STRONA TYTUŁOWA PRACY DYPLOMOWEJ

(*strona nr 1, ale numer strony nie może być widoczny*)

Student, z platformy *mojaPG,* pobiera stronę tytułową pracy dyplomowej. Wzory stron tytułowych zostały wprowadzone Zarządzeniem Rektora Politechniki Gdańskiej nr 15/2014 z dnia 24 marca 2014 r., jako:

- załącznik nr 1/1 – praca dyplomowa magisterska,
- załącznik nr 1/2 – praca dyplomowa inżynierska,
- załącznik nr 1/3 – projekt dyplomowy inżynierski,
- załącznik nr 1/4 – praca dyplomowa licencjacka.

Jeżeli praca dyplomowa/projekt dyplomowy, zwana dalej pracą, jest realizowana przez co najmniej 2 studentów, na stronie tytułowej najpierw umieszczone są dane studenta firmującego ten egzemplarz pracy, następnie dane pozostałych studentów.

Od strony **Streszczenie** praca wykonywana przez co najmniej 2 studentów zawiera identyczną treść.

OŚWIADCZENIE AUTORA PRACY

(*strona nr 2, ale numer strony nie może być widoczny*)

Student, z platformy *mojaPG,* pobiera **Oświadczenie** zgodne z załącznikiem nr 2 Zarządzenia Rektora Politechniki Gdańskiej nr 15/2014 z dnia 24 marca 2014 r.:

- załącznik nr 2/1 – Oświadczenie studenta, realizującego pracę dyplomową,
- załącznik nr 2/2 – Oświadczenie studenta, realizującego projekt dyplomowy,
- załącznik nr 2/3 – Oświadczenie studenta, realizującego pracę dyplomową w ramach programu o podwójnym dyplomowaniu.

STRESZCZENIE

Niniejsza praca przedstawia analizę sieci anonimizujących (ACN), koncentrując się na ich podstawach technicznych, praktycznych zastosowaniach oraz właściwościach związanych z bezpieczeństwem. Praca rozpoczyna się od teoretycznego przeglądu ewolucji ACN, rozróżniając anonimowość zapewnianą przez politykę od anonimowości wynikającej z projektu, a także omawiając kluczowe modele architektoniczne, takie jak sieci miksujące i trasowanie cebulowe.

Centralną częścią pracy jest wielokryterialna analiza porównawcza, łącząca przegląd literatury z eksperymentalnymi pomiarami pod kątem użyteczności i efektywności ACN w różnych scenariuszach zastosowań, przeprowadzona dla najważniejszych współczesnych sieci: Tor, I2P, Lokinet oraz Nym. Wyniki jednoznacznie pokazują, że nie istnieje jedna, uniwersalnie najlepsza ACN; optymalny wybór zawsze zależy od konkretnego przypadku użycia.

Aby wesprzeć praktyczne zrozumienie uzyskanych wyników, opracowano demonstrator dydaktyczny. Pozwala on studentom na eksplorację tematu sieci anonimizujących w ramach ćwiczeń laboratoryjnych, obserwację ich mocnych i słabych stron oraz zrozumienie, jak różne podejścia radzą sobie z wybranymi podatnościami. Praca kończy się rekomendacjami dotyczącymi dalszych badań, obejmującymi obserwację nowych projektów ACN, prowadzenie zaawansowanych pomiarów wydajności przy różnych konfiguracjach oraz analizę wpływu parametrów sieci na użyteczność i bezpieczeństwo.

Słowa kluczowe: Tor, I2P, Loki, Nym, ACN, analiza, przypadek użycia, obszar zastosowań

ABSTRACT

This thesis presents an analysis of anonymous communication networks (ACNs), focusing on their technical foundations, practical applications, and security properties. The work begins with a theoretical overview of the evolution of ACNs, distinguishing between anonymity by policy and anonymity by design, and examining key architectural models such as Mix-nets and onion routing.

A central part of the thesis is a multi-criteria comparative analysis, combining literature review with experimental measurements of ACN usability and effectiveness across different use case scenarios, performed for today's ACNs: Tor, I2P, Lokinet, and Nym. The results clearly demonstrate that no single ACN is universally optimal; instead, the best choice depends on the specific use case.

To support the practical understanding of these findings, an educational demonstrator was developed. This demonstrator enables students to explore ACN designs in hands-on laboratory exercises, observe their strengths and weaknesses, and understand how different approaches mitigate specific vulnerabilities. The work concludes with recommendations for future research, including monitoring new ACN designs, conducting advanced performance measurements under varied configurations, and studying the impact of network parameters on usability and security.

Keywords: Tor, I2P, Loki, Nym, ACN, analysis, use case, application area, usage

TABLE OF CONTENTS

ABBREVIATIONS

ACN – Anonymous Communication Network

P2P – Peer-to-peer

ECC – Eliptic-curve cryptography

PQC – Post-quantum cryptography

PDU – Protocol data unit

LAN – Local area network

DoS – Denial of Service

1. INTRODUCTION

1.1. Goal of the work

Ensuring the anonymity of communication in diverse network systems is a technical challenge that requires specific technical solutions that enable the achievement of this objective while maintaining the usability of the network system. The aim of this work is to analyse the types of technical solutions currently used in anonymising networks/anonymous communication networks (ACNs) in the context of their usability in specific usage scenarios.

