University of Southampton

Faculty of Engineering and Physical Sciences

Electronics and Computer Science

BlockDom: A Blockchain Based DNS Protocol Based on Sharding Architecture

by

George Giamouridis

January 2025

Supervisor: Dr Leonardo Aniello
Second Examiner: Dr Erisa Karafili

A dissertation submitted in partial fulfilment of the degree
of MSC Computer Science

University of Southampton

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
ELECTRONICS AND COMPUTER SCIENCE

Master of Science

by George Giamouridis

Web security has been and will always be one of the most important areas of research in computer science. So far, the web-based architecture seems to serve the needs of the users, but at the same time the security gaps are constantly increasing. Many of the security gaps occur in the OSI'S model protocols that are widely used on the web. The current thesis focuses on the vulnerabilities of the DNS (Domain Name System) protocol and presents a new solution which aims at the safest and most efficient use of this protocol.

DNS is a protocol that interprets a domain name as an IP address, meaning that converts a user-friendly domain into a computer-friendly IP address. The problems of the respective protocol differ as a result of affecting both its functionality and its security. DNS is a centralized protocol that depends solely on a central authority (ICANN) which is responsible for making decisions regarding its operation, resulting in various user's restrictions. Furthermore, the protocol is also threatened by popular web vulnerabilities and therefore users are constantly put at risk.

In this thesis we propose BlockDom, a blockchain based DNS, aims to completely transition from a traditional web protocol to a blockchain based solution. Essentially what we try to achieve is to implement a public blockchain network which is going to act like the current DNS protocol in a decentralized manner. The network will be able to handle all kinds of domain names, and whenever users request or register domains, the network will be responsible for performing such functions by using smart contracts. The aim of this project and therefore the main reason we decided to propose a blockchain based solution is to strengthen the security and performance of the current DNS protocol.

From time-to-time, similar works have proposed. However, our proposal is the first research approach aimed at replacing the entire DNS protocol with a blockchain-based solution.

Acknowledgements

I would like to express immeasurable appreciation and deepest gratitude for the help and support are extended to the following persons who in one way or another have contributed in making this thesis possible.

There are no proper words to convey my deep gratitude and respect for my thesis and research advisors, Dr. Leonardo Aniello. He has inspired me to become an independent researcher and helped me realize the power of critical reasoning. I would also like to thank Dr. BooJoong Kang for his unwavering support, his advises and his ideas which helped me to carry out and complete my thesis in a more scientific way.

I deeply thank my parents, Konstantinos and Stavroula for their unconditional trust, timely encouragement, and endless patience. It was their love that raised me up again when I got.

Finally, I cannot forget my friends who went through hard times together, cheered me on, and celebrated each accomplishment.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

Contents

List of Figures

Listings

Chapter 1

Introduction

1.1 Background

Every host participating in a TCP/IP network obtains a unique IP address. IP addresses identify computer or router interfaces, and each IP address contains information that is used to route IP packets. However, because users find it quite difficult to remember addresses of this type, they use symbolic names to refer to devices and networks. In order to communicate with one of these devices, it is necessary to know its IP address. Thus, instead of having to remember these addresses, it is common to use their symbolic names. Logically, this approach (address-names) works well on small networks with a limited number of devices. However, when we have transactions with the web, it is obvious that it is difficult to be able to memorize IP addresses.

The Domain Name System (DNS) is a distributed database on the Internet that allows translation between names and IP addresses. It is the mechanism of the Internet for naming resources we use in it, and that allows us to translate names into IP addresses and vice versa. It is a distributed database that is implemented in a hierarchy of multiple nameservers (DNS servers).

The DNS protocol includes three components. The **namespace**, the **servers** through which the namespace is made available, and the **resolvers** that ask the servers about the namespace. DNS data is stored locally, but is available worldwide. There is no computer with the entire DNS database. The DNS protocol is an application layer protocol that allows computers, hosts, routers, and DNS (Name Servers) servers to communicate to resolve names (translate names into IP addresses). It is a basic function of the backbone of the Internet, where DNS searches are done by any machine and any service. Results from remote name servers are temporarily stored in local memory to improve performance.

The Internet is conceivably divided into hundreds of different **high-level domains** that are broken down into **subdomains**, and so on, with many computers each. The areas can be represented by a tree, as shown in figure 1.1. The names of the regions form a hierarchy in such a way that the names are unique and easy to memorize. An organization is responsible for part of the namespace and can add additional levels to the hierarchy.

Each node in the DNS tree represents a **DNS name**. Each branch below a node is a DNS domain. The DNS domain may contain **hosts** or other subdomains.

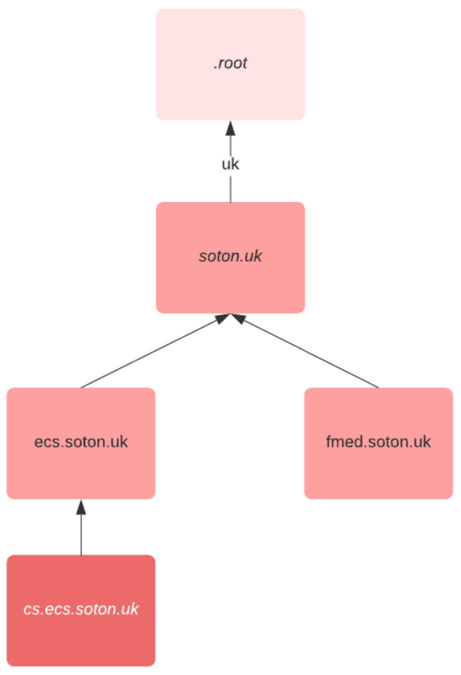


FIGURE 1.1: DNS hierarchy

The top of the tree is the root. The IANA (Internet Assigned Numbers Authority) is the official authority that manages the DNS root and below the top are the **top-level domains** (TLD) (1).

Below the first level area, there is a second level area, which usually identifies the organization or company to which the network belongs. These domains are called **second-level domains** domains, and each one is unique (2). The management of the namespace below the top-level domains has been delegated to organizations, which can further delegate the management of their subdomains. Each new subdomain corresponds to a **third-level** namespace.

DNS is organized as a distributed database that uses the client-server model. In order for DNS to work, it uses the nodes of this database, which are the Name Servers, which are located in different parts of the Internet, work together and inform users about which name corresponds to which IP address and vice versa. Figure 1.2 shows the namespace hierarchy which corresponds to a corresponding hierarchy of nameservers.

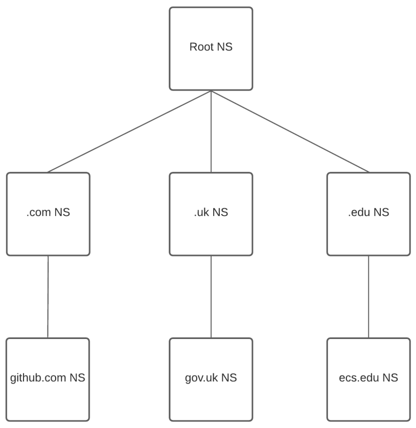


FIGURE 1.2: Hierarchical organized system

Each server is responsible for a compact part of the DNS namespace called a zone. The name server answers queries for the hosts on its domain. In figure 1.3 we see that each zone is nestled in a node of the tree. Zones are not domains. The zone is part of the DNS namespace that is generally stored in a file. The name server can split part of its zone and assign it to other servers (3).

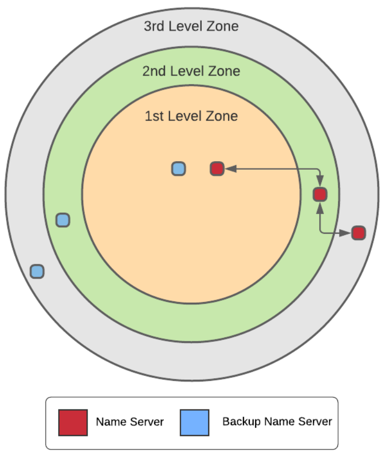


FIGURE 1.3: DNS zones

As we mentioned before, no DNS server has all the name matches of IP addresses. Multiple DNS servers may need to be queried to find a specific match. There must be a **primary server** and a number of secondary servers for each zone. The primary server maintains a zone file with the original zone information. The **secondary server** maintains copies of the data stored on the primary server.

The database can be updated dynamically by adding, deleting and modifying any information. When a host is added to a zone, the administrator adds the host information (IP address and name) to a file on the main server.

Almost every organization has a local name server, also known as the **default server**. When a query is asked, it is sent to the local server, which acts as an intermediary and forwards the query, if required. If the local name server has not found where to find the address that corresponds to a computer name, it asks the servers of other zones, reaching up to the root servers, if necessary.

The DNS protocol is of the client-server type and belongs to the application level of the TCP/IP model. The DNS client is called a **resolver**. The DNS protocol supports the conversion of names to addresses (resolution), as well as the updating of data between the name servers.

Name resolution is the process by which name analysts and servers work together to find data within the namespace. To find data, the name server only needs the name and IP address of the top (root) name servers. Top servers know all top-level areas and can indicate which servers to contact (4).

1.2 Aims and Objectives

In this thesis, we propose BlockDom, a blockchain based Domain Name System that aims to completely transition from a traditional web protocol to a blockchain based solution. Essentially what we try to achieve is to implement a public blockchain network which is going to operate like the current DNS protocol in a decentralized manner. The network will be able to handle all kinds of domain names, and whenever users request or register domains, the network will be responsible for performing such functions by using smart contracts. The aim of this project and therefore the main reason we decided to propose a blockchain based solution is to strengthen the security and performance of the current DNS protocol.

DNS protocol is a very big topic of discussion in terms of anonymity as users' requests are processed by the ISP and therefore may be tracked by them. Also, as long as DNS is a centralized protocol it is managed and maintained by a central authority (ICANN), which means that decisions about the way it operates are made solely by it (5). As a result, we can notice restriction of access to several websites, users blocking, and other similar decisions which are taken by a few people. Finally, we want to make sure that browsing the web is a secure process, as we know that DNS protocol is vulnerable to various attacks.

O1. Decentralization

Blockchain networks are decentralized networks, meaning that their functionality do not depend on any central authority. What is more, blockchain networks make use of strong cryptographic properties to ensure that data remain anonymous. Therefore, using such a network for handling domain names means that user's request will not be processed by any central authority. All the data included in the transactions will be processed by a peer network and stored in a secure public ledger where they will be remained anonymous as cryptographic techniques will be applied on them.

O2. Equal Participation

Lack of equal participation is another problem of centralized systems. A central authority is the only authorized entity that can take decisions regarding the network operations. Traditional DNS works that way and therefore decisions such as websites restrictions are taken solely by the ICANN organization. On the other hand, our blockchain network operates according to the P2P architecture. Thus, the users themselves are the ones who make the decisions in the network. The network will be public, meaning that everyone can join and leave at any time. All peers have the same rights, and they equally contribute to the network (6).

O3. Security

Finally, security in DNS is another big issue that our proposal aims to address. As long as DNS availability and functionality depends on a central server, attacks like DNS flood or cache poisoning are always likely to occur. The blockchain based DNS security relies on a strong consensus algorithm. This algorithm will be responsible for reaching consensus and ensuring that the network operates smoothly. The algorithm will be able to identify malicious transactions (domains that redirect to malicious websites) and prevent network confusion. Furthermore, DDoS attacks will not be a threat to the network's availability (7). In a centralized system, if the main server goes out of order (due to a DDoS attack) the whole network goes down. Instead, our network does not rely on any central server and therefore if one peer goes down, the whole network will continue to operate normally (8).

1.3 Thesis Overview

The current thesis consists of 5 chapters including introduction, literature review, design, testing and evaluation, and the conclusion.

1.3.1 Chapter 1: Introduction

In this section, we briefly introduce the DNS protocol and the way it operates by presenting its fundamental components and functionalities. We also address the main issues of such a protocol and present our potential solutions.

1.3.2 Chapter 2: Literature Review

The related work and current proposals will be shown in this chapter. We will first introduce some current blockchain based DNS solutions and briefly explain what they do. Finally, we will highlight the difference between them and BlockDom so to set some research questions on which we will rely in order to support our proposal.

1.3.3 Chapter 3: Implementation

In this chapter, we will present an extended view of the BlockDom design. We will first give an overall overview of the network's main functions, and then we will explicitly present the network implementation and how each component works.

1.3.4 Chapter 4: Testing and Evaluation

In the Testing and Evaluation chapter, we will demonstrate a series of testing in our network, and we will then evaluate the results and therefore the quality and resilience of the network. We will mainly focus on the performance and security evaluation by performing a series of measurements as well as simulation of some popular DNS attacks.

1.3.5 Chapter 5: Conclusion and Perspectives

This is the final chapter of the thesis, where we will wrap up our proposal. We will explain what we have achieved by implementing and dealing with such a protocol, and we will finally propose future improvements and plans.

Chapter 2

Literature Review

2.1 Blockchain Based DNS Alternatives

In the previous chapter, we saw that DNS is a centralized protocol operating under the ICANN organization, and we also mentioned some main issues of such protocol. The blockchain based DNS is one of the several approaches aiming to further improve the functionality, performance, and security of the DNS protocol since blockchain technology becomes more and more popular in our days.

The main idea behind a blockchain based DNS is presented in figure 2.1. Domain names alongside with their corresponded IP addresses are stored in a decentralized public ledger instead of a central database. The network is responsible for matching the requested domain names with the corresponded IP addresses by using smart contracts.

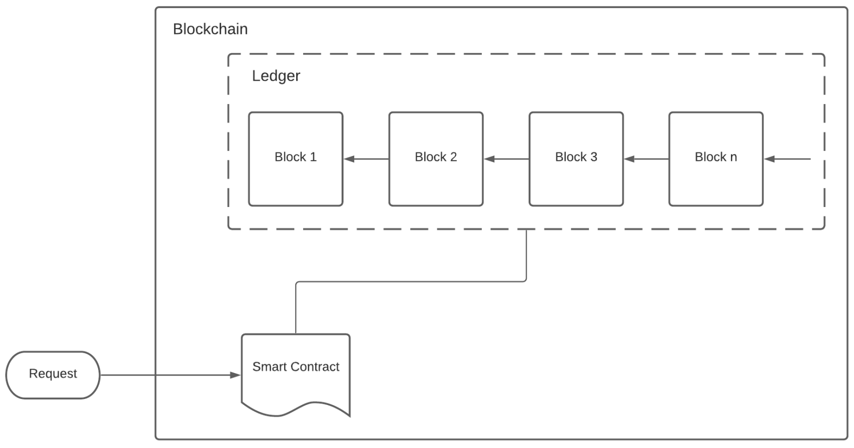


Figure 2.1: Blockchain based DNS

So far, several Blockchain-based DNS alternatives have been proposed. Below, we present the most popular blockchain based DNS alternatives.

