are made continuously and stacked. To send a keyword query, a user connects to the onion network by creating a path called a 'chain' or circuit, through which the transactions will be transmitted to the mempool in layer 3. Note that the transactions in the mempool are visible to all the Tor block nodes and search nodes in layer 3. A user sends a keyword query in a transaction through Tor nodes to the mempool. Tor nodes use the onion network to route a transaction to the mempool in layer 3. Search nodes related to a keyword query in the transaction send their search results to the first group in the stack. The groups achieve the PBFT consensus about the SERP ranking on the search results provided by the search nodes. The SERP ranking result by the PBFT consensus is then sent by the leader of the PoR to the user via the chain or circuit of Tor nodes in the onion network. In other words, users in layer 1 use the onion network in layer 2 to establish an anonymous connection with the DLT in layer 3. The reason for using the onion network topology along with the DLT is to enhance the privacy of users, because the privacy of users can be compromised in simple DLTs and there exist some flaws in onion routing. As DLTs also provide anonymity and privacy to some extent because a user is linked to a public key address in a DLT, no one will get to know the actual identity and address of the user. Coupling the two technologies (i.e., onion network and DLT) provides an enhanced and satisfactory anonymity and privacy to the users.

A. SEARCH MODEL

Search nodes use algorithms decided by the community for master indexing in their off-chain database; Likewise Tor block nodes who are involved in the consensus use algorithms decided by the community for SERP ranking on the results returned by the search nodes in response to a keyword query. Hence, more accurate, unbiased and efficient search results are provided, as shown in Figure 5.

The order of the search results is insignificant. The proposed model divides the Tor block nodes into groups of n random members to achieve multitasking. A single group participates in the consensus for a particular keyword query and other groups of n Tor block nodes participate in the consensus for other keyword queries. Hence, parallel consensus can be achieved. This increases the efficiency of the proposed model. It should be noted that, the groups of Tor block nodes which participate in the consensus are created randomly each time by using the first part of the PoR. We will describe the PoR in detail in Section V. The number of search nodes participating in the search results are not fixed. The reason for not fixing this number is to allow all the search nodes to participate if

they have some related search results, provided the results for a keyword query occur within a given time frame.

V. SMART CONTRACT AND CONSENSUS

Smart contract consists of a set of algorithms for master indexing and SERP ranking. These algorithms can be decided by the community. Some of the consensus algorithms are as follows: Proof of Work (PoW) [24] and Proof of Stake (PoS) [25] are examples of the most famous consensus algorithms. PoW is a piece of data (hash value) which is difficult (costly, time-consuming) to produce but easy to verify in order to ensure that certain requirements are met. Producing a PoW can be a random process with low probability. Therefore, many trials and errors are required on average before a valid PoW is generated. For more information on PoW, we recommend reading [24]. Proof of Stake (PoS) is a consensus algorithm by which a blockchain network aims to achieve a distributed consensus. In a PoS-based consensus, the creator of the next block is chosen via various combinations of random selection and wealth or age (the stake). For more information on PoS, we recommend reading [25]. The consensus mechanisms including PoW and PoS can be very time-consuming and inefficient. Churns due to these consensus examples also introduce performance delays in the network [26]. In order to achieve efficiency, we propose a new consensus algorithm, Proof of Randomness (PoR).

A. PROOF OF RANDOMNESS

PoR has two parts. The first part makes groups of random Tor block nodes called 'minimum randomness'. The second part achieves the PBFT consensus. To understand the minimum randomness, consider the following example:

$$M = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots\cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots\cdots & x_{2,n} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ x_{m,1} & x_{m,2} & \cdots\cdots & x_{m,n} \end{bmatrix}$$

$$N_i = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots\cdots & y_{1,n} \\ y_{2,1} & y_{2,2} & \cdots\cdots & y_{2,n} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ y_{n^o,1} & y_{n^o,2} & \cdots\cdots & y_{n^o,n} \end{bmatrix}$$

Let's suppose that a matrix M represents the set of groups selected previously by all Tor block nodes and genesis. Let N_i be the set of groups selected by a particular Tor block node x_i. Each row in M represents a group. M is the same for all Tor block nodes, while N_i is different for each Tor block node x_i. Each Tor block node maintains the same matrix M and different N_i for each Tor block node x_i. To add a row in M and N_i, x_i needs to prove minimum randomness. Consider that x_i selects a group of n random nodes, $y_{(n^o+1),1}, y_{(n^o+1),2} \cdots\cdots, y_{(n^o+1),n}$. To prove the minimum randomness for this group, the total-average $Total_{Ave}$, average (Ave_M) of M and the average (Ave_{N_i}) of N_i are calculated respectively using the following equations.