1.2. Scope of the work

This work provides an overview of ACNs, emphasising their technical aspects, identifies potential use cases and application areas, categorises these use cases, presents potential threats, and offers a comparative analysis of existing anonymous communication networks in terms of usability in various scenarios. The work also determines the desired directions for the development of ACNs and includes the design and implementation of an educational demonstrator to verify and illustrate key elements of the analysis.

1.3. Structure of the work

This work begins with a theoretical introduction explaining all the concepts necessary to understand the paper. After the introduction, an overview of the most prominent ACNs is provided, focusing on those most popular today and those that have significantly influenced current solutions. Following the overview, the use cases for ACNs are proposed. The use cases are then organised into groups of similar requirements. The criteria for each group are selected and appropriately weighted. After presenting the use cases, relevant threats are discussed, including limitations and possible attacks. Subsequently, the technical comparison of the most popular ACNs and their technical solutions is carried out in two stages. The first stage is the comparison based solely on the literature. Then, the experiments are performed and the results are used in the second stage, an empirical and experimental comparison. After comparisons, ACNs are evaluated in terms of usability for the identified use case groups and their requirements. Based on this evaluation, the best-suited ACN is identified for each use case group. Following the analysis, future directions for ACNs are described. After presenting future directions, an educational demonstrator is designed to verify and illustrate the ACN analysis. Finally, the work concludes with a summary of the findings and a discussion of the results.

2. THEORETICAL INTRODUCTION

2.1. Introduction

This chapter describes concepts that will be essential for understanding the material covered in the future chapters.

2.2. Anonymity and privacy

The definition of anonymity used in this thesis is based on a paper "A terminology for talking about privacy by data minimization" [1]. According to the paper: "Anonymity of a subject means that the subject is not identifiable within a set of subjects, the anonymity set.". Another important property in anonymous communication systems is unlikability. According to the same paper: "Unlinkability of two or more items of interest (IOIs, e.g., subjects, messages, actions, ...) from an attacker's perspective means that within the system (comprising these and possibly other items), the attacker cannot sufficiently distinguish whether these IOIs are related or not." Pseudonymity refers to the use of pseudonyms as identifiers, where a pseudonym is an identifier of a subject other than one of their real names. Privacy is the right to keep personal matters and relationships secret, as well as the ability to determine when, how, and which information about oneself is revealed. Anonymity can be considered as a method to achieve privacy.

2.2.1. Why is anonymity and privacy needed?

For the same reason that in all well-prosper democratic countries there are anonymous elections. No one should be obliged to reveal information they do not wish to disclose; individuals should have the right to decide to whom and what information they reveal. In recent years, privacy has been increasingly compromised as personal data have become highly valuable to large corporations. Data are sometimes said to be the petroleum of the 21st century, highlighting their significant value in the modern economy. As with many technologies, anonymity can be used for both positive and negative purposes. Although negative uses often receive more attention in the media, this thesis will primarily focus on beneficial use cases to provide a balanced perspective. It should be noted that attempts to restrict anonymous networks primarily affect users with legitimate intentions, as malicious actors are often able to circumvent such restrictions and may use alternative solutions that do not require scalability. Some organisations claim they do not retain the content of communications, but only metadata - such as who communicates with whom, when, where, and how often. NSA General Counsel Stewart Baker said: "metadata absolutely tells you everything about somebody's life. If you have enough metadata, you don't really need content." and General Michael Hayden, former NSA director, added that "We kill people based on metadata" [2]. Metadata, in fact, should not be neglected when it comes to the privacy topic. According to the European Data Protection Supervisor, privacy is a fundamental human

right and a component of a sustainable democracy [3].

2.2.2. Anonymity by design and anonymity by policy

Sometimes service providers, VPN providers in particular, advertise their services as anonymous, although in reality they can deanonymise users on demand, they just claim that they would not do it or that they keep logs of their user activity. However, history shows that such assurances are not always reliable and that users cannot be certain that anonymous service providers will not disclose or sell their information [4]. This approach is known as anonymity by policy. On the other hand, anonymity by design makes it impossible for the service provider to reveal the identity of the service users by creating a service or network in such a way. In other words, there is no need to trust any third party other than the code itself, which you can verify in most cases because they are usually open source. Proper privacy can only be achieved with anonymity by design.

2.3. Client-server and peer-to-peer models

2.3.1. Client-server model

The client-server model is currently the most popular computing paradigm. Essentially, it assumes that many clients connect to centralised servers. Today, these servers are typically run in large data centres controlled by major organisations.

2.3.2. Peer-to-peer model

Peer-to-peer (P2P) computing is a decentralised network architecture where each participant (peer) has equal privileges and can initiate or complete transactions without relying on a centralised server. While it may be surprising, the Internet and its predecessors were not designed to be client-server oriented. In fact, the basic idea of peer-to-peer computing dates back to the Arpanet, the predecessor of the Internet, and the first Request for Comments [5], which was associated with it. Arpanet was created in the way that it would be resilient to a potential nuclear attack, and therefore it was not centralised.