Namecoin is a fork of Bitcoin, and it is the first Blockchain-based DNS. Namecoin uses the virtual .bit TLD, which is not officially registered in the current DNS system. That means that users need to install additional software in order to resolve .bit domains (9).

Blockstack operates on the top of the Bitcoin blockchain. That means that Blockstack cannot accommodate large data such as name-value pair information.

Nebulis is a global distributed directory, works in the same way as Blockstack. Nebulis uses IPFS or MaidSafe in order to replace HTTP requests with Ethereum processes, suitable for DNS requirements.

Emercoin (EMC) is another DNS alternative uses the technology of the NameCoin system. Emercoin provides a Distributed DNS, free Secure Sockets Layer certificates ENUMER and a distributed database for storing VOIP addresses a Decentralized torrent tracker in the framework of the qualification work we are only interested in a decentralized DNS system, called EMCDNS. The technology supports only 4 domain zones (10).

DNSTSM is a consortium blockchain solution that has been proposed in order to strengthen the DNS security by improving credibility of DNS resolution results by establishing a complete chain of trust. In DNSTSM, consortium blockchain is proposed as a solution of the reduction with regard to the impact of illegal access and complicity tampering on the DNS cache credibility (11).

TD-Root is another blockchain based DNS alternative that aims to address some DNS issues including tamper-proofing, security and deployability by proposing a trustworthy decentralized DNS root management architecture called based on permissioned blockchain. TD-Root uses a consensus algorithm, in which credence value and penalty mechanism are introduced to ensure the strong consistency, scalability, and security of the blockchain (12).

B-DNS is a blockchain based DNS that aims to improve DNS security and performance by using a Proof-of-Stake based consensus algorithm. B-DNS main achievement is that the probability of a successful attack on it is 1% of a successful attack on legacy DNS, the attack cost goes up a million times in B-DNS, and the attack surface of B-DNS is far smaller than that of legacy DNS (13).

2.2 Blockahin Based DNS Discussion

Advantages

In terms of Zooko's triangle theory (14), a blockchain based DNS meets all the three requirements (human-meaningfulness, security, decentralization).

Traditional DNS is administrated and controlled by a central authority, which is able at any time to make changes without any permission. On the other hand, a blockchain based DNS is managed exclusively by the nodes that they are all equal to each other, meaning that only they can make changes to the existing records.

With regard to security and decentralization, as long as traditional DNS is a centralized system, security issues like DoS or DDoS attacks can occur and affect the system. However, in a blockchain based DNS all records are stored locally on every node, meaning that there is no central point of failure in case of an attack. Furthermore, the way a blockchain stores records is able to identify any record modifications and prevent attackers for growing their malicious activity.

Disadvantages

As we can see, the main problem with the existing blockchain based DNS alternatives is that they all store and manage a specific type of domain, and they also have storage restrictions and limitations (15). That means that different blockchain networks needed in order to resolve different domain names (TLDs) as each network deals with a specific type.

In terms of performance, as long as new domain are appended to the ledgers, the blockchain becomes bigger from time to time, and as a result functionalities like node synchronization and searching become more and more inefficient.

Finally, there is always the threat of known blockchain attacks. The most common one is the 51% attack, where the attacker creates a malicious chain and is able to publish it without any problem. That means that all the other participants follow this chain which is now a trust chain and all transactions that are not existed in it, will be overridden directly (16).

2.3 BlockDom Versus Other Popular Blockchains

In order for a node to be able to join and participate in a blockchain network, it is necessary to download and install the corresponding software (client) on its machine. As for all traditional blockchain networks, so for BlockDom the users can find the appropriate client software online and start using the network after installing it. The client software will install the blockchain source code (smart contracts) on the new user machine so that it can use the network functions.

In traditional blockchains, when installing the software, the current directory with all the blocks and transactions is also installed on the new node. Thus, the nodes replace the central servers and the data storage on the network becomes decentralized. Nodes now have direct access to the public blockchain ledger as it is stored in their local storage, so they can at any time confirm any kind of transaction.

In BlockDom blockchain, the installation and synchronization of the public ledger is not done during the installation of the client software. This is because the network is made up of many smaller sub-blockchain, which retain their own public ledger. When a node completes the software installation, the network uses the smart contracts to decide which sub-blockchain to join. When it is assigned to a sub-blockchain, then again with the use of a smart contract, the public ledger of that network will be set up and synchronized. Thus, the new node will store in its local storage only the public ledger of the sub-blockchain in which it is included, and not a single ledger with all the network's transactions. The node is able to interact with otehr nodes in the same sub-blockchain (confirm their transactions or propagate its own transactions) and can get access only to the ledger of this sub-blockchain (see "Chapter 3").

2.3.1 Comparison With Bitcoin

Clients

BlockDom follows Bitcoin's orientation, meaning that it is a public blockchain network available for everyone wishes to participate. Users do not need to provide any personal information in order to participate in it. Everyone can join the BlockDom network by downloading the BlockDom client software. Similarly to Bitcoin, users do need to synchronize the blockchain in order to be able to confirm transactions and also make new transaction. The main difference in BlockDom is that users do not need to synchronize the entire blockchain, but only the sub-blockchain to which they are assigned (see "Chapter 3").

Network Architecture

Bitcoin is a P2P network where each node maintains a shared public ledger on which the entire Bitcoin network is based, as all confirmed transactions are stored in it. The difference between BlockDom and Bitcoin is that BlockDom nodes does not maintain a single public ledger. As long as BlockDom is a blockchain consists of smaller sub-blockchain based on the sharding architecture, there is not a single shared ledger. Instead, each sub-blockchain maintains its own ledger where the confirmed transactions are stored.

Keys and Wallets

In Bitcoin, an asset's ownership is established through digital keys, addresses and digital signatures. The digital keys are not stored in the network, but are instead created and stored by end-users in a file, or simple database, called a wallet (17). On the other hand, BlockDom does not use wallets, as users do not own any asset for which they should prove their ownership. Furthermore, new transactions are not signed by the originator, but instead the validation is a process performed on the smart contract side.

Mining and consensus

In Bitcoin, Proof-of-Work is the process through which the network achieves consensus. PoW is the process by which new cryptrocurrencies are issued as a reward for the nodes that operate as miners. The main difference between Bitcoin and BlockDom is that BlockDom does not use mining as part of the consensus protocol. Thus mining nodes are not rewarded with any cryptocurrency and therefore the network does not issue or maintain any valuable asset.

2.3.2 Comparison With Ethereum

In general Ethereum blockchain shares many common elements with other open blockchains such as the P2P network architecture, a Byzantine fault–tolerant consensus algorithm for synchronization of state updates, the use of cryptographic primitives such as digital signatures and hashes, and a digital currency called ether. Ethereum's purpose is not primarily to be a digital currency payment network. While the digital currency ether is both integral to and necessary for the operation of Ethereum, ether is intended as a utility currency to pay for use of the Ethereum platform as the world computer.

Ethereum is designed to be a general-purpose programmable blockchain that runs a virtual machine capable of executing code of arbitrary and unbounded complexity. Ethereum's language is Turing complete, meaning that Ethereum can straightforwardly function as a general-purpose computer (18).

2.3.3 Smart Contracts

The concept of smart contracts has evolved, especially after the introduction of decentralized blockchain platforms with the invention of Bitcoin in 2009. Smart contracts are immutable computer programs that run deterministically in the context of an Ethereum Virtual Machine as part of the Ethereum network protocol.

BlockDom uses smart contracts like Ethereum does in order to be able to support its fundamental operations. When a new domain is requested or is about to be registered in the blockchain, a smart contract is responsible for carrying out these requests.

2.3.3.1 Transaction Gas

In Ethereum, gas is used to control the amount of resources that a transaction can use, since it will be processed on thousands of computers around the world. The Turing-complete computation model requires some form of metering in order to avoid denial-of-service attacks or inadvertently resource-devouring transactions. Ethereum blockchain requires from the users a fee called gas in order to use smart contracts (18). Instead, in BlockDom transactions are performed free of charge because the execution of the smart contracts is one of the core functionalities of the network. Also, as we have already mentioned, the network does not maintain any cryptocurrency and therefore its functionalities can be used without any charge.

Chapter 3

BlockDom Design

3.1 Requirements Analysis

3.1.1 Setting Up The BlockDom Client

BlockDom client is a Python app build by using the Flask. Flask is a micro-framework, meaning that it can provide libraries, tools, and technologies that allow us to build our application. The reason we used Flask for the development of BlockDom is because micro-frameworks have little dependencies to external libraries. That makes them light, and also there are little dependencies regarding updates for security bugs.

The client software can be found and downloaded here. Before running the application, you are required to download Python (version 3 or later) and Flask.

BlockDom client also uses some external Python packages in order to be able to add more functionality to the smart contracts. Make sure that you have installed the following Python packages before running the BlockDom client. **json, datetime, hashlib, socket, random, webbrowser**. You can see all the installed packages by using the listing command ***pip list***.

3.1.2 Running the BlockDom Client

To run the BlockDom client, simply open the directory you have downloaded from GitHub and run the following command:

```
python app.py
```

A url that redirects to the localhost will be shown up. Copy and paste it to your web browser to use the app.

3.2 BlockDom Overview

Current Blockchain-based DNS alternatives introduce several issues related to inefficient performance and security. In order to be able to deal with all these problems, we propose **BlockDom**, a public blockchain network which is going to handle all kinds of domain names.

The functionality of our network is similar to the existing proposals. The main difference is that our network architecture is based mainly on the **sharding** architecture, which essentially allows us to maintain smaller sub-blockchains and distribute the data (domains) in them. Each sub-blockchain will store a specific number of TLD's and when a user requests a name, the corresponding smart contract will search for that name within that specific sub-blockchain. By doing so, we will be able to significantly reduce the searching time and also the time that a new node will need to synchronize the blockchain. Thus, our network will be able to handle different types of domain names (TLDs) and will not be limited to just one specific type.

At this point we should mention that although BlockDom is a public blockchain network, it does not maintain a cryptocurrency. The main goal of this proposal is to introduce an improved protocol rather than a financial asset or a commercial product. Thus, users are not required to pay any fee for making transactions (transaction fees) as BlockDom is a potential web protocol instead of a service.

3.3 Fundamental BlockDom Components

BlockDom protocol consists of the following fundamental components:

P2P Network: Like all blockchain networks, BlockDom uses a P2P architecture. All nodes are equal meaning that they all have the right to request but at the same time they all contribute to the network.

Clients: Clients are users that participate in the network. We mentioned before, the network is public, meaning that everyone can join at any time. Whenever a user joins the network and synchronize the blockchain, it is considered as a client. Clients can request or register domain names, but at the same time are responsible for confirming other transactions.

Shards: In sharding architecture, shards are sub-entities of a main entity that use the same functionalities. In our network, a shard will be a sub-blockchain which is going to maintain and handle a limited number of TLDs.

Consensus Protocol: The consensus protocol is a set of rules that are going to be followed in order for the network to operate with fairness. Essentially, this is an algorithm that decides whether a block with a transaction is going to be included in the blockchain or not.

Transactions: Transactions are the main assets of the network. A valid transaction (which is going to be included in a block) contains the domain-IP pair. If the block that contains this transaction successfully appended in the blockchain, then this domain will be available to the network.

Smart Contracts: Smart contracts are responsible for the smooth network operation. BlockDom uses three smart contracts, as we will see later, each of them serving a different purpose. Smart contracts are the logic of the network and are responsible for supporting the functions it offers.

In the next sections, we are going to explain in details how each component operates within the network.

3.4 Overall Architecture

In this section, we are going to present the overall BlockDom architecture. We will first present the main network operations, and then we will focus on them in more details.

At the initial stage, the network will be consisted of one hard-coded sub-blockchain. This sub-blockchain will contain some blocks with transactions (domain-IP pairs) and will be the only record of the network status tracker.

The network uses smart contracts in order to serve different request and operate smoothly. There are three smart contracts that operate across the different sub-blockchains.

- **Client Role:** Specifies the sub-blockchain that the clients has to synchronize
- **Register Domain:** Registers a new domain in a sub-blockchain
- **Domain Tracker:** Searches the blockchain for requested domains

When new nodes join the network, the **Client Role** contract is responsible for assigning them to a sub-blockchain according to a distributed record that tracks the network status. After a node is assigned to a sub-blockchain, it has to synchronize this blockchain in order to be able to validate new transactions. After that, the node will be able to search for domains or create new transactions.

The network status is tracked in real-time by a distributed record called **NST** (Network Status Tracker). NST contains information about all the active sub-blockchains of the network, and whenever a new change occurs, the NST is updated. Each record of the NST contains the following information:

- **Blockchain ID:** The ID of the sub-blockchain
- **Active Users:** How many active users exists in this sub-blockchain
- **TLDs:** An array with all the TLDs that are registered in this sub-blockchain
- **Active TLDs:** The number of TLDs that are registered in this sub-blockchain

The Client Role contract gets access to the NST in order to find the sub-blockchains with the fewer active users. When such blockchain is found, the new user is assigned to it and the NST is updated.

When a new TLD is register to a sub-blockchain, it is recorded in the **TLDs** array. The TLDs array is considered **full** when five different TLDs has been appended to it. That means that this sub-blockchain is full and can only accept new domains with TLDs that exists in its TLDs array.

The main BlockDom functionality is presented in figure 3.1. After that, the overall flow of the two main functionalities of the BlockDom protocol (searching domains across the sub-blockchains and registering new domains) are shown in the diagrams.