$$Ave_M = \frac{(\sum_{i=1}^{m} Match(Row_i))}{m} \tag{1}$$

$$Ave_{N_i} = \frac{(\sum_{i=1}^{n^o} Match(Row_i))}{n^o} \tag{2}$$

$$Total_{Ave} = \frac{Ave_{N_i} + Ave_M}{2} + \alpha \tag{3}$$

Here, α is the matching factor. It is 33 if any matched row ratio ($Match(Row_i)$) is greater than 33%; Otherwise 0. To calculate $Match(Row_i)$, the members of the newly selected group are matched (the order of elements in the group is not important) with each row of the matrix M and N_i. Here m and n^o are the number of rows in M and N_i respectively. If any of Ave_{N_i}, Ave_m and $Total_{Ave}$ has a value greater than 33%, then x_i fails to prove the minimum randomness; Otherwise the x_i with minimum $Total_{Ave}$ successfully proves the minimum randomness. If more than one Tor block node has equal minimum randomness, then they are in tie. The Tor block node which is not in the tie with minimum Ave_{Match} proves the minimum randomness. Tor block nodes have to prove their minimum randomness within a time period of 0.1 second. This time is selected to improve overall efficiency and security. This process is repeated continuously and the groups created are staked. Search nodes send their results to the first group in the stack in response to a keyword query and the group is removed from the stack. Each member in the group performs a ranking of the search results for SERP and achieves the PBFT consensus on the ranking. PBFT works on the assumption that less than one-third of the peers are faulty, which means that the group should consist of at least $n = 3f + 1$ peers to tolerate f faulty peers [17]. Thus for $k = \frac{(n-1)}{3}$ malicious peers, the network requires $2f + 1$ peers to agree on the consensus. For more details on PBFT, we recommend reading [17] and [27]. The Tor block node which proves the minimum randomness in the first part of the PoR acts as a leader in PBFT. SERP obtained in the PBFT consensus is forwarded to the user via the onion network.

B. GROUP SIZE

In this section, we discuss a suitable size of a group for a successful PBFT consensus, since suitable size of a group is critical for security and efficiency. The group size depends on the total number of nodes and malicious nodes in the network. The number of nodes n in a group is decided using the hyper-geometric probability density function [28].

If a random variable X follows the hypergeometric distribution, its probability mass function (pmf) is given by

$$p_X(x) = Pr(X = x) = \frac{\binom{k}{x}\binom{z-k}{n-x}}{\binom{z}{n}} \tag{4}$$

$$Total\ Probability = \sum_{x=(67\%\ of\ n)}^{n} Pr(X = x) \tag{5}$$

Here, X is a random variable, $Pr(X = x)$ defines the probability of x malicious nodes in a group, z is the total number of Tor block nodes in the network, k is the total number of malicious Tor block nodes in the network, n is the number of Tor block nodes in a randomly chosen group and x is the number of

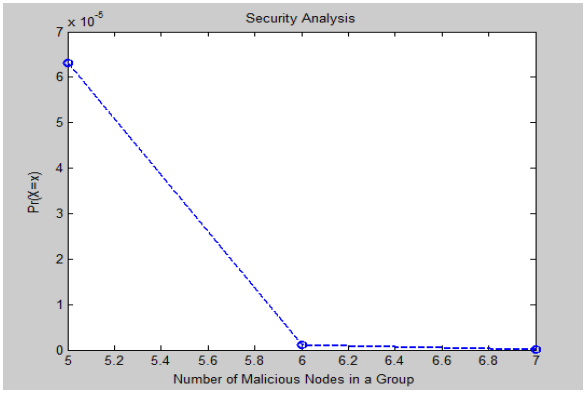

FIGURE 6. Graphical representation (z = 100 and k = 10).

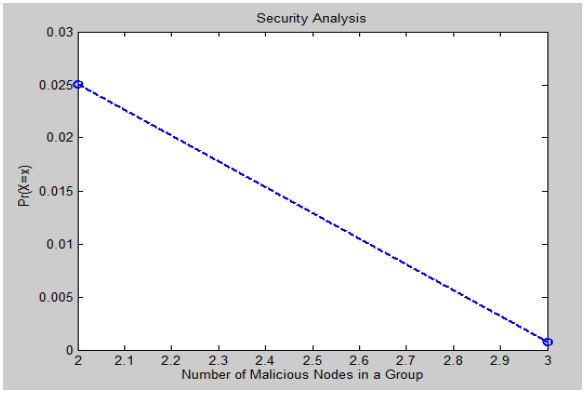

FIGURE 7. Graphical representation (z = 100 and k = 10).

malicious Tor block nodes in that group. Total Probability is defined as a probability of having $x = 67\%$ or greater in a group. We analyze the group size and security using varying values of k, n and z in a network and discuss them case by case. We select a random group of Tor block nodes and then calculate the Total Probability in that group. If the Total probability is less than 0.05, we consider it significantly secure because it is very unlikely, computationally hard and expensive to make a group of $x = 67\%$ or greater, with minimum randomness within the time period of 0.1 seconds.