2.4. Distributed Hash Tables

Distributed Hash Table is a distributed system of storing data in the form of key-value pairs. For example, we can think of our key-value pairs as positions in a dictionary where the key "Kashubians" would have the value "ethnic group native to the Pomerania region in northern Poland". In DHT, keys exist in the form of hashes. Thanks to this fact, the length of a key is not an issue as all keys have equal lengths, which is a property of hash functions. All possible keys create a keyspace. Thanks to another important property of hash functions, keys will be evenly distributed across the whole keyspace. If we consider hashes big enough, for example, of 256-bit length, then collisions should not be an issue either; that is another desired property of hash functions. Usually, the keyspace is big. If we consider 256-bit

hashes we have 2^{256} possible keys that correspond to some values. All needs to be stored somewhere, and one machine or computer is not always enough. Even if the data would fit in one device, it is often desired for storage to be distributed in order to gain several other benefits like the avoidance of a single point of failure or an ability to avoid trusting only one central entity that stores all the data. It is important to establish which keys will be assigned to which machine. There are several approaches to this topic, and the ones that are essential for the future sections will be described.

2.4.1. Chord

Chord was one of the first distributed hash tables (DHTs), proposed in 2001 [6]. Chord aimed to address the following issues related to development of peer-to-peer systems and applications:

- Load balancing - Chord evenly distributes keys over nodes providing load balancing.
- Decentralisation - Chord has fully distributed and hierarchy-less nature, removing single point of failures that were major issues with projects like Napster, and increasing robustness.
- Scalability - as the time complexity of the lookup in the Chord network is O(log(n)), it allows for large numbers of nodes in the network.
- Availability - Chord automatically adjusts to changes in the network like node joins and leaves, keeping the network functioning regardless.
- Flexible naming - The structure of the keys is not constrained; therefore, there is a wide range of possibilities of mapping names to Chord keys.

In the Chord paper, authors also point out potential use cases for Chord, including cooperative mirroring where entities in the network can distribute load across themselves - mirror their distributions, time-shared storage that solves the problem of sharing content by not-always available machines, distributed indices for keyword search and large-scale combinatorial search where nodes cooperate computationally in order to get a solution to a given problem. Chord has the structure of a ring where keys with bit identifiers are mapped. Due to consistent hashing, the keys are evenly distributed throughout the ring. The keys are stored in nodes, and the nodes are machines within the network. If a network uses m-bit identifiers, then the network consists of the maximum 2^m nodes and keys, as the nodes and keys are in the same identifier space. Each node has a set of pointers called fingers, each finger points to another node that is 2^k jumps away, where $0 <= k <= m - 1$, which means the "shortest" finger points to the next node in the ring and the "longest" one points to the node that is exactly half a ring away. The basic principle of an operation in the Chord network is simple: the keys are stored at the node that is their closest predecessor. Combining that with the fact of using finger pointers, in the worst case during the lookup the distance is cut in half; it is the reason why the lookup has O(log(n) time complexity. It is important to note that in most cases it will be a much smaller number as the keys are uniformly distributed, and the worst case occurs rarely. For example, in the 2^12 node network in theory there should be 12 hops, but according to the paper the most probable number is 6 hops.

2.4.2. Kademlia

Kademlia is a specific type of DHT and was proposed in 2002 [7]. In the Kademlia system, each node is placed in the keyspace. Each node stores values that are near him. In order to calculate the distance it uses the XOR operation metric. XOR operation can show the difference between two numbers as the output of bits with the same value will be 0 (0,0 or 1,1) and 1 if the bit is different (0,1 or 1,0). This implies that the distance between a given node to itself is 0 (as XOR of every bit with itself always gives 0 as a result) and the distance to any other key is always more than 0, assuming no collisions. Also, the order of the numbers in the XOR operation does not matter as XOR(A,B) is equal to XOR(B,A). The distances from a given node to other nodes can be visualised as a binary tree, which will also be this node routing table. It is important to note that this tree does not represent node identifiers but distances. Identifiers are stored as values on the leaves. A node in this binary tree represents a change in a given bit (value 1) or a lack of change (value 0); therefore, the leaf most left (all zeros) will be the node itself. The height of the tree represents the size of the keyspace as the keyspace will have a size of $2^{heights}$. For every possible divergence, that is, any difference in a bit that occurs after a series of zeros, a bucket is created. There is an arbitrary limit of nodes set in each bucket. The buckets that diverge earlier, for example, due to the difference on the first bit, have more possible space to fill, but they have the same limit as the ones that diverge later or in other words the ones that are closer to the node whose distance tree that is considered. This leads to an important property of the Kademlia system: there is more knowledge about the closer nodes than about the further ones. Thanks to this property, we have another benefit - O(log(n))-hop routing. Each time a node looks for a specific value by asking the most similar node, it usually halves the distance to that value.

2.5. Anonymous communication network

Anonymous Communication Network is a kind of network that allows anonymous uni or bidirectional communication. They are usually implemented as overlay networks. An overlay network is a logical network that operates on top of an underlying physical network.

2.6. Information security attributes

Information security, or InfoSec, is a practice of protecting information. Several attributes can be distinguished, slightly varying depending on the definition. From the point of view of this work, the most important will be four of them:

1. Confidentiality - Only authorized persons can access data. In terms of cryptography and encrypted messages, it comes down to the ownership of the appropriate decryption key.
2. Authenticity - Identity of the person that performs a certain operation is the same as declared.
3. Integrity - Data were not modified in an unauthorized way.
4. Non-repudiation - persons taking part in the information share process cannot deny their participation in it.