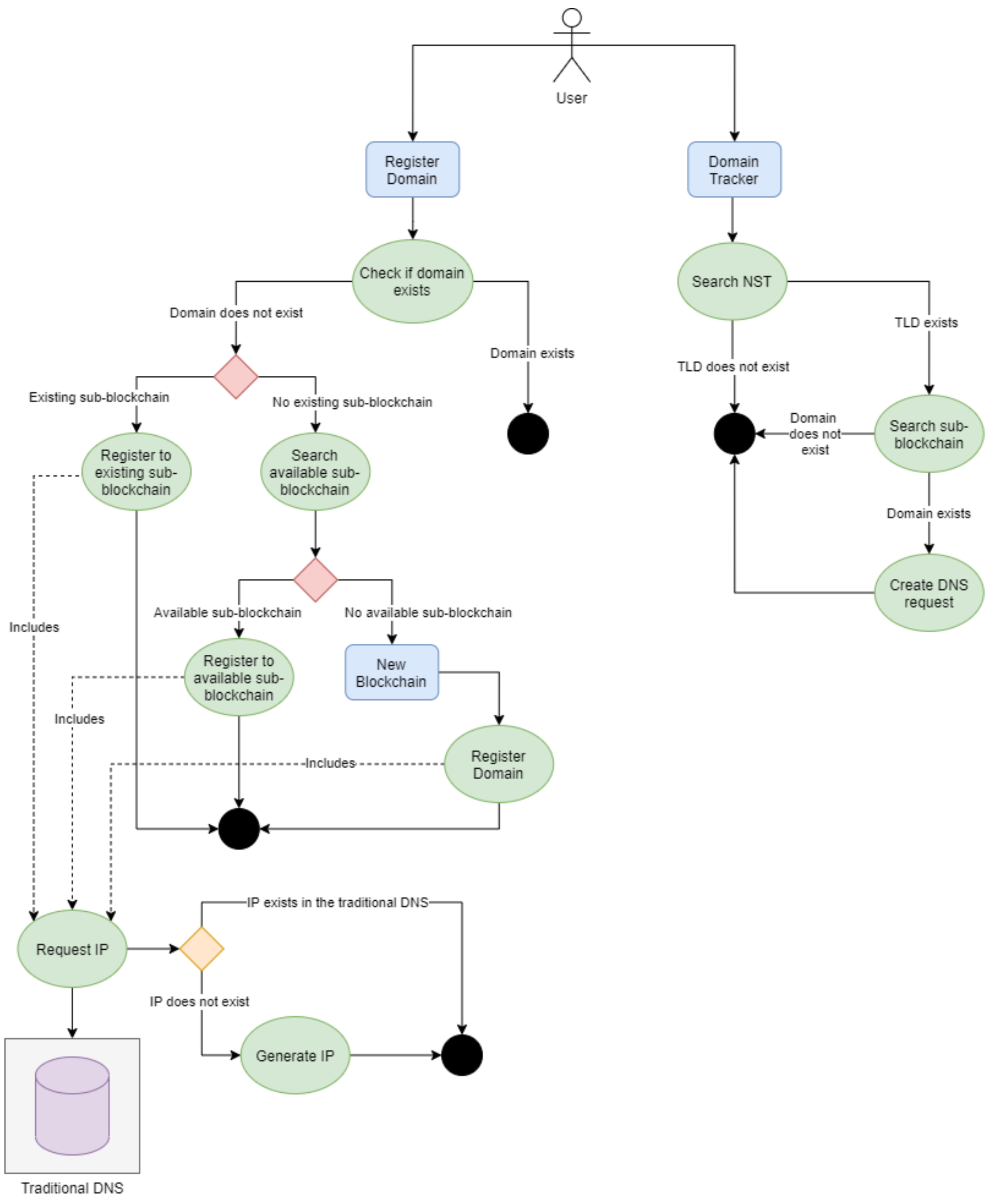


FIGURE 3.1: Overall BlockDom architecture

3.4.1 Registering Domains

Figure 3.2 demonstrates how a new domain is registered to a sub-blockchain.

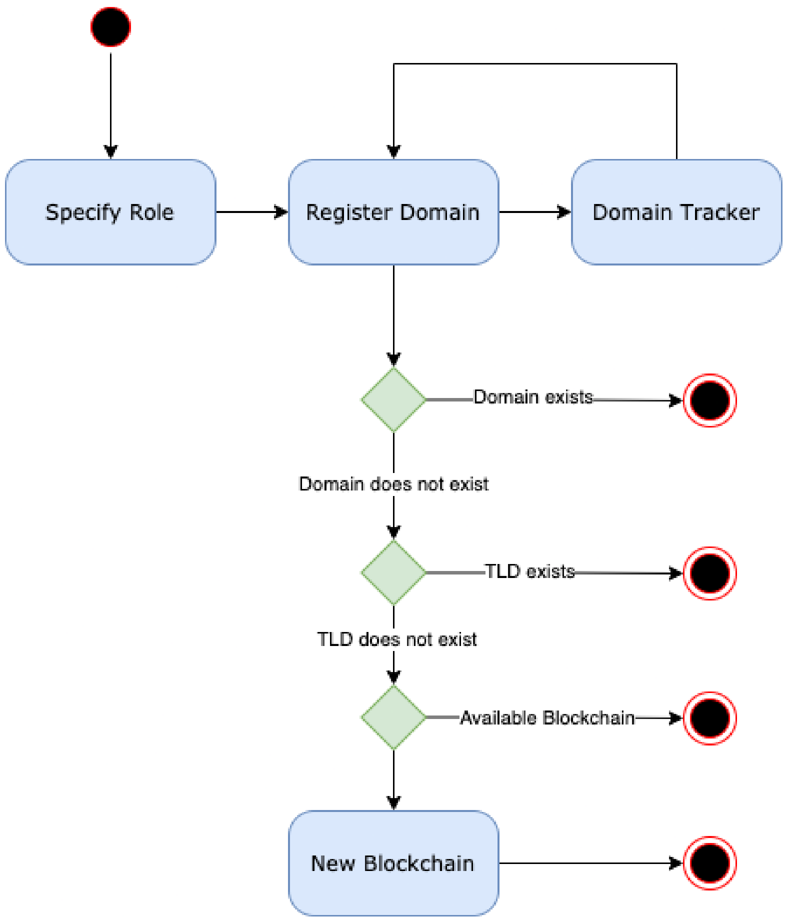


FIGURE 3.2: Registering domain flow

Before the requested domain is registered, the **Register Domain** contract will first conduct a search in order to check if it already exists in a sub-blockchain. If the domain exists, the contract return a message to the user to inform him that this domain is unavailable. If the domain does not exist, the contract checks whether the TLD exists or not. If the TLD exist in a sub-blockchain, the contract registers the new domain to this sub-blockchain. If the TLD does not exist in any sub-blockchain, the contract checks if there is any available sub-blockchain in order to register this TLD (as we mentioned before, sub-blockchain can store up to five different TLDs). If an available sub-blockchain is found, the contract will register the TLD in this sub-blockchain alongside with the domain. If there is any available sub-blockchain, the contract will call the **newBlockhain** contract to generate a new sub-blockchain and will store the new domain alongside with its TLD in it.

3.4.2 Searching Domains

Figure 3.3 demonstrates how the network performs the searching process of a specific domain name.

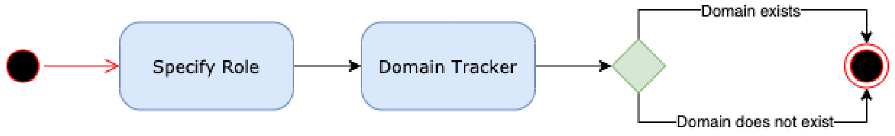


FIGURE 3.3: Searching domain flow

The requested domain will be processed by the **Domain Tracker** contract. The contract will first check if the TLD exists in the NST. If the TLD exists, then it will search the sub-blockchain in which this TLD is stored in order to find the block with the requested domain. If the TLD does not exist (that means that the domain does not exist) the contract returns a NXDOMAIN (19) code to the user. If the contract manages to find the requested block, it will return the domain-IP pair to the user. Otherwise, it will again return a NXDOMAIN code.

Now that we have briefly introduced the way BlockDom operates, we are going to focus on each component separately and explain its functionalities explicitly.

3.5 Clients

A client could be any user that wishes to participate and use the BlockDom network. As we mentioned before, BlockDom is a public blockchain, meaning that anyone can join at any time. All clients participating in the network have the same rights. Clients can be defined as users who have synchronized a specific sub-blockchain in their device and perform requests in the network. In order for a client to be able to participate in the network, the appropriate software is required. The software can be found online and anyone can download it and participate in the network.

When a new client downloads the software for the first time, the **Client Role** contract will specify the sub-blockchain which is needed to be synchronized according to the NST (we will explain how the network discovery operates later). After the synchronization finishes, the user has successfully joined the network and is now ready to perform request in and make new transactions.

3.6 The Network

3.6.1 Sharding Architecture

BlockDom's main difference from other traditional blockchains is that blocks with transactions are not stored in a single public ledger. Instead, BlockDom uses the **sharding** architecture and therefore multiple ledgers are maintained in different sub-blockchains (20). As a result, BlockDom can be considered as a collection of autonomous sub-blockchains, each of it operating on its own. Each sub-blockchain maintains a ledger where blocks with transactions are stored in it. Those transactions are related only to this sub-blockchain, and they do not have any relationship with any other sub-blockchain.

Although sub-blockchains are fully autonomous and have nothing to do with each other, they are able to exchange data through smart contracts. As we will see later, in some cases a sub-blockchain may not be able to host a new transaction if it is full. Thus, the transaction should be forwarded to an available sub-blockchain where it can be hosted. A smart contractor is responsible for this transfer. Remember that all nodes, regardless of the sub-blockchain in which they participate, use the same smart contracts that were installed during the installation of the client software.

All sub-blockchains are designed in such a way that they can store up to five different TLDs and an unlimited number of domains followed by one of these TLDs. When a new node is assigned to a sub-blockchain, it synchronizes the ledger of this sub-blockchain and is responsible for validating transactions only within this sub-blockchain.

Figure 3.4 shows two different shards (sub-blockchains). The blue shard stores domains with the .com .net .gr and .edu TLDs. The green shard stores domains with the .uk .soton and .org TLDs. The two shards have no relationship to each other. They both maintain their own ledger with their own blocks and transactions.

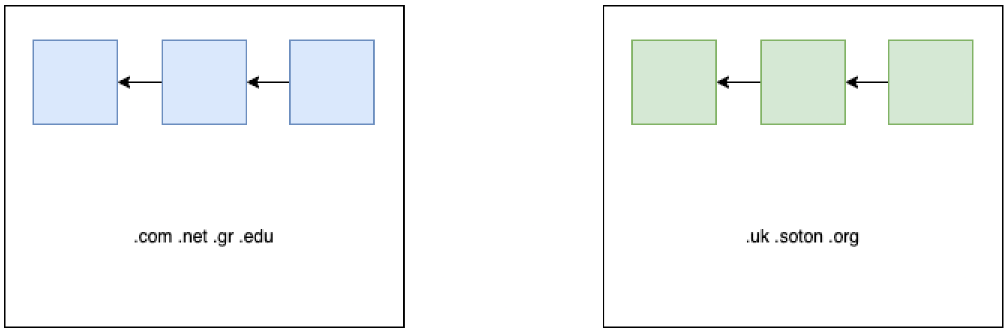


FIGURE 3.4: Two different shards

3.6.2 Nodes

In BlockDom although all nodes are equal according to the P2P rules, they have different functionalities according to their role. There are two different node types. The **Full Node** and the **Seeder**.

A Full Node is a regular node participating in a sub-blockchain and maintaining its ledger. This node is able to verify new transactions and propagate them to the rest network. Full nodes are simple users running the BlockDom client in their devices. A Full Node is connected to other nodes in the same sub-blockchain according to the flat topology of the P2P architecture. This node is also able to making new transactions and broadcasting them to the sub-blockchain. Full Nodes have the ability to join and leave the network anytime.

A Seeder node is a long-running, stable node that is listed in the BlockDom client as a *seed node*. Each Seeder maintains the following data:

- The blockchain (ledger) of the sub-blockchain
- A list with all the active node of the sub-blockchain

Each sub-blockchain has a Seeder node. When a new node joins the sub-blockchain, the Seeder is responsible for establishing a TCP connection between the new node and some active network's nodes. Seeders are also responsible for maintaining the blockchain of the sub-blockchain in case there are no active users in it (21).

In BlockDom when a new transaction is made and all sub-blockchains are unavailable, a new sub-blockchain is needed in order to host the transaction. When a new sub-blockchain is generated, the corresponding smart contract also generates a Seeder node and assigns it to the new sub-blockchain. The Seeder will be always available in order for new nodes to be able to connect with other nodes and participate in the network.

3.6.3 Network Discovery

When a new node joins the network for the first time, the network is responsible for assigning this node to a sub-blockchain. All nodes are distributed equally to all the active sub-blockchains in order for the network to have an equilibrium.

Assigning a user to a sub-blockchain is the first operation that happens when a user joins the network. For this process, the Client Role contract is invoked. This contract gets access to the NST that contains information with regard to the current network status. The NST is updated in real time whenever a new user joins or leave the network or new domains are registered to the network or new sub-blockchains are generated. Below is an instance of the NST.

```
"NST": [
    {
      "blockchain_id": "639ecfda46dc8f4f9795d892d16
                        1dd521d7f8cc47bcbf60f6942e6d9c7f949a0",
      "active_users": 57,
      "tld": ["com", "org", "net", "gr", "uk"],
      "active_tld": 5
    },
    {
      "blockchain_id": "c052b93579a91556e1158f07d0f91e44
                        0c5422a7cfd542e9814b827918c55e14",
      "active_users": 19,
      "tld": ["gov", "edu", "mil", "bio"],
      "active_tld": 4
    }
]
```

Listing 3.1: NST instance

The NST in listing 3.1 represents a network with two sub-blockchains. The first sub-blockchain has 57 current active users and 5 active TLDs. This is a full sub-blockchain, meaning that only domains with the .com .org .net .gr .uk TLD can be registered in this sub-blockchain. If a user wants to register a new TLD, the network will look for another available sub-blockchain to do so. The second sub-blockchain has 19 current active users and 4 registered TLDs. This is an available sub-blockchain and if a user wants to register a new TLD, this sub-blockchain is available to host it.