1) For example, consider a network of $z = 100$ nodes and $k = 10$; That is, there are 10% malicious nodes. We select a random group of 7 nodes and then calculate the Total Probability. In this case, Total Probability is 6.4×10^{-5} which is less than 0.05. Hence, $n = 7$ in this case is suitable and has a significant level of security against 10% malicious nodes in the network. A graphical representation of the above result is shown in Figure 6.
2) For example, consider a network of $z = 100$ Tor block nodes and $k = 10$; That is, there are 10% malicious Tor block nodes. We select a random group of 3 Tor block nodes and then calculate the Total Probability. In this case, Total Probability is 0.0258 which is less than 0.05. Hence, $n = 3$ in this case is suitable and has a significant level of security against 10% malicious Tor block nodes in the network. A graphical representation of the above result is shown in Figure 7.
3) For example, consider a network of $z = 100$ Tor block nodes and $k = 50$; That is, there are 50% malicious Tor block nodes. We select a random group of 20 Tor block nodes and then calculate the Total Probability. In this case, Total Probability is 0.0392 which is less than 0.05. Hence, $n = 20$ in this case is suitable and has a significant level of security against 50% malicious Tor block nodes in the network. A graphical representation of the above result is shown in Figure 8.
4) For example, consider a network of $z = 100$ Tor block nodes and $k = 50$; That is, there are 50% malicious Tor block nodes. We select a random group of 10 Tor block nodes and then calculate the Total Probability. In this case, Total Probability is 0.1589 which is greater than 0.05. Hence, $n = 10$ in this case is not suitable and has no significant level of security against 50% malicious Tor block nodes in the network. A graphical representation of the above result is shown in Figure 9.
5) For example, consider a network of $z = 100,000$ Tor block nodes and $k = 50,000$; That is, there are 50% malicious Tor block nodes. We select a random group of 10,000 Tor block nodes and then calculate the Total

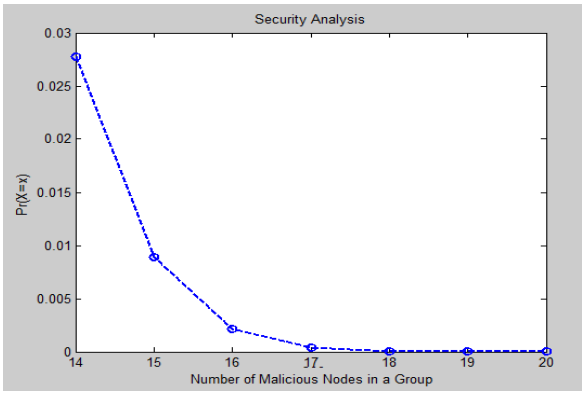

FIGURE 8. Graphical representation (z = 100 and k = 50).

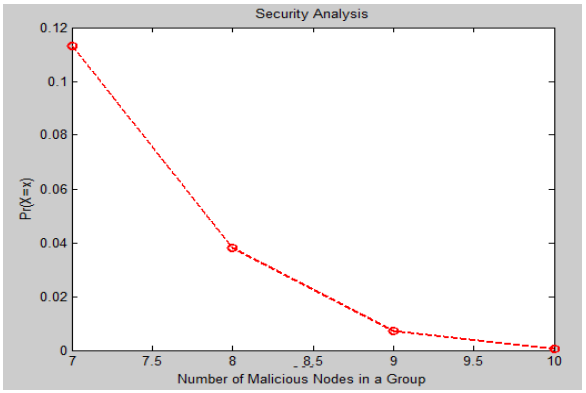

FIGURE 9. Graphical representation (z = 100 and k = 50).

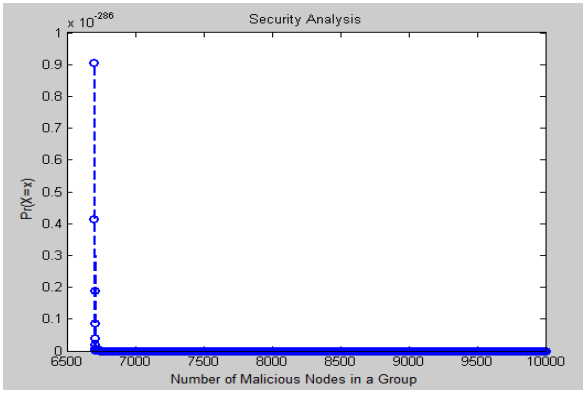

FIGURE 10. Graphical representation (z = 100,000 and k = 50,000).