2.7. Cryptography basics

2.7.1. What is cryptology, cryptography and cryptanalysis?

Cryptology is the science of secure communication. Cryptography is a science of concealing the meaning of a message. Cryptanalysis, on the other hand, is a science of breaking the concealment provided by cryptography. In essence, it can be said that cryptology is combined cryptography and cryptanalysis.

2.7.2. Symmetric cryptography

In symmetric cryptography, one shared key is used for both sides of a communication to encrypt and decrypt messages. Symmetric cryptography ensures confidentiality, as (ideally) the key is needed in order to read the content of a message, and the key is only shared between two sides of the communication that are often referred to as Alice and Bob. Symmetric cryptography ciphers can be divided into two subcategories:

1. Block ciphers - Fixed-size blocks are encrypted. In case the data does not fit the block, a padding is used.
2. Stream ciphers - One bit or byte is encrypted at a time.

2.7.3. Asymmetric cryptography

In asymmetric cryptography, or public-key cryptography, each entity uses a pair of keys instead of one shared symmetric key. The private key is a part that, as the name suggests, is specific to the entity exclusively and cannot be revealed to anyone else. However, the public key can be freely distributed. The message encrypted with the public key can only be decrypted with the private key from the same pair. Analogically, the message "encrypted" with the private key can be "decrypted" with the public key, although calling it encryption and decryption does not make much sense as the public key is freely distributed, therefore everyone can read the message. The more appropriate terms that are used in this example are signing and verifying, as if we can decrypt a given message with a certain public key, then we are sure that the one who signed it must be the owner of the private key from the same key pair. Encryption with asymmetric cryptography is in general much slower than with symmetric one; therefore, it is most often used for different purposes, such as the digital signature, which will be described later.

Elliptic-curve cryptography (ECC)

An approach to asymmetric cryptography using elliptic curves over finite fields has gained popularity in recent years, although it is not a new technique, as its origin dates back to 1985. ECC is believed to offer the same level of security as other widely used asymmetric cryptographic methods but with significantly shorter key lengths. One of the popular examples of a curve that can be efficiently used in this kind of cryptography is Curve25519.

2.7.4. Hash functions

Hash function algorithms allow for transformation of any input into a fixed-size output, called a hash, in a deterministic way, which means that each time a certain input produces the same output if the same hash function and its parameters are used. Another important feature of hash functions is its one-sidedness, in that the original input cannot be determined based on the produced hash. The hash function should also produce output that is uniformly distributed, and it should be hard to find two inputs that produce the same output - this situation is called a hash collision.

2.7.5. Digital signature

One of the most popular use cases of asymmetric cryptography is the digital signature. It is a mathematical scheme for verifying the authenticity, integrity, and non-repudiation of origin for the given message. In a simplified way, it works as given: The sender generates a private/public key pair and distributes the public key. He generates a hash of the message he has written and signs this hash with his private key. The receiver receives the message along with the signed hash. As the hash function is known, he creates a hash of the message himself with it, verifies the signed hash, and compares the two result hashes. If they are equal, then the signature is valid.

2.7.6. Quantum and post-quantum cryptography

Quantum cryptography is the science of using quantum mechanics in cryptography. Quantum cryptography algorithms are designed to work specifically on quantum computers. On the other hand, post-quantum cryptography (PQC) refers to the development of cryptographic algorithms that work on traditional machines but that are resistant to potential attacks that use quantum computers.

2.7.7. Popular cryptographic algorithms and hash functions

- AES - Advanced Encryption Standard, the most popular block cipher today, considered as a safe symmetric cryptography solution, especially with longest possible 256-bit keys. Due to its popularity, it often has dedicated hardware support.
- RSA - one of the first asymmetric cryptography algorithms and still the most popular one. The name RSA is an abbreviation from its creators' surnames: Rivest, Shamir, Adleman. Its security relies on the difficulty of the factoring problem, which means factoring the product of two large prime numbers. It can be used for both encryption and digital signature.
- ElGamal - the second most popular asymmetric cryptography algorithm, after the RSA. It is based on the discrete logarithm problem. It supports both encryption and digital signatures.
- DSA - Digital Signature Algorithm is an asymmetric cryptography algorithm used specifically for the digital signatures, not for the encryption. It is based on the discrete logarithm problem.
- ECDSA - Elliptic Curve DSA is a variant of DSA utilising elliptic-curve cryptography.
- EdDSA - Edwards-curve DSA is a variant of DSA utilising Schnorr signature based on twisted

Edwards curves. Even though the name might suggest correlation with ECDSA they both are completely different signature schemes. Ed25519 is a specific example of the EdDSA variant and uses the edwards25519 curve, which is related to the popular Montgomery curve Curve25519.

- Diffie-Hellman Key Exchange - or simply Diffie-Hellman is a key agreement algorithm that two parties can use in order to agree on a shared secret. The shared secret is converted into keying material, and the keying material is used as a symmetric encryption key. Diffie-Hellman Key Exchange utilises the properties of modular arithmetics.
- ECDH - Elliptic-curve version of the DH key agreement. Usually, Curve25519 is chosen as a curve because it is fast and not proprietary.
- SHA - Secure Hash Algorithms are a family of hash functions. Currently, there are four versions of SHA:
    1. SHA-0 - refers to the original hash function with 160-bit output that was published under the name SHA.
    2. SHA-1 - improved version of SHA-0, still utilising 160 bits. NSA designed it as part of the DSA algorithm.
    3. SHA-2 - a family of two hash functions designed by NSA: SHA-256 and SHA-512. The postfix in the name determines the length of the output of the hash function (256 bits or 512 bits). There are also modified and truncated versions of these two standards: SHA-224, SHA-384, SHA-512/224, and SHA-512/256.
    4. SHA-3 - hash function, also known as Keccak. Although the hash lengths are the same as in the SHA-2 family, the inner workings of the algorithm are significantly different.