When the Client Role contract is invoked, it goes through all the NST records and tries to spot the sub-blockchain with the fewer active users. In the case of the above NST, the contract will choose the second sub-blockchain as it has 19 active users and the first one 57. The new user will be assigned to the second sub-blockchain. The *active_users* field will be increased by one and become 20. The **Client Role** contract which is presented in listing 3.2 is responsible for identifying the sub-blockchain with the fewer active users, assigning the user to it and updating the NST.

```
contract ClientRole():
    open(NST):
        min = 1000

        for i in records:
            if records['active_users'] < min:
                min = records['active_users']
                min_id = records['blockchain_id']

        for i in records:
            if records['blockchain_id'] == min_id:
                records['active_users'] += 1

    update(NST)
```

Listing 3.2: Client Role smart contract

In line 2 the contract accesses the NST in order to read its records. In lines 5-8, the contract searches for the sub-blockchain with the fewer active users. After the sub-blockchain assignment, in lines 10-12 the contract increases the active users field and finally in line 15 updates the NST.

When the node successfully assigned to a sub-blockchain, it must discover other nodes on the same sub-blockchain in order to participate in the network. The Seeder node is responsible for helping the new node in discovering others. The Seeder maintains a full list of all the active nodes participating in the sub-blockchain. When the new node joins the sub-blockchain for the first time, the seeder will randomly pick some nodes and send their IP to the new node. To connect with these nodes, the new node will establish a TCP connection and start a *handshake*.

Once the connection is established, the new node will send a message containing its own IP address, to its neighbors. The new node will be added to the list of the seeder, so the next time a new node will join the network will be able to connect with it.

3.7 Transactions

A transaction is originated whenever a user registers a new domain. Each transaction contains the domain-IP pair, and stored in a unique block inside a sub-blockchain. The first step of the transaction lifecycle is the transaction origination. In order for a transaction to be generated and included in a block, the **Register Domain** contract is invoked. This contract is responsible for identifying the sub-blockchain in which the new transaction will be hosted.

At first, the Register Domain contract conducts a search in the sub-blockchains (at the same way the Domain Tracker contract does) and checks if the requested domain already exists. If so, the user is informed by an error message. In case the requested domain is available, the Register Domain contract should find a sub-blockchain to host it.

When the transaction is successfully propagated to the whole network, it will then be mined by a node, appended to a block and finally recorded on the sub-blockchain. Once the block with the transaction is appended to the blockchain it is permanent part of this specific sub-blockchain, but it can be accessed (request the domain) by any node in any sub-blockchain through the smart contracts.

3.8 Transaction Structure

As shown in table 3.1, BlockDom transactions are data structures containing the domain-IP pair that is going to be appended in the blockchain.

Field	Description	Size
Domain	Domain that is going to be registered	30 bytes
IP	Corresponding domain IP address	15 bytes

Table 3.1: Transaction structure

The maximum size of a transaction is 45 bytes. Table 3.2 shows how the domain name size is split in the transaction. The 30 bytes of the domain name means that it should be a maximum of 26 bytes, as the "." and the TLD are 4 bytes together.

26 bytes	1 byte	3 bytes
domain	.	TLD

Table 3.2: Domain name structure

The remaining 15 bytes come from the IP field. Table 3.3 shows how the IP size is split in the transaction. The IP can have a maximum of 12 numbers (four triads) which is 12 bytes, and three dots which is 3 bytes.

3 bytes	1 byte	3 bytes	1 byte	3 bytes	1 byte	3 bytes
195	.	145	.	213	.	223

Table 3.3: IP address structure

The size of a transaction is checked through a validity test performed by the nodes before it is propagated in the network, as we will see later. In listing 3.3 we see a valid transaction ready to be propagated to the network.

```
{
    domain : "google.com"
    IP     : "217.160.0.201"
}
```

Listing 3.3: Valid transaction

3.8.1 Broadcasting and Propagating Transactions

After a node creates a new transaction, it has to broadcast it to the rest network before it is mined and appended to the blockchain. In order for a transaction to be propagated in the network, all nodes should perform a validity check. The node that created the transaction will propagate it to all its neighbors and its neighbors will propagate it to their neighbors and so on, until the transaction is propagated across the whole network.

Each node will perform a validation according to a checklist of the following criteria in order to confirm that the transaction is valid and can further be propagated.

- The transaction syntax must be correct
- The transaction data structure must be correct
- The transaction size should be less than the maximum transaction size
- The transaction size should be less than the maximum block size

If the transaction meets all the criteria, the node will propagate it to the other nodes to which it is connected, and a success message will be returned synchronously to the originator. If the transaction is invalid, the node will reject it and synchronously return a rejection message to the originator (17).

For example, the IP address of the transaction in linting 3.4 is syntactically incorrect. When a node receives the following transaction will discard it in order not to be propagated across the rest network.

```
{
    domain : "example.com"
    IP     : "184.1.3"
}
```

Listing 3.4: Invalid transaction

The node should be able to identify the issue in this transaction and return a reject message to the originator before it is propagated to the rest network. If the transaction is valid (no node returns a reject message) it is ready to be propagated in the network and mined into a new block.

Notice that the transaction validation process takes place in the network layer. When the transaction is validated will then moved to the consensus layer where it will be mined and included in a block.

3.8.2 Hosting Transactions

There are three different cases in finding a sub-blockchain to host the new transaction.

- The TLD of the requested domain is already hosted in a sub-blockchain and therefore the new transaction can be hosted in it
- The TLD of the requested domain does not exist. An existing sub-blockchain is available (hosts less than five TLDs) and therefore the new TLD can be registered in it and host the new domain.

- The TLD of the requested domain does not exist. All sub-blockchains are unavailable (they all host up to five TLDs). A new sub-blockchain should be generated to host the new domain.

```
1  contract registerDomain():
2      open(NST):
3          if tld_exists:
4              open(blockchain):
5                  mining(blockchain)
6          else:
7              blockchain_to_append = isFull()
8              if blockchain_to_append == None:
9                  new_blockchain()
10             else:
11                 mining(blockchain_to_append)
```

Listing 3.5: Identifying the host sub-blockchain

The code in listing 3.5 presents a part of the Register Domain contract, which is responsible for identifying the sub-blockchain in which the new transaction will be hosted. The NST is accessed in order to check whether the TLD of the requested domain exists or not. If the TLD exists (lines 3-5) the contract will send the transaction to that sub-blockchain and append it in there. If the TLD does not exist (lines 6-11), the contract will call the **isFull** function as shown in listing 3.6, which is responsible for finding available sub-blockchains.

```
1  function isFull():
2      open(NST):
3          for i in records:
4              if records['active_tld'] < 5:
5                  blockchain = records['blockchain_id']
6
7      return blockchain
```

Listing 3.6: The isFull function

In lines 3-5, the *isFull* functions goes through all the sub-blockchains of the NST, and checks if there is any sub-blockchain with less than five active TLDs. If such a blockchain is found, that means that the new TLD can be appended in it. If all sub-blockchains are full, the function will return a *None* value, meaning that there is no available sub-blockchain to host the new TLD.

If the function returns a *None* value, the contract will generate a new sub-blockchain and send the new transaction in it (lines 8-9). Otherwise, the contract will send the transaction to the available sub-blockchian that the *isFull* function returned.

3.9 The Blockchain

BlockDom is a network consists of multiple-smaller-autonomous sub-blockchains. Each of these blockchains is an order back-linked list of blocks that contains transactions. Blocks are linked to each other by referring to the previous block of the chain. Each block inside the blockchain is identified by a unique hash (block header), which is generated by using the SHA256 hashing algorithm. Each block refers to its parent (previous block) through the *prevHash*. All the linked blocks create a chain going back all the way to the *Genesis Block*, which is the initial block of each blockchain.

3.9.1 Structure of a Block

A block is a data structure that stores transactions. While most popular blockchains like Bitcoin and Ethereum store multiple transactions within a block, BlockDom can store only one transaction per block. This is because blocks in BlockDom have to be accessed fast (in terms of performance) by the **Domain Tracker** contract in order to spot the requested domains. The algorithm will first go through the NST records and finally through the sub-blockchain blocks. By using such a technique, we manage to perform a domain searching with O(n^2) complexity as shown in listing 3.7.

```
for j in records:
    for i in blocks:
        if domain[i] == requested_domain:
            domain_found
```

Listing 3.7: Searching a block with one transaction

Instead, if we would store multiple transactions within a block, the searching algorithm would take O(n^3) as shown in listing 3.8. The algorithm would first go through the NST records, then through the sub-blockchain blocks, and finally through the transactions of the block.

```
for k in records:
    for i in blocks:
        for j in transactions:
            if domain[i][j] == requested_domain:
                domain_found
```

Listing 3.8: Searching a block with multiple transactions

Table 3.4 show the structure of a block. Each block has a maximum size of 200 bytes and consists of 5 components as shown below:

Field	Description	Size
Header	Unique block identifier	64 bytes
Previous Hash	Unique previous block identifier	64 bytes
Timestamp	Exact date and time of the block generation	25 bytes
Domain	Domain name of the transaction	40 bytes
IP	Corresponded domain's IP of the transaction	15 bytes

TABLE 3.4: Block structure

The header is the unique block identifier, meaning that each block has a unique header. The header is a 64 bytes hash which results by the hashing of the *domain*, the *IP* and the *prevHash* by using the SHA256 hashing algorithm.

$$Header = Domain + IP + Previous\_Hash$$

3.9.2 Genesis Block

The first block in each sub-blockahin is called Genesis Block. Whenever a new TLD is about to be registered in a sub-blockchain, the **Register Domain** contract checks if there is any available sub-blockchain in order to host the new TLD. If the contract is unable to locate an available sub-blockchain, a new sub-blockchain is generated.

As we discussed earlier, the header of each block is the result of hashing the *domain*, the *IP* and the *prevHash*. The Genesis Block is the first block to be appended in the blockchain, and therefore there is no parent block in which the Genesis Block refers to. Thus, the Genesis's Block *prevHash* is a simple string "Genesis Block".

$$Genesis\_Header = Domain + IP + "Genesis \quad Block"$$

Listing 3.9 shows the Genesis Block of a sub-blockchain.

```
{
  "header": "30bc46b8c96616ff01deb15a07b4399d2aa2fb2b35bf7f83341054c2413cb5d5",
  "prevHash": "Genesis Block",
  "timestamp": "Fri Jul 30 12:35:51 2021",
  "domain": "gov.all",
  "IP": "99.32.41.70"
}
```

LISTING 3.9: Genesis block

As we can see, the *prevHash* is just a string ("Genesis Block") and the *domain* is the "gov.all". Thus, if we hash the *domain*, the *IP* and the *prevHash* by using the SHA256 hashing algorithm we will produce the header of this block.

3.9.3 Generating New Sub-Blockchains

As mentioned earlier, the initial network will contain just one hard-coded sub-blockchain. While users register more and more domains in the network, new sub-blockchain will be required in order to host them. A new sub-blockchain is generated whenever all the existing sub-blockchains are unavailable, meaning that they all host five different TLDs. Consider a user wants to register a new domain with a TLD that does not exist in the network. That means that the new TLD has to be registered in a blockchain and stored in it. The network consists of the following two sub-blockchains.

```
"NST": [
    {
      "blockchain_id": "639ecfda46dc8f4f9795d892d16
                        1dd521d7f8cc47bcbf60f6942e6d9c7f949a0",
      "active_users": 57,
      "tld": ["com", "org", "net", "gr", "uk"],
      "active_tld": 5
    },
    {
      "blockchain_id": "c052b93579a91556e1158f07d0f91e44
                        0c5422a7cfd542e9814b827918c55e14",
      "active_users": 19,
      "tld": ["gov", "edu", "mil", "bio", "eco"],
      "active_tld": 5
    }
]
```

Listing 3.10: Full sub-blockchains

As we can see in listing 3.10 both sub-blockchains are full because they both host five different TLDs. That means that the new TLD cannot be registered in neither of them. Thus, a new sub-blockchain should be generated. The Register Domain contract uses the **newBlockchain** function as shown in listing 3.11 whenever a new sub-blockchain is needed to be generated.

```
function newBlockchain():
    blockchain_id = random.getrandbits(256)

    open(NST):
        new_blockchain = {
            "blockchain_id": blockchain_id,
            "active_users": 1,
            "tld": [tld],
            "active_tld": 1
        }

        records.append(new_blockchain)

    open(blockchain_id):
        append_genesis_block()
```

Listing 3.11: The New Blockchain contract

In line 2 the function generates an ID for the new sub-blockchain. The ID is a unique random SHA256 hash. After that in lines 4-12 the function updates the NST by appending the new sub-blockchain in it. The new record is an object similar to those that already exist in the NST. The *blockchain_id* field is the unique ID of the new sub-blockchain. The *active_users* field is set to 1 as the Seeder node tha will be generated by the contact will be the only node in the network. The new TLD is appended to the *tld* array and the *active_tld* field becomes 1 as only one TLD exists in this new sub-blockchain.

3.10 Consensus

When a new transaction successfully validated and propagated to the whole network, it is then ready to be appended into a block and linked in the blockchain. The transaction will be transferred from the network layer to the consensus layer.

Traditional blockchains like Bitcoin and Ethereum use consensus algorithms like Proof-of-Work and Proof-of-Stake accordingly in order to generate new blocks and append them to the blockchain (22). The nodes that perform such processes, called miners in the case of PoW and stakers in the case of PoS. Miners and Stakers offer to the network their processing power or part of their assets, and they get to exchange an amount of cryptocurrencies as a reward for their contribution. However, as we mentioned earlier, BlockDom is a non-commercial public blockchain that does not issue or maintain a cryptocurrency. As a result, the consensus algorithm that is used is not profitable for those nodes, but at the same time does not require from them any significant processing resources or anything valuable.

In BlockDom, the process of generating a new block and appending it in the blockchain is called **construction** and the node who is responsible for it, is called **constructor**. Construction is essentially a consensus protocol through which a node constructs a new block with a transaction and appends it in the sub-blockchain. BlockDom's consensus algorithm follows a similar architecture like Hyperledger's consensus, where nodes provide a guaranteed ordering of transactions and validate the block of them (23). The main difference is that BlockDom does not divide the nodes into categories like Hyperledger does. Instead, all BlockDom nodes perform the same consensus functionalities.