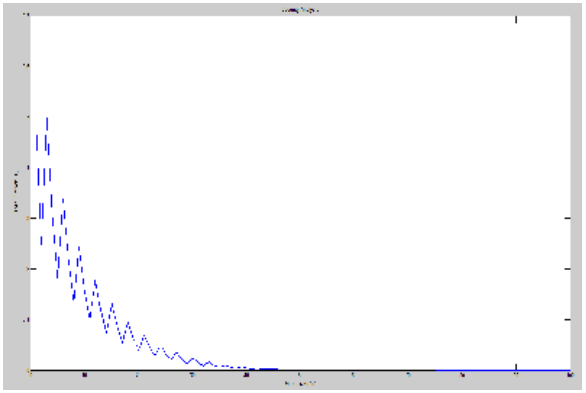

FIGURE 12. Graphical representation (z = 100 and k = 50).

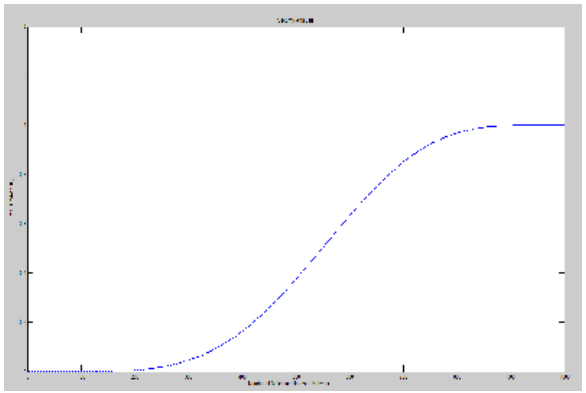

FIGURE 11. Graphical representation (z = 1000 and n = 10).

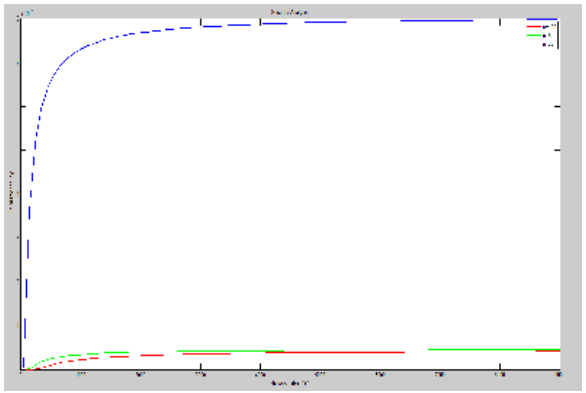

FIGURE 13. Graphical representation (k = 50% and n = 30, 50, 100).

Probability. In this case, Total Probability is 1×10^{-286} which is much less than 0.05. Hence, $n = 10,000$ in this case is suitable and has a significant level of security against 50% malicious Tor block nodes in the network. A Graphical representation of the above result is shown in Figure 10.

6) Now, we will analyze the effect of k by keeping n and z constant. For $z = 1000$ and $n = 10$; We change k from 1 to 1000. Figure 11 shows the relationship between the probability $P(X = x)$ and k, as k varies from 1 to 1000. Figure 11 shows that, with increase in the number of malicious Tor block nodes, the probability of a group being malicious increases for constant z and n. So, for a greater number of malicious Tor block nodes, the group size should be greater. In other words, with the increasing number of malicious Tor block nodes, for a fixed group and network size, the system security decreases.
7) To analyze the effect of varying n, we keep k and z constant. We change n from 1 to 100 for $z = 100$, and $k = 50$. Figure 12 shows the relationship between the Total Probability and varying n from 1 to 1000.
Figure 12 shows that, with the increasing n, the probability of a group being malicious increases at some points and decreases at other points. For example, the probability of being malicious for $n = 5$ is lower than that of being malicious for $n = 8$. The Total Probability is zigzagging for varying values of n. Hence, we can conclude that increasing the value of n does not always increase the level of security. The value of n for lower peaks having a Total Probability of less than 0.05 can be considered as an optimal group size for a given number of z and k.
8) Now we will see the effect of changing z by keeping n and the rate of k constant. A group size of $n = 100$ with $k = 50\%$ in a network of $z \geq 1000$ has a significant level of security. This is due to the Total Probability becoming constant for varying z by keeping n and k constant, with a value less than 0.05, as shown by Figure 13.

From the above examples, we can conclude that group size is proportional to the size of the malicious nodes in the network. If the network has a small percentage of malicious Tor block nodes, a small sized group can be made. Hence, the smaller the group size, the greater the efficiency of the network. This is due to the number of groups being large in the network and performing in parallel. As the number of malicious Tor block nodes increases in the network, the efficiency of the system decreases, providing a sufficient level of security. The Total Probability is less than 0.05 and remains

almost constant for a network of $z \geq 1000$, keeping k and n constant. In other words, Total Probability for a group being malicious remains constant. Hence, the community can decide the tolerance level for malicious nodes at the initialization of the network. When a network increases, it has no effect on the security level of the network, thereby keeping the group size and malicious nodes' rate constant. With an increase in size, the network will become more efficient. As such, since more groups can be made, more transactions can occur.