2.8. ISO/OSI reference model and TCP/IP suite

Two most popular networking models used today are ISO/OSI and TCP/IP. They both help us to analyse the end-to-end communication between two parties. They both consist of several layers: ISO/OSI consists of 7, while TCP/IP of 4. Each of these layers has certain clear goals to fulfill. Each layer has certain protocols and protocol-specific units of information composed of user data and control information called protocol data units (PDUs).

The ISO/OSI model has the following layers:

1. Physical layer - the first and the lowest layer, responsible for transmission of unstructured data between a device and transmission medium. The PDU of the physical layer is a bit.
2. Data link layer - the second layer, responsible for node-to-node data transfer between two directly connected devices. In this layer, we can distinguish two sublayers:
    - Media Access Control - usually implemented in hardware, manages access control, encapsulates and decapsulates data, adds header and trailer
    - Logical/Data Link Control (LLC/DLC) - is responsible for communication with physical and network layers and for the flow control.

The PDU of the data link layer is a frame. In real-world examples, technologies designated for the lowest layers usually cover both the physical and data link layer. Examples include Ethernet, the most popular wired local area network (LAN) technology today, or 802.11, sometimes vulgarly referred to as Wi-Fi, the most popular wireless LAN technology.

3. Network layer - the third layer, responsible for addressing hosts, connectionless communication, routing, and relaying messages. It is also the first layer that is "sentient" of the network, which means that there is something outside the current machine or current link. The most popular protocol of this layer is the IP protocol in two versions: IPv4 and IPv6. The PDU of the network layer is a packet.
4. Transport layer - the fourth layer, responsible for interprocess communication. Its role is to address ports and control connection, flow and errors as well as message multiplexation and demultiplexation. There are two most popular protocols in this layer: TCP, responsible for reliable communication with the cost of delays, and UDP, responsible for unreliable communication, but with smaller delays. The PDU in the transport layer is a segment for TCP and a datagram for UDP.
5. Session layer - the fifth layer, responsible for maintaining and managing interprocess sessions. The PDU of the session layer is data.
6. Presentation layer - The sixth layer, sometimes referred to as the syntax layer, is responsible for translating, coding, and decoding data between applications. The PDU of the presentation layer is also data.
7. Application layer - the seventh and the last layer in the ISO/OSI model application layer is responsible for providing an interface for communication between applications. Once again, the PDU of the application layer is data.

For the sake of simplicity, it can be said that in the TCP/IP model there are four layers that correspond to one or more ISO/OSI layers:

1. Link layer - the layer that serves as a combination of ISO/OSI first and second layer.
2. Internet layer - the layer that corresponds to the network layer from the ISO/OSI model.
3. Transport layer - the layer that corresponds to the layer with the same name from the ISO/OSI model.
4. Application layer - the layer that combines session, presentation, and application layers from the ISO/OSI model.

The ISO/OSI was a model created from the ground up in the late 1970s and early 1980s as a framework based on the set of tasks that needs to be fulfilled in order to have reliable communication and each layer in this model was assigned a specific role. One of the most criticised elements of the ISO/OSI model were layers 5 and 6 which were thought to be an exaggeration, as they can be easily integrated with the application layer. That is also why often, when referring to the application layer, the fifth, sixth, and seventh layers are grouped together. Regardless of the criticism, the model is still used as

a reference model. TCP/IP is an evolving model whose origins date back to ARPANET [5]. It has a simpler, more concise, and pragmatic approach compared to the ISO/OSI model, and it has specific protocol solutions. An important difference between ISO/OSI and TCP/IP models is the fact that in the ISO/OSI the communication is only possible between adjacent layers, while in TCP/IP any layer can refer to any other layer, including the same layer. An example can be the IP protocol utilising ICMP and vice versa. This paper will focus on layers 3-7 from the ISO/OSI model (or layers 2-4 from the TCP/IP model). When referring to the seventh layer from the ISO/OSI model, fifth and sixth will also be included implicitly in the reference for the sake of simplicity.

2.9. Blockchain and cryptocurrencies

Blockchain is a shared, immutable, and distributed ledger that records information in a growing list of entries called blocks. Blocks are chained together using hash functions. Each new block depends on the integrity of the previous blocks in an unaltered stage and therefore the history of a blockchain cannot be altered. Blockchains are often used as the backbones of various cryptocurrencies, digital currencies based on cryptography, where transactions are written as records on the blockchain with an appropriate consensus mechanism. In today's world, cash usage is declining as countries shift toward cashless payments. This trend poses a major threat to privacy. Unlike cash transactions, which are private and typically known only to the two parties involved, cashless payments lack this privacy. Every cashless payment is visible to the government, tax authorities, banks, and possibly many more entities. This information can be used against the people who make such transactions. Cryptocurrencies offer a way to restore the privacy of transactions. They allow for private peer-to-peer transactions between users. Additionally, cryptocurrencies have excellent transfer properties, enabling reasonably quick transfers, regardless of the day of the week, if it is a national holiday or not. Unlike traditional bank transfers, cryptocurrency transactions cannot be blocked or restricted by financial institutions or government entities.