Consensus algorithm is mainly used for solving the Byzantine Generals Problem occurred in distributed systems. The data can be delivered between different nodes through peer-to-peer communications. However, some nodes may be maliciously attacked, which will lead to the changes of communication contents. Normal nodes need to distinguish the information that has been tampered and obtain the consistent results with other normal nodes. This also needs the design of the corresponding consensus algorithm (24).

3.10.1 Choosing Constructors

When a new transaction is validated, it has to proceed from the network layer to the consensus layer. The consensus layer contains the construction process through which the transaction will be included in a block and then appended in the sub-blockchian. When the transaction reaches the consensus layer, the algorithm has to decide which node will construct the new block. According to which sub-blockchain has been selected to host the new transactions, there are three different cases regarding the selection of the constructor.

- **case 1:** If the transaction is going to be hosted in the same sub-blockchain with its originator, the originator of the transaction will be responsible for the construction process
- **case 2:** If the transaction is going to be hosted in an external existing sub-blockchain (a different sub-blockchain than the one the originator operates), the algorithm will choose randomly a node from this external sub-blockchain and assign it with the construction process
- **case 3:** If the transaction requires the generation of a new sub-blockchain, the algorithm will move the originator of the transaction from the current sub-blockchain to the new sub-blockchain and assign it with the construction process

```
 1 function constructorPicker(id):
 2     blockchain_assigned = client_role()
 3
 4     if id == blockchain_assigned:
 5         # case 1
 6     else:
 7         open(NST):
 8             for i in records:
 9                 if id == records['blockchain_id']:
10                     # case 2
11                     another = True
12                     break
13                 else:
14                     another = False
15             if another == False:
16                 # case 3
```

Listing 3.12: The constructor picker function

The *constructorPicker* function in listing 3.12 takes one parameter, which is the ID of the blockchain that the new transaction will be hosted. In line 2 the *clientRole* function is invoked in order to identify the ID of the blockchain that the node operates. If the blockchain ID that the transaction will be hosted is the same with the blockchain ID that the node operates, that means that we are in the first case where the transaction will be hosted at the same network with the node (lines 4-5). Otherwise, the function will access the NST and check if the blockchain ID exists in it. If the blockchain is found, that means that we are in the second where the transaction will be hosted in an external sub-blockchain (lines 9-12). The transaction will be forwarded to the external sub-blockchain and a node will be chosen randomly for the construction process. If the blockchain is not found, that means that we are in the third case, where a new sub-blockchain will be generated (lines15-16). The node will be moved in this blockchain and construct the block with its own transaction.

3.10.2 Block Construction

The final stage of the transaction lifecycle in the consensus layer is the block construction. The block will then be ready to be appended in the blockchain and reached by other nodes. When the constructor has been determined successfully, the **construction** function is responsible for creating a new block and appending the transaction in it as shown in listing 3.13.

```
 1 function construction(id, reg_domain, new_tld):
 2     open(blockchain_path):
 3         block_data = reg_domain + ip + blocks[-1][’header’]
 4
 5         new_block = {
 6             "header": sha256(block_data),
 7             "prevHash": blocks[-1][’header’],
 8             "timestamp": datetime.now(),
 9             "domain": reg_domain,
10             "IP": getIP(reg_domain)
11         }
12
13         blocks.append(new_block)
14
15     if new_tld == True:
16         open(NST):
17             for i in records:
18                 if records[’blockchain_id’] == id:
19                     records[’tld’].append(tld)
20                     records[’active_tld’] += 1
```

Listing 3.13: The construction function

The function takes three parameters. The ID of the blockchain that the transaction will be hosted, the requested domain, and a boolean variable indicates whether the TLD of the requested domain already exists or not. At first, the function will access the blockchain in which the new block will be appended (line 2). As we mentioned in another section, each block consists of five fields. In lines 5-11, the function constructs the new block. The *header* of the block (line 6) results from the hashing of the domain name and the IP address (transaction) plus the header of the previous block (line 3). The *prevHash* field (line 7) is simply the header of the previous block (the last block in the blockchain). The *timestamp* field (line 8) indicates the exact date and time of the block creation. Lines 9-10 is the transaction that contains the requested domain name and its corresponding IP address. After the creation of the new block, the function is ready to append and link it in the sub-blockchain (line 13).

The last requirement is the NST update. The *new_tld* variable indicates whether the TLD of the requested domain already exists or not. If the variable has a True value, that means that the TLD does not exist. In that case, the new TLD will be appended either in an available existing sub-blockchain or in a new sub-blockchain. In both cases, the new TLD should be recorder in the NST. Lines 15-20 are responsible for updating the NST. In lines 17-19, the algorithm searches for the blockchain in which the new TLD will be appended. When it is found, it is appended to the TLDs array (line 19) and the *active_tld* variable is increased by one.

3.10.3 Genesis Block Construction

The Genesis Block is the first block of a blockchain. Whenever a new sub-blockchain is generated, the constructor node is actually responsible of constructing the Genesis Block. The Genesis Block is constructed in the same way all the other blocks are constructed. The **newBlockchain** function is responsible for constructing the Genesis Block and appending it to the sub-blockchain, as shown in listing 3.14.

```
 1 function newBlockchain(reg_domain):
 2     blockchain_id = random.getrandbits(256)
 3 
 4     open(NST):
 5         new_blockchain = {
 6             "blockchain_id": blockchain_id,
 7             "active_users": 1,
 8             "tld": [tld],
 9             "active_tld": 1
10         }
11 
12         records.append(new_blockchain)
13 
14     open(blockchain_id):
15         block_data = reg_domain + ip + 'Genesis Block'
16 
17         new_block = {
18             "header": sha256(block_data),
19             "prevHash": 'Genesis Block',
20             "timestamp": datetime.now(),
21             "domain": reg_domain,
22             "IP": getIP(reg_domain)
23         }
24 
25         blocks.append(new_block)
```

Listing 3.14: The Genesis block construction function

In line 2 the function generates a random ID for the new sub-blockchain. Then in lines 4-10 the new sub-blockchain is recorder in the NST. In line 14 the function gets access to the new sub-blockchain in order to originate and append the Genesis Block. The Genesis block is generated in lines 17-23. As long as Genesis Block is the first block in the blockchain, there is no previous block to refer to. Thus, the header of the Genesis block results from the hashing of the domain name plus the IP (transaction) plus a fixed string 'Genesis Block' (line 15). This fixed string represents the *prevHash* field. The remaining fields are the same as the fields of a regular block.

3.10.4 Assigning IP Addresses

At that stage, we should mention that the assignment of an IP address to a domain name occurs during the construction procedure. When the node generates a new block, the *ip* field gets a value by invoking the **getIP** function as shown in listing 3.15.

```
1 function getIP(reg_domain):
2     try:
3         host_ip = socket.gethostbyname(reg_domain)
4     except:
5         host_ip = generateIp()
6 
7     return host_ip
```

LISTING 3.15: The getIP function

If the requested domain already exists, and it is registered to a tradition DNS server, the function will make a request in the server in order to get its IP (lines 2-3). This process is essentially a transition of a traditional DNS record to the BlockDom network. If the requested domain cannot be found, that means that it does not exist anywhere and the function will generate a new IP address for it (lines 4-5).

Figure 3.5 shows how *getIP* function operates:

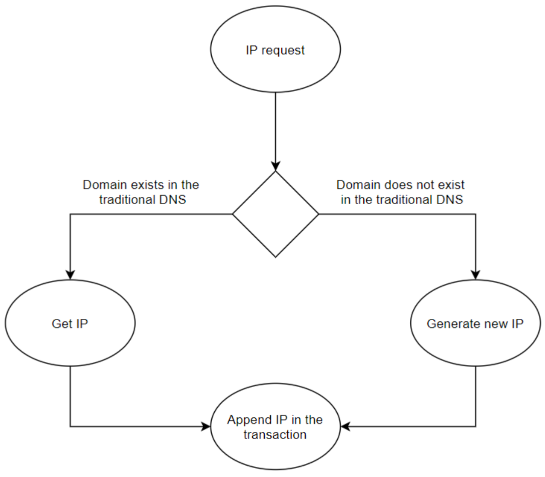
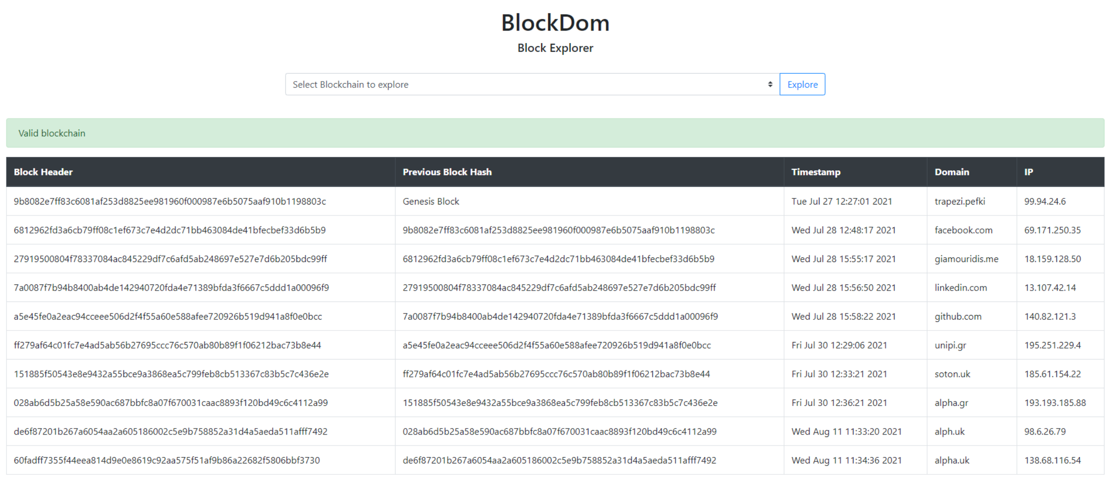

IP request
Domain exists in the traditional DNS
Domain does not exist in the traditional DNS
Get IP
Generate new IP
Append IP in the transaction

FIGURE 3.5: getIP function flow

3.10.5 Block Explorer

Once the block has been constructed is now part of the sub-blockchain. All BlockDom nodes regardless of which sub-blockchain they are on can access the **Block Explorer** and see all the transactions that have been made so far. Remember that blockchain networks are open and visible to anyone.


BlockDom

Block Explorer

Valid blockchain

Block Header	Previous Block Hash	Timestamp	Domain	IP
9b8082e7ff83c6081af253d8825ee981960f000987e6b5075aaf910b1198803c	Genesis Block	Tue Jul 27 12:27:01 2021	trapezi.pefki	99.94.24.6
6812962fd3a6cb79ff08c1ef673c7e4d2dc71bb463084de41bfecbef33d6b5b9	9b8082e7ff83c6081af253d8825ee981960f000987e6b5075aaf910b1198803c	Wed Jul 28 12:48:17 2021	facebook.com	69.171.250.35
27919500804f78337084ac845229df7c6afd5ab248697e527e7d6b205bdc99ff	6812962fd3a6cb79ff08c1ef673c7e4d2dc71bb463084de41bfecbef33d6b5b9	Wed Jul 28 15:55:17 2021	giamouridis.me	18.159.128.50
7a0087f7b94b8400ab4de142940720fda4e71389bfda3f6667c5ddd1a00096f9	27919500804f78337084ac845229df7c6afd5ab248697e527e7d6b205bdc99ff	Wed Jul 28 15:56:50 2021	linkedin.com	13.107.42.14
a5e45fe0a2eac94cceee506d2f4f55a60e588afee720926b519d941a8f0e0bcc	7a0087f7b94b8400ab4de142940720fda4e71389bfda3f6667c5ddd1a00096f9	Wed Jul 28 15:58:22 2021	github.com	140.82.121.3
ff279af64c01fc7e4ad5ab56b27695ccc76c570ab80b89f1f06212bac73b8e44	a5e45fe0a2eac94cceee506d2f4f55a60e588afee720926b519d941a8f0e0bcc	Fri Jul 30 12:29:06 2021	unipi.gr	195.251.229.4
151885f50543e8e9432a55bce9a3868ea5c799feb8cb513367c83b5c7c436e2e	ff279af64c01fc7e4ad5ab56b27695ccc76c570ab80b89f1f06212bac73b8e44	Fri Jul 30 12:33:21 2021	soton.uk	185.61.154.22
028ab6d5b25a58e590ac687bbfc8a07f670031caac8893f120bd49c6c4112a99	151885f50543e8e9432a55bce9a3868ea5c799feb8cb513367c83b5c7c436e2e	Fri Jul 30 12:36:21 2021	alpha.gr	193.193.185.88
de6f87201b267a6054aa2a605186002c5e9b758852a31d4a5aeda511afff7492	028ab6d5b25a58e590ac687bbfc8a07f670031caac8893f120bd49c6c4112a99	Wed Aug 11 11:33:20 2021	alph.uk	98.6.26.79
60fadff7355f44eea814d9e0e8619c92aa575f51af9b86a22682f5806bbf3730	de6f87201b267a6054aa2a605186002c5e9b758852a31d4a5aeda511afff7492	Wed Aug 11 11:34:36 2021	alpha.uk	138.68.116.54

FIGURE 3.6: Block explorer

Block Explorer is a BlockDom's functionality which allows users to explore the transactions in the sub-blockchains. Every active sub-blockchain is listed in the Block Explorer and all the transactions can be found there. Figure 3.6 shows all the transactions that are stored in a sub-blockchain.

Chapter 4

Testing and Evaluation

As we mentioned in our initial objectives, our proposal of a blockchain based DNS focuses mainly on solving security and performance gaps of the traditional DNS protocol. Following the presentation of the implementation of the protocol, it is very important to support what we have presented so far through a series of experiments. In this section we will focus on the two properties (security and performance) which were mentioned in the project aims, and we will conduct a series of experiments to show that the protocol we propose corresponds to all these objectives.

4.1 Performance Evaluation

The efficiency of the protocol in terms of complexity and performance is one of the main challenges of this research. The main functionalities of the BlockDom protocol are requesting and registering domains in the network. Thus, we will be able to characterize the protocol as efficient if and only if the user's requests are served in a time efficient period. In order to be able to calculate whether our protocol is behaving efficiently in terms of performance, we need to perform a series of real-time testing and draw a conclusion. For these experiments, we will use the searching algorithm and measure the average throughput and latency needed for the network to response to user's requests. We will also perform the same measurements with the registration algorithm and see how much time does the network require in order to register a new domain.