VI. ANALYSIS OF THE PROPOSED FRAMEWORK

In this section, we discuss our proposed framework in two perspectives. First, we provide various attack scenarios and explain the resilience of the proposed framework against those attacks. Second, we compare the proposed framework with the centralized approach and discuss communicational overhead and efficiency.

A. SECURITY ANALYSIS

1) *Privacy Preserving and Anonymity:* A key aspect of privacy in DLTs is the use of private and public keys. In DLTs the identity of a user is concealed behind powerful cryptography; Linking public addresses to individual users is difficult to achieve. This property of the DLTs is inherited in the proposed framework. Layer three only knows the public key of users and public keys do not give away personal data. Second, the onion network between layer 1 and layer 3 enhances the privacy and anonymity of the users by anonymous communication through onion routing.
2) *Transparency and Distributed Search:* DLTs are transparent, as users are far more open in the way they interact with the technology. All actions are unable to be traced back to an identity; While all transactions can be verified. Search algorithms used in the framework are available to each participant. Decision-making (master indexing and SERP ranking) is distributed across multiple actors in the network. This removes the need for centralized trust, thus providing transparency and distributed searches in the proposed framework.
3) *Community-driven Decision-making:* The community can decide algorithms used for SERP ranking and master indexing. This removes monopolized decision-making of the centralized systems, thus opening new paradigms of fair and non-monopolized decision-making.
4) *Single Point of Failure:* Since the framework includes distributed actors in the network, the failure of a few actors in the network will not affect the whole system as in contrast to the centralized approach.

B. ATTACK SCENARIOS

Theorem 1: A malicious Tor block node cannot compute the minimum $Total_{Ave}$ value by calculating the $Total_{Ave}$ value of all other Tor block nodes, which is computationally expensive and time consuming.

Proof: A malicious Tor block node tries to compute the minimum $Total_{Ave}$ value by calculating the $Total_{Ave}$ value of all other Tor block nodes which will be computationally expensive and time consuming; As each Tor block node has its own different N_i to provide its $Total_{Ave}$ result, within a certain period of time, and M and N_i are changing for each successful PoR, the malicious Tor block node will need to compute the $Total_{Ave}$ for each Tor node block in the network. Moreover, the malicious Tor block node cannot pre-compute $Total_{Ave}$ because M and N_i are changing continuously. Hence, the malicious Tor block node will be unable to generate its results within the given time period, even though it tries to compute $Total_{Ave}$ for all other Tor block nodes to make an attack.

Theorem 2: Malicious nodes cannot perform collusion attacks because of PoR.

Proof: Since PoR checks whether the same group is used more than once, although some Tor block nodes try colluding, they will not be able to prove the minimum randomness in the first part of PoR. Hence, collusion attacks will not be possible.

Theorem 3: A malicious node cannot perform a Sybil attack using its own multiple Tor block nodes because the owners of Tor block nodes can be identified using the equality test.

Proof: We deploy an identity verification method using the Public Key Encryption with Equality Test (PKEET) [20], which eliminates the Sybil attack by using the equality test. By using the equality test, it can check whether two ciphertexts encrypted under (possibly) different public keys contain the same message or not. The Sybil attack is prevented, because registration requires a real world identity (*ID*). The *ID* can be a citizen number, for example. Hence, fake identities would not work. The registration process is explained below.

1) An owner first contacts the registration authority (*RA*) of the network and provides his identity *ID* and a number k. Here, k is the number of the Tor block nodes he wants to own in the network.
2) RA encrypts the *ID* with its public key using PKEET, k times and gives k different ciphertexts $C_1, C_2, C_3, \ldots, C_k$ back to the user. Each ciphertext C_i is used as an identity of a Tor block node, as shown in Figure 14.

RA broadcasts the trapdoor to all the Tor block nodes. Each ciphertext C_i indicates the identity of a Tor block node. Using the equality test, other Tor block nodes can verify the *IDs* of different Tor block nodes, but the *IDs* remain anonymous. If the *IDs* of more than one Tor block nodes in the PoR are the same for a particular group, then the network considers it malicious and aborts the PoR. Hence, the PoR eliminates Sybil attacks.

A Tor block node needs to be registered with an identity before participating in the network. Someone can have more than one Tor block node in the network, but a Tor block node

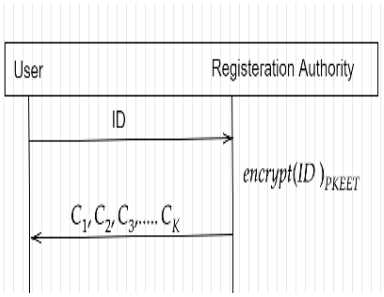

FIGURE 14. Registration.