2.10. Network protocols overview

- IP - Network layer communication protocol in the ISO/OSI model and the Internet layer protocol in the TCP/IP responsible for connectionless communication, relaying datagrams and addressing hosts. It is the most popular protocol for this layer. IP is often associated with convergence as it serves as a universal protocol that supports both upper and lower layer protocols. The IP protocol exists in two versions: IPv4 and IPv6. The IPv4 was introduced first and it had some significant flaws, among which the most critical one is the address space - it was not predicted that the Internet would be so huge that 4 billions of potential addresses would not be enough; as we know today, it is definitely not. There are many actions taken to postpone the issues related to the exhaustion problem as much as possible like the introduction of classless addressing, public/private address distinction with NAT mechanism, but the only viable long-term solution is moving on to

the newer version of the IP protocol, IPv6. It can have up to 2^{128} addresses, for a comparison there are about $2^{6}2$ sand grains on Earth, so it is more than enough for any possible use cases that we can predict today.

- TCP - one of the two most popular transport layer protocols responsible for interprocess connection-based communication. Provides connection, flow, and error control. TCP is used when reliability is more important than potential delays.
- UDP - second of the two most popular transport layer protocols responsible for connectionless interprocess communication. It lacks error control and flow control. It does not guarantee the delivery of a datagram or its correct order; however, these mechanisms can be implemented in the higher layers.
- TLS - Transport Layer Security (TLS) is a protocol widely used for ensuring confidentiality and authentication for client-server applications. The predecessor of the TLS protocol was SSL; therefore, TLS is sometimes still referred to as SSL for historical reasons.
- SOCKS - The SOCKS protocol allows network packets to be exchanged between the client and the server with an intermediary proxy server between. Currently, the latest version of this protocol is SOCKS5.
- BitTorrent - BitTorrent is a peer-to-peer protocol for file sharing. Its important feature is decentralisation, meaning that there is no possibility to take down one central server as it was for some other peer-to-peer file-sharing solutions like Napster. Users utilise BitTorrent clients, the programs that allow them to share files via BitTorrent. The communication is optionally assisted by the BitTorrent trackers that provide available files and find peer users, known as seeds. The other option, depending on the BitTorrent client implementation, is, for example, a distributed hash table as an alternative for the trackers. In BitTorrent files are divided into pieces and the file that holds a list of these pieces is called Torrent. After downloading a piece, the user starts sharing it with other peers, becoming one of the sources of this piece (seed). Each piece is associated with its hash to provide integrity.

3. OVERVIEW OF THE ANONYMOUS COMMUNICATION NETWORKS

3.1. Introduction

This chapter reviews the history and evolution of Anonymous Communication Networks (ACNs), highlighting key milestones, major designs, technical solutions, and the most influential systems, especially those in use today. The section provides context for understanding how modern ACNs developed and introduces the main networks that will be analysed in the following chapters.

3.2. History of ACNs

The first paper that described the ACN idea is considered to be David Chaum's master thesis about Mix networks [8], written in 1979, published in 1981. Even to this day, most of the anonymous communication systems utilise ideas that were first described by Chaum. Due to his merits for anonymity online, he was repeatedly awarded. The Chaum Mix-nets was ahead of its times, not only in terms of demand for the anonymous communication, but also in terms of the computer power needed for repeatedly performing complex public-key cryptographic operations. It took more than a decade for the first anonymous communication networks to be successfully deployed.

First ACN solutions were remailers - a proxy servers that received messages and acted accordingly to the instructions that were attached to the message. The first widely used ACN-like solution was the Penet remailer [9], deployed in 1993, however it was not yet utilising Chaum's concept - it was a simple single-hop pseudonymous remailer, also known as Type 0 remailer, and did not ensure privacy by design. It served as a proxy that stripped the metadata of the message and forwarded it to the recipient with the possibility of reply. The Penet had several security flaws, including lack of encryption and logging user activity, and was eventually discontinued.

Around the same time, although initially less known, the Cypherpunk remailer was introduced, otherwise known as Type I remailer. It was associated with the Cypherpunk movement - a movement that since late 1980s promoted cryptography and privacy for everyone, instead of only the military as it used to be before. It put a strong emphasis on encryption, lack of logging and enabled people to chain their messages through multiple servers - it was the first real life implementation of Chaum's concepts, although partially. The first faithful to the original implementation of Chaum's Mix-nets design was Mixmaster, also known as Type II remailer, created around 1995. It implemented the most important Mix-nets concepts: layered encryption, batching and mixing and gained significant recognition. Over the years many Mix network-based solutions were introduced, addressing issues related to the previous designs.

Mix-net is not the only design that was proposed. In 1988 David Chaum published a paper about DC-nets [10] - an alternative approach for achieving anonymity for both sender and receiver. Similarly to the mix networks, it gave a foundation for further researchers and many later designs based

their principle of operation on DC-nets. Even though there were attempts of deploying ACNs based on DC-nets, none of the ACNs used today utilise this approach due to several limitations.