4.1.1 Throughput and Latency

In this performance series of testing and evaluation, we are going to examine two different BlockDom processes. The first one is the requesting of a specific domain, while the second one is the registration of a new domain in a sub-blockchain (origination of the transaction — block construction). These are the two main processes of the protocol, and it is very important to have a view of some crucial network traffic measurements like the throughput they require and the latency of the packet transmission.

Both testing will be performed in a real-time environment by running the BlockDom network in a local network.

4.1.1.1 Requesting Domains

Requesting a domain in the blockchain is a process that require O(n^2) time in terms of complexity, as we can see in listing 4.1. Each NST record (sub-blockchain) contains an array with all the domains that are stored in it. Thus, the Domain Tracker contract should go through the entire NST and then through the TLD array in order to find the requested domain.

```
for i in records:
    for j in domains:
        if domain[i][j] == requested_domain:
            domain_found
```

Listing 4.1: Domain Tracker contract complexity

Figure 4.1 shows the throughput used by the Domain Tracker contract when a domain name is requested.

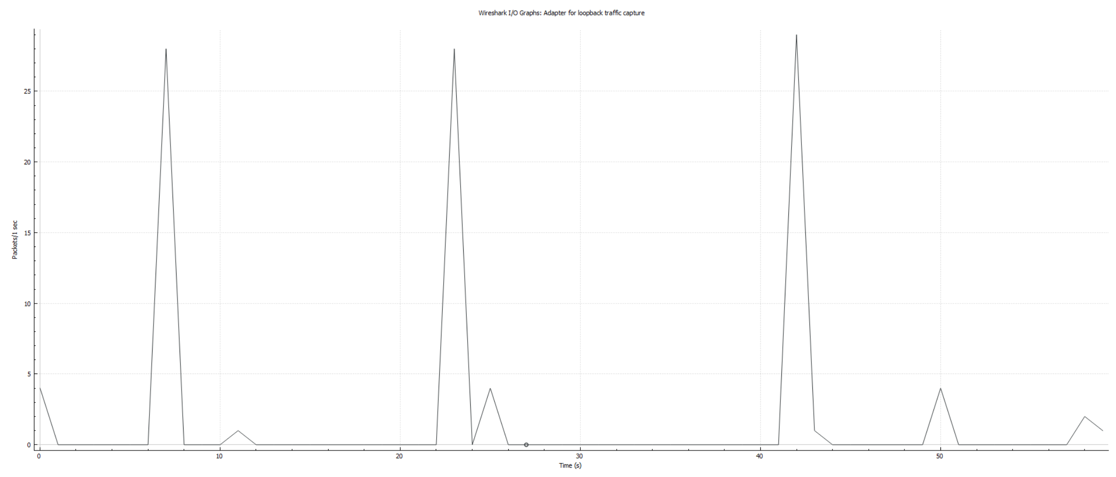
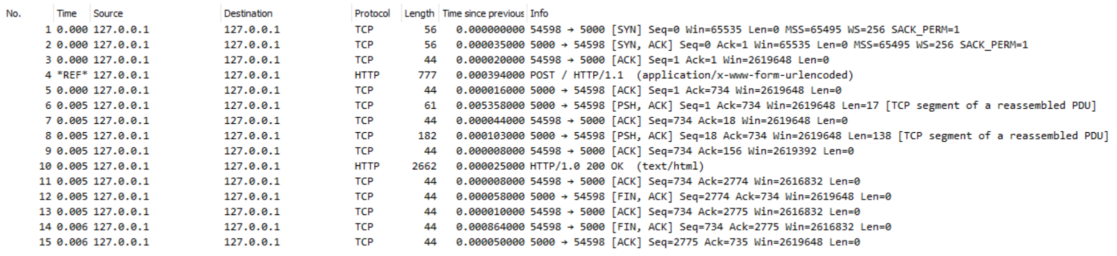


Figure 4.1: Domain Tracker throughput

In this graph, we can see that the Domain Tracker contract has been invoked three time, meaning that three domain requests have been made. The throughput required by the Domain Tracker contract in order to perform each searching process is

$$T = 28 Packets/second$$

The latency of the requesting process is shown in figure 4.2:



No.	Time	Source	Destination	Protocol	Length	Time since previous	Info
1	0.000	127.0.0.1	127.0.0.1	TCP	56	0.000000000	54598 → 5000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
2	0.000	127.0.0.1	127.0.0.1	TCP	56	0.000035000	5000 → 54598 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
3	0.000	127.0.0.1	127.0.0.1	TCP	44	0.000020000	54598 → 5000 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
4	*REF*	127.0.0.1	127.0.0.1	HTTP	777	0.000394000	POST / HTTP/1.1 (application/x-www-form-urlencoded)
5	0.000	127.0.0.1	127.0.0.1	TCP	44	0.000016000	5000 → 54598 [ACK] Seq=1 Ack=734 Win=2619648 Len=0
6	0.005	127.0.0.1	127.0.0.1	TCP	61	0.005358000	5000 → 54598 [PSH, ACK] Seq=1 Ack=734 Win=2619648 Len=17 [TCP segment of a reassembled PDU]
7	0.005	127.0.0.1	127.0.0.1	TCP	44	0.000044000	54598 → 5000 [ACK] Seq=734 Ack=18 Win=2619648 Len=0
8	0.005	127.0.0.1	127.0.0.1	TCP	182	0.000103000	5000 → 54598 [PSH, ACK] Seq=18 Ack=734 Win=2619648 Len=138 [TCP segment of a reassembled PDU]
9	0.005	127.0.0.1	127.0.0.1	TCP	44	0.000008000	54598 → 5000 [ACK] Seq=734 Ack=156 Win=2619392 Len=0
10	0.005	127.0.0.1	127.0.0.1	HTTP	2662	0.000025000	HTTP/1.0 200 OK (text/html)
11	0.005	127.0.0.1	127.0.0.1	TCP	44	0.000008000	54598 → 5000 [ACK] Seq=734 Ack=2774 Win=2616832 Len=0
12	0.005	127.0.0.1	127.0.0.1	TCP	44	0.000058000	5000 → 54598 [FIN, ACK] Seq=2774 Ack=734 Win=2619648 Len=0
13	0.005	127.0.0.1	127.0.0.1	TCP	44	0.000010000	54598 → 5000 [ACK] Seq=734 Ack=2775 Win=2616832 Len=0
14	0.006	127.0.0.1	127.0.0.1	TCP	44	0.000864000	54598 → 5000 [FIN, ACK] Seq=734 Ack=2775 Win=2616832 Len=0
15	0.006	127.0.0.1	127.0.0.1	TCP	44	0.000050000	5000 → 54598 [ACK] Seq=2775 Ack=735 Win=2619648 Len=0

Figure 4.2: Requesting domain latency

The 4th packet is the POST request to the sub-blockchain that contains the requested domain. When the sub-blockchain response is ready to get back to the client, a TCP request will be made and the response will be forwarded to the originator node. Te node will finally receive an HTTP successful message (10th packet) which means that the request has been served successfully by the sub-blockchain. We can see that the round-trip time (RTT) of this packet took 0.005 ms. and therefore we say that the latency of the domain request is 0.005 ms.

4.1.1.2 Creating Transactions and Blocks

The Register contract is responsible for the creation of a new transaction. This process is also executed in O(n^2) time, as the contract first goes through the NST (like the Domain Tracker contract) in order to make sure that the requested domain does not already exist. After that check, the contract will either call the *mining* function in order to mine (construct) and create a new block for the new transaction or the *newBlockahin* function in order to create a new sub-blockchain.

Figure 4.3 show the throughput that is used when a domain is registered in an already existing sub-blockchain.

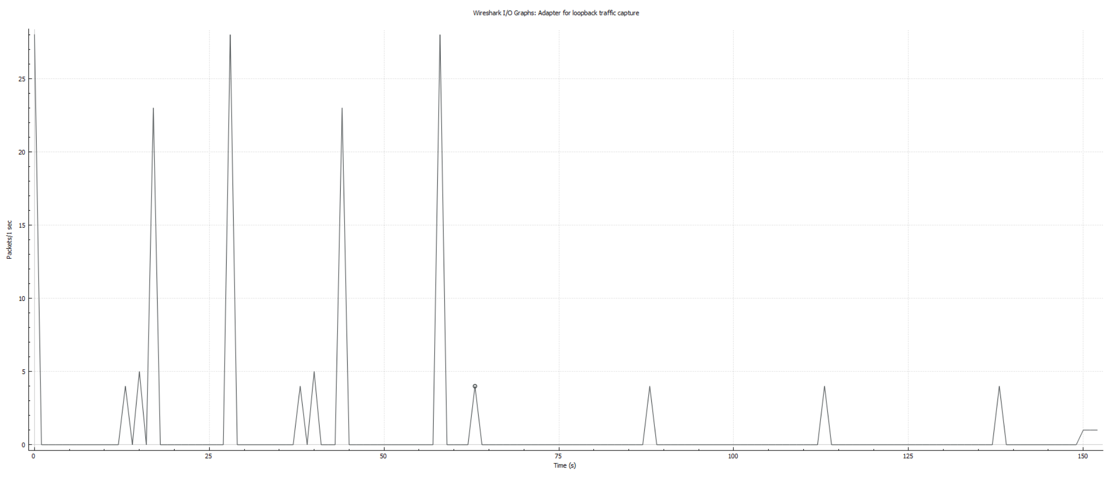
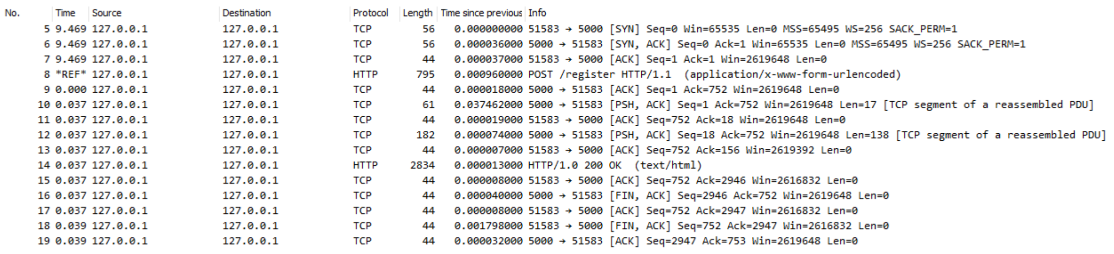


Figure 4.3: Registering domain in existing sub-blockchain throughput

The throughput in that case depends on the existence of the requested domain in the traditional DNS servers. If the domain name is already registered in it, the algorithm requires more resources because the *getIP* function should make a request to the DNS server in order to obtain its information. Otherwise, the app will generate a new IP. The average throughput for the creation of a new transaction and a block is

$$T = 26.5 Packets/second$$

Figure 4.4 shows the time needed for the application to append the new transaction in a sub-blockchain.



```
No.    Time  Source          Destination     Protocol Length Time since previous Info
     5 9.469 127.0.0.1       127.0.0.1       TCP        56   0.000000000 51583 → 5000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
     6 9.469 127.0.0.1       127.0.0.1       TCP        56   0.000036000 5000 → 51583 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
     7 9.469 127.0.0.1       127.0.0.1       TCP        44   0.000037000 51583 → 5000 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
     8 *REF* 127.0.0.1       127.0.0.1       HTTP      795   0.000960000 POST /register HTTP/1.1  (application/x-www-form-urlencoded)
     9 0.000 127.0.0.1       127.0.0.1       TCP        44   0.000018000 5000 → 51583 [ACK] Seq=1 Ack=752 Win=2619648 Len=0
    10 0.037 127.0.0.1       127.0.0.1       TCP        61   0.037462000 5000 → 51583 [PSH, ACK] Seq=1 Ack=752 Win=2619648 Len=17 [TCP segment of a reassembled PDU]
    11 0.037 127.0.0.1       127.0.0.1       TCP        44   0.000019000 51583 → 5000 [ACK] Seq=752 Ack=18 Win=2619648 Len=0
    12 0.037 127.0.0.1       127.0.0.1       TCP       182   0.000074000 5000 → 51583 [PSH, ACK] Seq=18 Ack=752 Win=2619648 Len=138 [TCP segment of a reassembled PDU]
    13 0.037 127.0.0.1       127.0.0.1       TCP        44   0.000007000 51583 → 5000 [ACK] Seq=752 Ack=156 Win=2619392 Len=0
    14 0.037 127.0.0.1       127.0.0.1       HTTP     2834   0.000013000 HTTP/1.0 200 OK  (text/html)
    15 0.037 127.0.0.1       127.0.0.1       TCP        44   0.000008000 51583 → 5000 [ACK] Seq=752 Ack=2946 Win=2616832 Len=0
    16 0.037 127.0.0.1       127.0.0.1       TCP        44   0.000040000 5000 → 51583 [FIN, ACK] Seq=2946 Ack=752 Win=2619648 Len=0
    17 0.037 127.0.0.1       127.0.0.1       TCP        44   0.000008000 51583 → 5000 [ACK] Seq=752 Ack=2947 Win=2616832 Len=0
    18 0.039 127.0.0.1       127.0.0.1       TCP        44   0.001798000 51583 → 5000 [FIN, ACK] Seq=752 Ack=2947 Win=2616832 Len=0
    19 0.039 127.0.0.1       127.0.0.1       TCP        44   0.000032000 5000 → 51583 [ACK] Seq=2947 Ack=753 Win=2619648 Len=0
```

FIGURE 4.4: Registering domain in existing sub-blockchain latency

Again, the 8th packet is the initial POST request. The server response is shown in the 14th packet. The round-trip time (RTT) for this packet is 0.037 ms. and thus we say that the latency of the transaction creation is 0.037 ms.

4.2 Security Evaluation

Security is one of the key features of a successful implementation. As long as DNS protocol is vulnerable to various attacks, user's security is remaining weak. BlockDom handles all the data in such a way that it is able to prevent and mitigate popular DNS attacks that may cause network confusion and threaten the users. In order to be able to claim that our protocol meets the security criteria, we need to make sure that possible attacks that may occur are not affecting the CIA triad (Confidentiality, Integrity, Availability). To achieve such a goal, we are going to conduct a simulation of popular DNS attacks to the network, and through security analysis we will prove that our protocol meets all the security requirements.