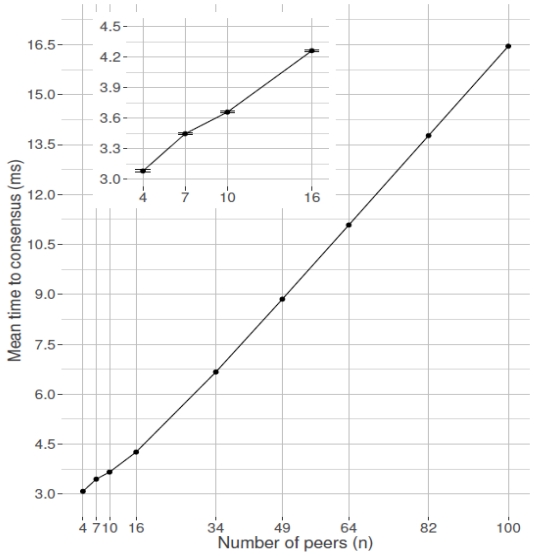

FIGURE 15. Time to consensus for a large number of peers.

cannot select more than one Tor block node with the same *ID* in the PoR for a particular group. The owners of Tor block nodes are kept anonymous but verifiable using PKEET.

VII. PERFORMANCE EVALUATION

In this section, we discuss the performance evaluation of the proposed framework.

A. SEARCH TIME AND RESOURCES

Search time plays a great role in search engines. Users usually prefer search engines with minimum search time and accurate results. In Google, every keyword query has to travel on average 15000 miles to a data center and back to the user with the answer. A single Google keyword query uses 1,000 computers in 0.2 seconds to retrieve an answer [29]. In [30], researchers modeled the PBFT consensus process using Stochastic Reward Nets (SRN) to compute the mean time to complete the PBFT consensus for networks up to 100 peers. Their results showed that the mean time of consensus for n = 100 is about 16.5 milliseconds (0.0165 seconds), as shown in Figure15.

As we have mentioned earlier, groups are made and continuously stacked. Whenever a user sends a keyword query in the mempool, it is assigned to the first group in the stack. Hence, no time is required to make groups for a keyword query. The search nodes will take time, as they will search for data related to the keyword query. Since search nodes are like the data centers of centralized search engines, we assume that they also take about 0.2 seconds to retrieve an answer to a keyword query. Similarly, Tor block nodes in a group take 0.2 seconds to rank the results provided by search nodes in SERP by relevance to a keyword query. The mean time to consensus for n = 50 is about 0.009 seconds; Hence byzantine consensus in the proposed framework requires $0.2 + 0.2 + 0.009 =$ .409 seconds to retrieve and rank results in SERP. Note that there is no significant difference in search time between the centralized and proposed framework. Moreover, centralized systems have limited resources (such as servers). The proposed framework is public and has more resources available as compared to the centralized system, which makes the proposed framework more efficient. According to Google, it can answer 63000 searches per second with about 2.5 million servers. As we have mentioned above, in the presence of 50% malicious Tor block nodes, the size of group $n = 50$ in a network of size $z \geq 1000$ is secure and the time required to answer a search query is 0.409 seconds. The number of Tor block nodes needed to answer a single query is 50 (group size). Let N_T denote the number of Tor block nodes required to answer 63000 queries per second. Then N_T is given by the following equations:

$$Number\ of\ searches\ per\ second = \frac{1}{0.409} = 2.44. \tag{6}$$

$$N_T = \frac{63000 \times 50}{2.44} = 1,290,983 \approx 1.3\ million. \tag{7}$$

From equation 7, it is clear that the proposed framework can answer 63000 searches per second with just 1.3 million Tor block nodes. Hence, based on the above discussion, we can conclude that the proposed framework is much more efficient and practical than Google.

B. PERFORMANCE MATRIX

The performance of the proposed framework depends on the number of groups; The greater the number of groups is, the greater the performance of the proposed framework will be. Hence, the number of groups in the proposed framework can outgrow as compared to the total number of servers of Google or centralized approaches. It means that more keyword quires can be answered. Our proposed consensus algorithm PoR is much more efficient because it is easy to verify and compute, if the Tor block nodes remain random and act honestly. Due to the distributed nature of groups, the efficiency of the proposed framework increases. There is always a trade-off between privacy, security and efficiency. The new economics of privacy helps explain this complex trade-off [31]. There is a trilemma between security, privacy and efficiency. This can be visualized in form of a triangle known as the DCS triangle, as shown in Figure 16. It shows

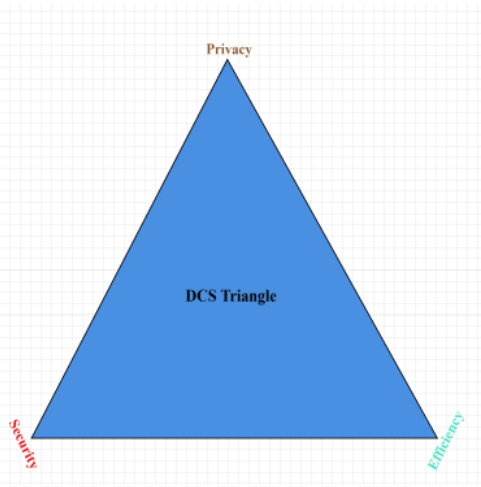

FIGURE 16. DCS triangle.