Another breakthrough design was onion routing [11, 12], a low-latency, connection-based, bidirectional communication network, that similarly to the Mix-nets utilise layered encryption. The work on it began in 1995 and was funded by ONR - the science and technology agency for the U.S. Navy. The first paper about the onion routing was released in 1996, however the final version of it was first published in 1997 and a year later it was published in IEEE Journal on Selected Areas in Communication. Most of the ACNs used today use architecture of the onion routing. One of the examples is Tor - the most popular ACN today, first deployed in 2002 as a direct successor to the original onion routing idea that was meant to be used by common people. The design was described in 2004 paper [13]. During the years it evolved and introduced many new features that can be seen today. Another prominent project that is based on a slightly modified onion routing is I2P - the modification is named Garlic Routing. I2P is a peer-to-peer network - every user also acts as a router. Theoretically, the work on I2P started in 2001; however, the architecture was entirely redesigned in 2003. Similarly to Tor, it is constantly evolving. One of the newer onion routing-based ACNs that can be used today is Lokinet - first described in 2018 [14]. Lokinet implements a custom protocol called LLARP - Low Latency Anonymous Routing Protocol, and contrary to Tor and I2P it is placed on the network layer and can support any IP-based protocol.

While onion routing dominated the scene of ACNs, they have some drawbacks, particularly vulnerability for traffic analysis. Even though the Mix-nets are more resilient to this kind of attack, they suffer from a significant latency. Loopix, proposed by Piotrowska et al. in 2017 [15], is an acceptable-latency Mix-net architecture which led to a new rise in the research on Mix-nets. One of the results of this rise is the Nym network [16], first described in 2021. The official launch of NymVPN - a service that allows for the utilisation of the Nym network - was performed in early 2025.

3.3. Single-hop proxies

The principle of operation of single-hop proxies is straightforward: strip the metadata related to the sender before forwarding it to the recipient. The major advantage of them is their low latency due to the simplicity and the fact that there is only one node between the sender and the receiver. Such a single point of failure, however, is not a desirable characteristic of an anonymous system, regardless of the claims of not keeping sensitive user data, as it imposes anonymity by policy rather than anonymity by design. As this architecture does not provide the anonymity by design property, it will not be treated as a proper anonymous communication network, however due to its historical meaning it is mentioned here. Notable examples from this category include Penet [9] and Anonymizer [17]. Additionally, many VPN services commonly used today can also be considered as examples of this category.

3.3.1. Penet

Penet was the first widely known single-hop proxy and the most prominent example of the family of pseudonym-based solutions, called pseudonymous remailers or nym servers. Sometimes they are also referred to as Type 0 remailers. Their common characteristic is that they remove sender-specific information that could potentially identify senders, assign them a pseudonym, and forward the message to the recipient with the possibility to reply. Penet was created in 1993 by Johan "Julf" Helsingius from Finland. The creation was a result of a discussion on a Finnish Usenet newsgroup about whether people should be required to use their real information for online communication. Johan believed that if such accountability were enforced on Internet users, they would find a way around it. To prove his point, he created Penet. Remailers serve as an intermediary between sender and receiver. Penet, which served as such a remailer, removed metadata that could potentially identify the sender and then forwarded the message without sensitive data to the receiver. It also served as an e-mail box that allowed users to assign their pseudonymous identities to their messages and receive messages sent to their anonymous addresses. Penet had some vulnerabilities that could potentially compromise its users. For example, it sent the messages in plaintext. The lack of confidentiality of these messages might have led to the disclosure of sensitive information in case of the presence of an eavesdropper. Penet also kept a list of its users with their real e-mail addresses and its mapping to the anonymous ones. It posed a threat in case of breaking into the server. Penet was closed due to legal issues in 1996.

3.4. Mix-nets

3.4.1. Original design

The first system that provided anonymity by design was described by David L. Chaum in his master's degree thesis "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms" in February 1979. Mix networks use public key cryptography in order to hide both the sender and receiver of an electronic mail as well as the content of their communication. They do it by encrypting a message with multiple layers of public key encryption and passing them through multiple nodes called mixes. Each node changes the message order, delays them, adds random information and padding in order to minimise the risk of traffic analysis and associating the sender with a given message; in other words, to obfuscate correlation between the input and the output. It also removes the corresponding layer of encryption and passes it to the next node in the path. In essence, we can say that the sender knows all nodes in the path, each node knows only its neighbours, and the receiver knows only the final (exit) node. So the identity of the recipient is only known to himself, the sender, and the exit node; and the identity of the sender is only known to the first (entry) node. The sender can send his message anonymously while the recipient can respond to the untraceable return address, constructed using a real address, two random strings, and public key cryptography. Communication participants exchange these addresses through multiple nodes, but only the addressee can decrypt them. In 1964 [18] Paul Baran, Polish Jewish American engineer, proposed a solution to the traffic analysis problem; the downside of

this solution was that it required a trusted common authority. Chaum's Mix-net network does not need a trusted third party to achieve traffic analysis resilience. Mix-nets also propose digital pseudonyms in order to identify entities online. They are nothing more than public keys that can be used for signature verifications made by an anonymous entity. The authority creates a list of pseudonyms and can decide which are trustworthy and which are not. Pseudonyms, contained in a letter from each entity to the authority, are processed together in a single batch and create together a complete roster. These letters can be additionally protected in order to provide an additional level of security. A person cannot be traced by his pseudonym in the final list; therefore it allows verifying and communicating securely with assured anonymity. One of the major disadvantages of the Mix-net is high latency correlated with public key cryptography, delays, and other aspects that are supposed to provide anonymity; therefore, anonymity for more latency-sensitive purposes must use other solutions.