4.2.1 CIA Triad in BlockDom

4.2.1.1 Confidentiality

Traditional blockchain users maintain a wallet in which their private keys are stored in order to sign and creating new transactions. Users can access and manage their assets only by using their private keys, meaning that private keys are significantly sensitive data in terms of security and must remain confidential.

BlockDom is an open, non-commercial blockchain network. We mention at the beginning that BlockDom does not issue or maintain any cryptocurrency (valuable asset). Blocks are constructed by the users that make the transactions without having any benefit, but at the same time without contributing with their processing resources or their assets. As long as users do not own any asset within the network, there is no need for maintaining a wallet with keys. Furthermore, transactions are not considered as private data as they are listed publicly and can be viewed by everyone like in all blockchain networks. Thus, confidentiality in BlockDom is ensured as there are no private data that require to remain secure and confidential.

4.2.1.2 Integrity

Blockchain is an append-only data structure, meaning that data can be only appended in it but neither modified nor deleted. The Blockchain data structure is an ordered back-linked list of blocks of transactions. Blocks are linked back, each referring to the previous block in the chain. Each block within the Blockchain is identified by a hash, generated using a cryptographic hash algorithm, on the header of the block. Each block also refers to a previous block, known as the parent block, through the previous block hash field in the block header. In other words, each block contains the hash of its parent inside its own header.

The ”previous block hash” field is inside the block header and thereby affects the current block’s hash. The child’s own identity changes if the parent’s identity changes. When the parent is modified in any way, the parent’s hash changes. The parent’s changed hash necessitates a change in the previous block hash pointer of the child. This, in turn, causes the child’s hash to change, which requires a change in the pointer of the grandchild, which in turn changes the grandchild and so on. This cascade effect ensures that once a block has many generations following it, it cannot be changed without forcing a recalculation of all subsequent blocks.

In BlockDom, transactions that include the domain-IP pair can be considered as sensitive data, and therefore we have to ensure their integrity in order to state that the protocol meets the security requirements. As we have seen in the previous chapter, the header of a blocks results from the hashing of the domain name plus the IP address plus the previous (parent) block header. As long as the parent header is included in the header of the next (child) block, any change in the data of the transaction will affect the header of the parent and therefore any block after him. The breach of integrity will be immediately detected by the network mechanisms and users will be informed. Attacks like DNS spoofing are impossible to affect the network, as we are going to see later.

4.2.1.3 Availability

One of the main security features of the blockchain technology is the assurance of network availability in case of an overload attack like DoS or DDoS. Blockchain networks are decentralized entities as they are designed and implemented based on the P2P architecture. Therefore, there is no central point of failure (central server or database), but instead the availability of the network is solely depending on each network peer. Thus, it is extremely difficult for an attacker to set all nodes out of order through a DoS attack and affect network availability.

Due to its architecture, BlockDom adds an extra layer of security in terms of availability. As we have seen so far, BlockDom is a collection of autonomous sub-blockchains, each of it operates on its own. Therefore, there is a network fragmentation, meaning that the network is split into smaller chunks. As long as each sub-blockchain is fully autonomous, even an attacker manages to set a sub-blockchain out of order, the rest sub-blockchains will not be affected, and the rest network will still remain available. Attacks such as DNS flooding (as we will see below in the attack's simulation) can be prevented because of the BlockDom's architecture, and ensure permanent network availability.

Once we have laid the foundations for the security of our protocol, we will continue with a series of more specialized experiments to prove the security quality of the protocol under real circumstances. At this point we have selected three of the most popular attacks we encounter in the DNS protocol as shown in 4.5 (25).

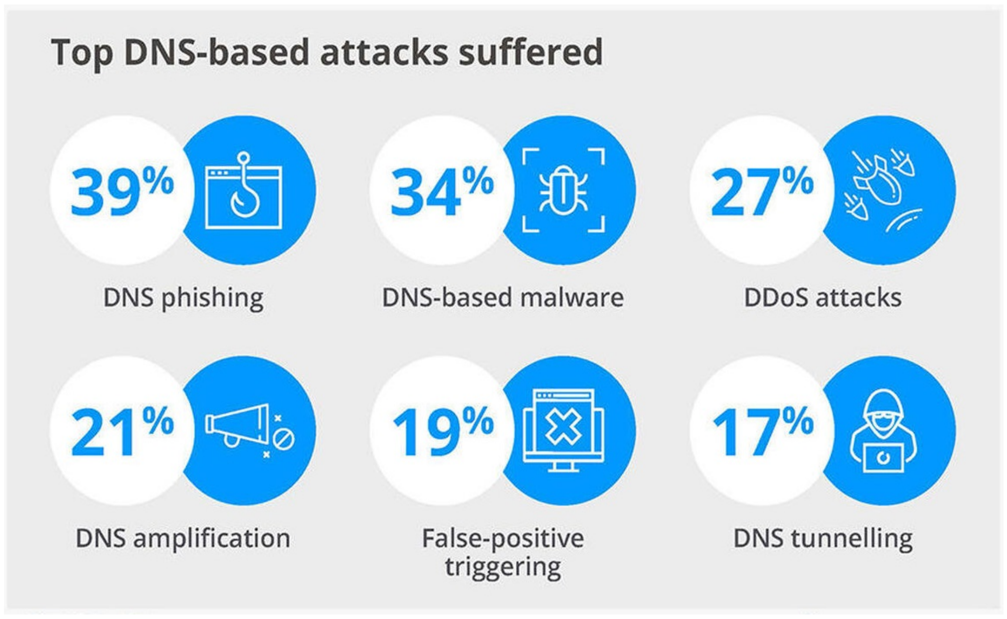


FIGURE 4.5: DNS attacks popularity

4.2.2 DNS Flood Attack

When a DNS flood attack occurs, the DNS server receives a large number of requests for service. As a result, the server is no longer available for users, as it is not possible to send the corresponding number of responses (26). In general, flooding attacks target network availability by denying service to clients while taking advantage of the skewed distribution of functionality between clients and servers. In the case of DNS, the fact that the nameservers for a zone are completely responsible for serving the zone's records and, in turn, for the operation of any sub-zones implies that their availability is critical and makes them an attractive target for flooding attacks.

Due to the fact that Blockchain networks are not hosted on a centralized server or computer, but instead each node is connected to the network from different locations, they do not have any downtime issues. Furthermore, as we mentioned before, BlockDom adds an extra layer of security as sub-blockchains are split and operate on their own. Thus, if a flood attack successfully exploits a sub-blockchain, the rest of them will not be affected.

4.2.3 DNS Spoofing

DNS spoofing is a type of attack in which malicious data is entered into the DNS server. This attack is known as DNS cache poisoning. However, this attack is different because it targets the DNS server and not the victim's computer (27). BlockDom does not have access to user's cache memory in order to store data and therefore all data (the blockchain) are stored in the main user's storage unit. In that case, a DNS spoofing attack would aim to alter data in the blockchain which is stored inside the user's storage, and broadcast the cascaded blockchain to the rest network.

Consider the spreading of the following script in listing 4.2 and infection of an active user.

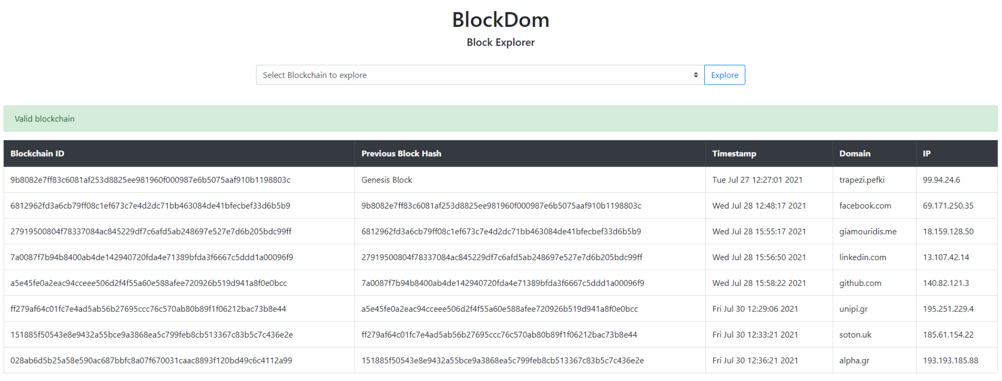
```
open(blockchain):
    for i in blocks:
        blocks['IP'] = '138.20.12.1'
```

Listing 4.2: DNS spoofing script

In just three lines, the malicious script gets access to a specific sub-blockchain and changes all the IP addresses of the records to redirect to a specific IP address, perhaps a fraudulent website. Now, when users request domains that are stored in this sub-blockchain, they will be redirected to the malicious website and risk their safety. Traditional DNS mechanisms are unable to detect such an alteration, and therefore users are often redirected to malicious websites instead of being redirected to the website they have requested.

On the other hand, a blockchain network like BlockDom is able to identify such an attack and prevent users for being redirected to fraudulent websites. As we mentioned earlier, each block in the blockchain is connected with its previous block through their headers. The slightest change in the block's data will affect the header of the block and therefore the rest chain.

Consider the sub-blockchain in figure 4.6 that contains eight blocks (eight transactions).



Valid blockchain

Blockchain ID	Previous Block Hash	Timestamp	Domain	IP
9b8082e7ff83c6081af253d8825ee981960f000987e6b5075aaf910b1198803c	Genesis Block	Tue Jul 27 12:27:01 2021	trapezi.pefki	99.94.24.6
6812962fd3a6cb79ff08c1ef673c7e4d2dc71bb463084de41bfecbef33d6b5b9	9b8082e7ff83c6081af253d8825ee981960f000987e6b5075aaf910b1198803c	Wed Jul 28 12:48:17 2021	facebook.com	69.171.250.35
27919500804f78337084ac845229df7c6afd5ab248697e527e7d6b205bdc99ff	6812962fd3a6cb79ff08c1ef673c7e4d2dc71bb463084de41bfecbef33d6b5b9	Wed Jul 28 15:55:17 2021	giamouridis.me	18.159.128.50
7a0087f7b94b8400ab4de142940720fda4e71389bfda3f6667c5ddd1a00096f9	27919500804f78337084ac845229df7c6afd5ab248697e527e7d6b205bdc99ff	Wed Jul 28 15:56:50 2021	linkedin.com	13.107.42.14
a5e45fe0a2eac94cceee506d2f4f55a60e588afee720926b519d941a8f0e0bcc	7a0087f7b94b8400ab4de142940720fda4e71389bfda3f6667c5ddd1a00096f9	Wed Jul 28 15:58:22 2021	github.com	140.82.121.3
ff279af64c01fc7e4ad5ab56b27695ccc76c570ab80b89f1f06212bac73b8e44	a5e45fe0a2eac94cceee506d2f4f55a60e588afee720926b519d941a8f0e0bcc	Fri Jul 30 12:29:06 2021	unipi.gr	195.251.229.4
151885f50543e8e9432a55bce9a3868ea5c799feb8cb513367c83b5c7c436e2e	ff279af64c01fc7e4ad5ab56b27695ccc76c570ab80b89f1f06212bac73b8e44	Fri Jul 30 12:33:21 2021	soton.uk	185.61.154.22
028ab6d5b25a58e590ac687bbfc8a07f670031caac8893f120bd49c6c4112a99	151885f50543e8e9432a55bce9a3868ea5c799feb8cb513367c83b5c7c436e2e	Fri Jul 30 12:36:21 2021	alpha.gr	193.193.185.88

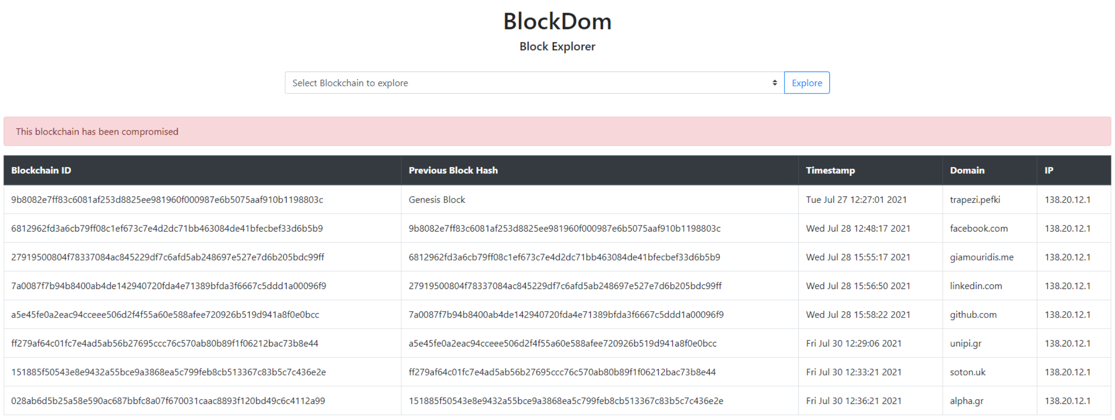
Figure 4.6: Valid sub-blockchain

BlockDom network uses a validation algorithm that checks a sub-blockchain whenever a change occurs and confirms whether it is valid or invalid.

```python
if len(blocks) > 1:
    check = genesis['domain'] + genesis['IP']
    header = sha256(check)
    invalidGen = False

    if genesis['header'] != header:
        # invalid blockchain
        invalidGen = True

    for i in range(1, blocks):
        if invalidGen == True:
            break

        header = sha256(blocks['domain'] + blocks['IP']
                    + blocks['previous_hash'])

        if blocks['previous_hash'] != previous['header'] or
            blocks['header'] != header:
            # invalid blockchain
else:
    current = blocks[0]
    header = sha256(current['domain'] + current['IP'])
    if current['header'] != header:
        # invalid blockchain
```

LISTING 4.3: Blockchain validator

The algorithm in listing 4.3 goes through all the blocks and checks if there is any alteration in the block headers. If any change is found, that means that there was an attempt of modifying the block and the rest blockchain has been affected. We will run the script of the listing 4.2 in order to change all the IP addresses of the records and redirect users to a specific IP.