TABLE 1. Comparison with existing consensus algorithms.

Algorithm	Vulnerable to a 51% Attack	Sharding
PoW	Yes	No
PoS	Yes	Yes
PoR	No	Yes

TABLE 2. Comparison with a centralized approach.

Model	Centralized	Proposed
Transparency	No	Yes
Privacy	No	Yes
Single Point of Failure	Yes	No
Community-driven Decision-making	No	Yes
Search Time	Fast	Dynamic, depends on the number of groups

that it is impossible to achieve privacy, security, and efficiency simultaneously; You need to choose any two but not all. Hence, depending on the number of Tor block nodes in the network, the proposed framework can dynamically be fast. There can be some computational delays, but there are always trade-offs between privacy, security and efficiency. Matrix M and N_i contain the most recent m rows (m is 50%) and the initial 50% can be marked as historic after a fixed time period, for example, one year. Hence M and N_i are always scaled, which makes the proposed framework more efficient and secure.

C. COMPARISON WITH EXISTING CONSENSUS ALGORITHMS

Let's say that we split a network which is using PoW into BA and BB. BA has 90% of the hashpower and BB has 10%. It means that it will take only 5.1% of the total hashpower to control BB. A PoS based distributed ledger is shredded and shared amongst the entire network. Therefore, the entire network does not need to be involved with transactional validation. Sharding essentially speeds up the validation process as only the information that is needed is validated rather than having to check with the entire network. But both PoW and PoS are vulnerable to a 51% attack. Table 1 shows the comparison of PoR with PoW and PoS. Table 2 shows the comparison of the proposed framework with the centralized approach.

VIII. CONCLUSION AND FUTURE WORK

The proposed framework provides a transparent, decentralized and privacy preserving search engine. Our proposed framework exhibits several advantages over centralized approaches. For example, the community can do fair indexing and ranking of the search results, removing the monopoly of centralized search engines, which can pose threats to our privacy and manipulate the search results. The community can be awarded with benefits, and community-driven decisions can be made. The proposed framework removes the biased master indexing and SERP ranking. Our proposed consensus algorithm PoR has less computational and communication overhead than PoW and PoS. Moreover, the churns in the network cannot affect the efficiency of the proposed framework. The proposed framework shows its applicability by showing the trade-offs between privacy, security and efficiency, as compared to the centralized systems.

REFERENCES

[1] O. Tene, "What google knows: Privacy and Internet search engines," *Utah L Rev.*, vol. 4, p. 1433, Feb. 2008.

[2] S. Gupta, N. Rakesh, A. Thakral, and D. K. Chaudhary, "Search engine optimization: Success factors," in *Proc. 4th Int. Conf. Parallel, Distrib. Grid Comput. (PDGC)*, Dec. 2016, pp. 17–21.

[3] B. Morton, "The electronic commonwealth: The impact of new media technologies on democratic politics," *Government Publications Rev.*, vol. 16, no. 2, pp. 193–194, Mar. 1989.

[4] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. Int. Workshop Peer-to-Peer Syst.* Cambridge, MA, USA: Springer, 2002, pp. 53–65.

[5] M. Herrmann, K.-C. Ning, C. Diaz, and B. Preneel, "Description of the YaCy distributed Web search engine," KU Leuven ESAT/COSIC, IMinds, Ghent, Belgium, Tech. Rep., 2014.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 149–160, Oct. 2001.

[7] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *Proc. IEEE 25TH IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2006, pp. 1–12.

[8] M. Herrmann, R. Zhang, K.-C. Ning, C. Diaz, and B. Preneel, "Censorship-resistant and privacy-preserving distributed Web search," in *Proc. 14th IEEE Int. Conf. Peer-to-Peer Comput.*, Sep. 2014, pp. 1–10.

[9] Presearch-Inc. (Jul. 2017). The Community-Powered Search Engine. Presearch Whitepaper. [Online]. Available: https://whitepaper.io/document/210/presearch-whitepaper

[10] J. Buys, *DuckDuckGo: A New Search Engine Built From Open Source*, GigaOM OStatic Blog, San Francisco, CA, USA, 2010.

[11] L. Rainie, S. Kiesler, R. Kang, M. Madden, M. Duggan, S. Brown, and L. Dabbish, "Anonymity, privacy, and security online," *Pew Res. Center*, vol. 5, pp. 1–35, Sep. 2013.