This blockchain has been compromised

Blockchain ID	Previous Block Hash	Timestamp	Domain	IP
9b8082e7ff83c6081af253d8825ee981960f000987e6b5075aaf910b1198803c	Genesis Block	Tue Jul 27 12:27:01 2021	trapezi.pefki	138.20.12.1
6812962fd3a6cb79ff08c1ef673c7e4d2dc71bb463084de41bfecbef33d6b5b9	9b8082e7ff83c6081af253d8825ee981960f000987e6b5075aaf910b1198803c	Wed Jul 28 12:48:17 2021	facebook.com	138.20.12.1
27919500804f78337084ac845229df7c6afd5ab248697e527e7d6b205bdc99ff	6812962fd3a6cb79ff08c1ef673c7e4d2dc71bb463084de41bfecbef33d6b5b9	Wed Jul 28 15:55:17 2021	giamouridis.me	138.20.12.1
7a0087f7b94b8400ab4de142940720fda4e71389bfda3f6667c5ddd1a00096f9	27919500804f78337084ac845229df7c6afd5ab248697e527e7d6b205bdc99ff	Wed Jul 28 15:56:50 2021	linkedin.com	138.20.12.1
a5e45fe0a2eac94cceee506d2f4f55a60e588afee720926b519d941a8f0e0bcc	7a0087f7b94b8400ab4de142940720fda4e71389bfda3f6667c5ddd1a00096f9	Wed Jul 28 15:58:22 2021	github.com	138.20.12.1
ff279af64c01fc7e4ad5ab56b27695ccc76c570ab80b89f1f06212bac73b8e44	a5e45fe0a2eac94cceee506d2f4f55a60e588afee720926b519d941a8f0e0bcc	Fri Jul 30 12:29:06 2021	unipi.gr	138.20.12.1
151885f50543e8e9432a55bce9a3868ea5c799feb8cb513367c83b5c7c436e2e	ff279af64c01fc7e4ad5ab56b27695ccc76c570ab80b89f1f06212bac73b8e44	Fri Jul 30 12:33:21 2021	soton.uk	138.20.12.1
028ab6d5b25a58e590ac687bbfc8a07f670031caac8893f120bd49c6c4112a99	151885f50543e8e9432a55bce9a3868ea5c799feb8cb513367c83b5c7c436e2e	Fri Jul 30 12:36:21 2021	alpha.gr	138.20.12.1

FIGURE 4.7: Compromised sub-blockchain

We now see in figure 4.7 that all the IP addresses have been changed in order to redirect the user to a specific website that listens on 138.20.12.1. Now, when users request domains from this sub-blockchain, they will be redirected to this website. However, BlockDom's mechanism has identified the manipulation of this sub-blockchain and warns users with an alert.

4.2.4 DNS Fast Flux

The basic idea behind DNS fast fluxing is to have multiple IP addresses associated to a single domain. The addresses will be changing with extremely high frequency through the change of DNS records. Adversaries use DNS fast fluxing often with the help of botnets in order to keep their malicious properties up and running, and stop security teams from blocking their IP address (28).

Consider an adversary who has registered the Hallfax.org domain in the transaction shown in listing 4.4. Notice that the adversary has used a typo-squatting domain name (replaced the letter "i" with the letter "l") which is a misspelled domain name, registered to profit from users' typing mistakes or deceive users into believing that they are the correct target domain (29).

```
{
  "header": "c1000eef54a06dc08255943714a9c28d9cf50e7312711b7776044afc24039947",
  "prevHash": "6619c5939fe681f33e1b463d354685da3a9ec497be00996dbd60333f89792661",
  "timestamp": "Thu Aug  5 11:23:40 2021",
  "domain": "Hallfax.org",
  "IP": "36.11.12.6"
}
```

Listing 4.4: Fraud domain record

As long as the *Hallfax.org* domain is available, the node will construct a block in order to include it in it and append to the blockchain. The *getIP* function will assign the 36.1.12.6 IP address to this domain and the registration will be completed. Consider now that the adversary has a list with compromised machines and their IP addresses. His aim is to assign all the IP addresses to the domain he has registered. These IP addresses will operate as a proxy and finally redirect to his malicious website. In order to do so, the adversary has to add all these IP addresses in the block where his domain is registered, as shown in line 6-7 in listing 4.5.

```
1 {
2   "header": "c1000eef54a06dc08255943714a9c28d9cf50e7312711b7776044afc24039947",
3   "prevHash": "6619c5939fe681f33e1b463d354685da3a9ec497be00996dbd60333f89792661",
4   "timestamp": "Thu Aug  5 11:23:40 2021",
5   "domain": "Hallfax.org",
6   "IP": ["48.28.25.99", "13.33.99.30", "87.28.97.91", "28.81.59.96", "95.36.43.89",
7         "71.78.68.78", "89.56.8.46", "9.25.21.30", "36.85.15.37", "72.48.54.8"]
8 }
```

Listing 4.5: Multiple IP addresses pointing at the same domain

Now, the initial IP address that redirects to the malicious domain will be very hard to be found by a security team who aims to take the domain down.

As we mentioned before, blocks with transactions are linked to each other through their headers. Although the adversary has not made any change to the initial transaction as shown in listing 4.6 the block has been compromised because new data were added in it.

```
{
    "domain" : "Hallfax",
    "IP"     : "36.11.12.6"
}
```

Listing 4.6: Initial block transaction

We said before that even a significantly small change to an already appended block will affect the rest of blockchain. Thanks to the *validatioBlockchain* algorithm, BlockDom is able to identify this change. From the moment that Hallfax.org domain is assigned to 36.11.12.6 and the block has been appended in the blockchian, nothing can change added or deleted from it.

Chapter 5

Conclusion and Perspectives

5.1 Conclusion

As examined in this thesis, the environment in which traditional DNS protocol operated hides several threats which may seem dangerous both to the protocol itself and to the users who use it. Our focus on the implementation of a decentralized — blockchain based DNS protocol enabled us to reveal the practical and theoretical aspects of DNS security to a greater degree than any previous study.

We proposed BlockDom, a blockchain based DNS which is the first approach of replacing a traditional protocol with a blockchain based one, with a view to the gradual decentralization of the web. In this thesis, we presented a first approach of the BlockDom network, by explaining its aims and objectives. We then went through the differences between BlockDom and some popular blockchains (Bitcoin and Ethereum) and compared the way they operate. After that, we explained how BlockDom works by presenting a fundamental overview of its implementation. We presented the way BlockDom handles transactions, blocks and how it achieves consensus. Finally, we performed a series of performance and security testings in order to evaluate the network, and prove that the replacement of the traditional DNS with a blockchain-based DNS, does not negatively affect the basic operation of the protocol, and at the same time can combine equal participation in the network, anonymity, security, and central authority independence.

By presenting BlockDom our main goal is to propose a safer and more efficient version of the traditional DNS protocol. Taking advantage of the blockchain technology we managed to present a robust protocol which will be able to completely replace the traditional DNS protocol offering more security and better performance.

We hope that this thesis builds on the current research outcomes of blockchain based DNS proposals and signifies a milestone in information security in the age of blockchain.

We expect that many more research efforts will follow by expanding on the works classified in this thesis and by applying the techniques outlined here to various business contexts.

5.2 Future Work

As long as sharding architecture can ensure efficient performance in the network, we will mainly focus on strengthen its security. In two out of the three cases, the node that creates the transaction mines itself the block in which it will be included. In the third case, according to which the transaction is forwarded to another sub-blockchain and mined by a different node, there is a high risk of manipulating the transaction. The consensus algorithm should ensure that in case the external node is malicious and wants to modify the transaction content, its attempt will be rejected, and the malicious block will not be added to the blockchain.

Another very important feature we would like to integrate in the BlockDom protocol is the ability of data modification without affecting the integrity of the network. In traditional DNS, records (IP-domain) are often updated, because the domain or the IP address of a host may have been changed. Similarly, BlockDom blockchain should be able to perform updates of its records without affecting the whole network.

5.3 Project Management

Deviation from the original plan

In the initial BlockDom plan, we designed the protocol in such a way that users would need a personal wallet in order to store their keys securely and make new transactions. However, the consensus algorithm we developed seemed able to confirm the new transactions without requiring users to sign them with their private keys. In most cases, the originator of the transaction is the one who will create the block in which the transaction will be included. Thus, we decided that the network users do not need to use wallets and therefore the implementation of the protocol proceeded without their use as we had presented in the original plan.

Furthermore, in the initial plan, we proposed the issuance of a cryptocurrency. The reason for this proposal was because we wanted to use an already existing consensus algorithm in order to make sure that our protocol would meet the security criteria (at the consensus layer). Therefore, we initially chose the PoW algorithm as the consensus algorithm for our protocol. However, after much research, we decided that our work is intended exclusively for technological purposes and not for a commercial-financial product. Thus, we decided to create our own consensus algorithm, which is not related to the issuance of any cryptocurrency.

General Reflection and Potential Modifications

The DNS protocol is one of the most important and popular web protocols. At the same time, however, this protocol presents large security gaps which strengthen the action of malicious users. From a general point of view, we consider that our proposal managed to bring to the surface many of the gaps of the DNS protocol and to suggest solutions which may seem useful enough to cover them. Although many of our proposals have not been scientifically researched by other researchers (as we are the first to propose them), we have managed to show through real scenarios that they are realistic and worth researching.

Finally, the part of testing and evaluation, would be one of the first things we would change the way of conducting. In the present thesis, during the execution of the experiments, a local network was used in which a single node could perform the functions of the protocol and give us the corresponding results. As we all know, blockchains consist of many nodes and so our results do not accurately represent the performance of the network in cases of high volume data usage or high traffic. Thus, the biggest change we would suggest would be to organize a more integrated experiment in a more realistic environment so that our results are more accurate.

Bibliography

[1] M. Cotton, L. Eggert, et. al., “Internet assigned numbers authority (iana) procedures for the management of the service name and transport protocol port number registry,” August 2011. [Online]. Available: https://bit.ly/3glEZOk

[2] Kevin Wood, “What is a second-level domain?” September 2019. [Online]. Available: https://bit.ly/3Bb9tuv

[3] Microsoft, “Overview of dns zones and records,” April 2021. [Online]. Available: https://bit.ly/3DcA4cs

[4] Tony Redmond, “Exchange, windows, and the active directory,” 2008. [Online]. Available: https://bit.ly/3zjNOzO

[5] JONATHAN WEINBERG, “Icann and the problem of legitimacy,” 2008. [Online]. Available: https://bit.ly/3D9uQ0W

[6] Arnaud Legout, Guillaume Urvoy-Keller, Pietro Michiardi, “Understanding bittorrent: An experimental perspective,” November 2005. [Online]. Available: https://bit.ly/2Wonshu

[7] Saurabh Verma, Ali Hamieh, et al., “Stopping amplified dns ddos attacks through distributed query rate sharing,” August 2021. [Online]. Available: https://bit.ly/3sELpNl

[8] Jingqiang Liu, Bin Li, et al., “A data storage method based on blockchain for decentralization dns,” June 2018. [Online]. Available: https://bit.ly/3sELpNl

[9] HU Wei-hong, AO Meng, SHI Lin, XIE Jia-gui, LIU Yang, “Review of blockchain-based dns alternatives,” March 2017. [Online]. Available: bit.ly/3D8ujMy

[10] DmitryBagay, “Blockchain-based dns building,” April 2020. [Online]. Available: https://bit.ly/3yxYBEY

[11] ZHONG YU, DONG XUE, et. al., “Dnstsm: Dns cache resources trusted sharing model based on consortium blockchain,” January 2020. [Online]. Available: https://bit.ly/3t2GndQ

[12] Guobiao He, Wei Su, et. al., "Td-root: A trustworthy decentralized dns root management architecture based on permissioned blockchain," January 2020. [Online]. Available: https://bit.ly/3yv4l2A

[13] Zecheng Li, Shang Gao, et. al., "B-dns: A secure and efficient dns based on the blockchain technology," March 2021. [Online]. Available: https://bit.ly/3kHfV5O

[14] A. Swart, "Squaring the triangle: Secure, decentralized, human-readable names," December 2013. [Online]. Available: https://bit.ly/2XBnSRY

[15] Enis Karaarslan, Eylul Adiguzel, "Blockchain based dns and pki solutions," October 2018. [Online]. Available: https://bit.ly/3D7vUm7

[16] Congcong Ye, Guoqiang Li, et. al., "Analysis of security in blockchain: Case study in 51%-attack detecting," September 2018. [Online]. Available: https://bit.ly/2XR1Yui

[17] Andreas M. Antonopoulos, *Mastering Bitcoin.* 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., December 2014.

[18] ——, *Mastering Ethereum.* 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., December 2018.

[19] Extreme Marketing Team, "Monitoring dns nxdomain," March 2021. [Online]. Available: https://bit.ly/3j13FgO

[20] Gang Wang, Zhijie JerryShi, Mark Nixon, Song Han, "Sok: Sharding on blockchain," October 2021. [Online]. Available: https://bit.ly/3mpA4Qb

[21] bit2me ACADEMY, "What is a seed node?" July 2020. [Online]. Available: https://bit.ly/3sz0y2G

[22] JAKE FRANKENFIELD, "Consensus mechanism (cryptocurrency)," August 2021. [Online]. Available: https://bit.ly/3xZZvdm

[23] The Linux Foundation, "Hyperledger architecture, volume 1," 2017. [Online]. Available: https://bit.ly/2WjoviI

[24] Du Mingxiao, Ma Xiaofeng, et. al., "A review on consensus algorithm of blockchain," October 2017. [Online]. Available: https://bit.ly/3kgelHS

[25] Lance Whitney , "How dns attacks threaten organizations," June 2020. [Online]. Available: https://tek.io/3t87ynt

[26] Hitesh Ballani, Paul Francis, "Mitigating dns dos attacks," October 2008. [Online]. Available: https://bit.ly/3D8dVMi

[27] U. Steinhoff, A. Wiesmaier, and R. Araújo, "The state of the art in dns spoofing," 2006. [Online]. Available: https://bit.ly/3mnPxQG

[28] Cloudflare, “What is dns fast flux?” [Online]. Available: https://bit.ly/3z9vCbO

[29] Constantinos Patsakis , Fran Casino, et. al., “Unravelling ariadne’s thread: Exploring the threats of decentralised dns,” July 2020. [Online]. Available: https://bit.ly/37ZsvHF