[12] D. He, C. Chen, J. Bu, S. Chan, and Y. Zhang, "Security and efficiency in roaming services for wireless networks: Challenges, approaches, and prospects," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 142–150, Feb. 2013.

[13] J. Y. Hwang, S. Lee, B.-H. Chung, H. S. Cho, and D. Nyang, "Group signatures with controllable linkability for dynamic membership," *Inf. Sci.*, vol. 222, pp. 761–778, Feb. 2013.
[14] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, "Blockchain: A distributed solution to automotive security and privacy," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 119–125, 2017.
[15] C. Agbo, Q. Mahmoud, and J. Eklund, "Blockchain technology in healthcare: A systematic review," *Healthcare*, vol. 7, no. 2, p. 56, Apr. 2019.
[16] T. Lyons, L. Courcelas, and K. Timsit, "Blockchain and the GDPR," Eur. Union Blockchain, Tech. Rep., Oct. 2018.
[17] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002.
[18] A. Uyar, "Investigation of the accuracy of search engine hit counts," *J. Inf. Sci.*, vol. 35, no. 4, pp. 469–480, Apr. 2009.
[19] M. AbuTaha, M. Farajallah, R. Tahboub, and M. Odeh, "Survey paper: Cryptography is the science of information security," Palestine Polytech. Univ., Palestine, TX, USA, Tech. Rep., 2011.
[20] S. Ma, Q. Huang, M. Zhang, and B. Yang, "Efficient public key encryption with equality test supporting flexible authorization," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 3, pp. 458–470, Mar. 2015.
[21] M. Pilkington, "11 blockchain technology: Principles and applications," in *Research Handbook on Digital Transformations*, vol. 225. Cheltenham, U.K.: Edward Elgar Publishing, 2016.
[22] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Anonymous connections and onion routing," in *Proc. IEEE Symp. Secur. Privacy*. May 1997, pp. 44–54.
[23] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing for anonymous and private Internet connections," *Comm. ACM*, vol. 42, no. 2, pp. 9–41, 1999.
[24] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*. Pittsburgh, PA, USA: Citeseer, 2008.
[25] S. King and S. Nadal, *PPcoin: Peer-to-Peer Crypto-Currency With Proof-of-Stake*, vol. 16. China: Self-Published Paper, 2012.
[26] M. A. Imtiaz, D. Starobinski, A. Trachtenberg, and N. Younis, "Churn in the bitcoin network: Characterization and impact," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2019, pp. 431–439.
[27] T. Swanson, "Consensus-as-a-service: A brief report on the emergence of permissioned, distributed ledger systems," Allquantor, Tech. Rep., Apr. 2015.
[28] W. L. Harkness, "Properties of the extended hypergeometric distribution," *Ann. Math. Statist.*, vol. 36, no. 3, pp. 938–945, Jun. 1965.
[29] J. Dean, "Building software systems at Google and lessons learned," Stanford Univ., Stanford, CA, USA, Tech. Rep., Nov. 2010, vol. 10.
[30] H. Sukhwani, J. M. Martinez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of PBFT consensus process for permissioned blockchain network (Hyperledger Fabric)," in *Proc. IEEE 36th Symp. Reliable Distrib. Syst. (SRDS)*, Sep. 2017, pp. 253–255.
[31] W. Kerber, "Digital markets, data, and privacy: Competition law, consumer law and data protection," *J. Intellectual Property Law Pract.*, vol. 11, no. 11, pp. 856–866, 2016.

ALI RAZA received the B.S. degree in computer engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2017. He is currently pursuing the M.S. degree in electronics and computer engineering with Hongik University, South Korea. He is also a Researcher with the Information Security and Machine Learning Lab, Hongik University. His research interests include cryptography, cybersecurity, wireless networks, blockchain, and its applications in the Internet of Things.

KYUNGHYUN HAN received the B.S. and M.S. degrees in computer engineering from Hongik University, South Korea, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree in electronics and computer engineering. He is also a Researcher with the Information Security and Machine Learning Lab, Hongik University. His research interests include cybersecurity, machine learning, and blockchain.

SEONG OUN HWANG (Senior Member, IEEE) received the B.S. degree in mathematics from Seoul National University, in 1993, the M.S. degree in information and communications engineering from the Pohang University of Science and Technology, in 1998, and the Ph.D. degree in computer science from the Korea Advanced Institute of Science and Technology, South Korea. He worked as a Software Engineer with LG-CNS Systems, Inc., from 1994 to 1996. He worked as a Senior Researcher with the Electronics and Telecommunications Research Institute (ETRI), from 1998 to 2007. He worked as a Professor with the Department of Software and Communications Engineering, Hongik University, from 2008 to 2019. He is currently a Professor with the Department of Computer Engineering, Gachon University. His research interests include cryptography, cybersecurity, and artificial intelligence. He is also an Editor of *ETRI Journal*.