3.4.2. Early related designs

Various anonymous communication systems utilised concepts from the original Mix-nets design, introducing certain modifications in order to address certain aspects of anonymous communication. Many of these systems were associated with large latencies associated with the Mix-nets architecture that made it impossible to use them in more time-sensitive scenarios like web browsing. For these systems we can mention, among others, Babel, an anonymous remailer described in 1996 [19] which aimed to quantify dimensions of anonymity providing a way of measuring it, as well as enhancing the network security. Another example was a paper about Stop-And-Go MIXes (SG-MIXes) [20] that introduced random delays in order to be more resilient to traffic analysis and remove the requirement of collecting a fixed-size batch of messages. ISDN-MIXes [21] is another Mix-net-based design from 1991 for untraceable communication for services like telephony. One of its major advantages is a very small bandwidth overhead. The first successfully deployed Mix-net-based solutions that gained significant recognition were anonymous remailers. In 2000 Web MIXes system [22] was proposed as one of the first described solutions for Internet browsing utilising Mix-nets. The project was successfully deployed and gained recognition. After initial Mix-nets designs, where mixes were treated as specific servers to whom clients connect, propositions to make peer-to-peer designs occurred where each user also relayed traffic for others, acting as a mix in the network. Within this category we can name solutions like MorphMix [23] or Tarzan [24].

3.4.3. Anonymous remailers

Anonymous remailers were the first real-life implementations of Chaum's ideas that gained significant recognition. There are three types of anonymous remailers, or four, if pseudonymous remailers that were described before are counted.

Cypherpunks remailers

The first anonymous remailers, also known as Type I remailers, were initially created by the Cypherpunk movement. Cypherpunk remailers can only be used for one-way communication as there is no possibility for the recipient of the message to reply. Cypherpunk remailers, as the whole Cypherpunk movement, attached huge importance to the cryptographic aspect of communication; therefore, messages were encrypted in order to provide confidentiality, usually with the software from the PGP family. Another important concept that was introduced with Type I remailers was chaining, meaning that instead of one remailer that will be more than intermediate in the connection between the sender and the receiver, getting rid of the single point of failure issue and increasing the anonymity of the sender, as the second and every next remailer in the path does not know anything about the sender's identity and the first remailer does not need to know the identity of the sender. Contrary to the Penet remailer, Type I remailers do not keep a list of users nor any activity logs. One of the greatest weaknesses of cypherpunk remailers is their vulnerability to traffic analysis and therefore correlating messages with the sender by the order of them. Cypherpunk remailers gave the foundation for other types of anonymous remailers.

Mixmaster

Mixmaster, also known as Type II remailer, is another type of anonymous remailer that, similarly to the Type I, allows for one-way anonymous communication, unless, of course, the sender himself includes a reply address in the message itself. It was created by Lance Cottrell around 1995 and later maintained by Len Sassaman and Peter Palfrader. Mixmaster operates on fixed-size packets, and one of the greatest improvements in comparison to the Type I remailer is message reordering (mixing) that prevents the possibility of traffic analysis. Mixmaster sends messages through the intermediary nodes via the SMTP protocol.

Mixminion

Mixminion, also known as Type III remailer, further addresses the issues related to its predecessors. Similarly to Type II remailers, it uses fixed-size packets that are reordered and forwarded with appropriate encryption through a chain of servers called mixes with their public keys. A single mix cannot correlate the sender with the recipient. In order to reply to messages, Mixminion utilises Single-Use Reply Blocks, also called SURBs, that are a way of anonymous responses. They are hidden partial paths included in the message that give the possibility of replying without revealing the sender's identity. Each mix in the chain can unwrap its layer and encrypt the message. The initial release of Mixminion was in December 2002, and one year later the paper describing the design was published [25].

3.4.4. JAP

JAP is an anonymous communication network known under many names: Web MIXes, Java Anon Proxy, JAP, JonDonym, JonDo. Despite the naming confusion, all of them refer to a proxy system designed for providing anonymity on the Web. The system was first described in 2000 in a paper "Web MIXes: A system for anonymous and unobservable Internet access" [22]. Later, the project was deployed as a part of the AN.ON project of the Technische Universität Dresden, the Universität Regensburg, and Privacy Commissioner of the state of Schleswig-Holstein under the name Java Anon Proxy (JAP), which was directly taken from one of the system components. In 2007, the project diverged into for-profit JonDonym (AN.ON substitute) and the proxy itself got the name JonDo (JAP substitute) and the entity responsible for the commercial branch development was JonDos GmbH. Regardless of these administrative and name changes, the system will be treated as one within this paper. It initially supported only HTTP and HTTPS traffic, although it was later expanded for other protocols, and had several interesting technical solutions that made it suitable for web browsing even though it utilised high-latency architecture of Mix-nets. Key components of the WebMIXes include:

- Java Anon Proxy client - the client software written in Java in order to achieve platform independence. It connects to the first MIX of the MIX network through the Internet. It periodically sets up dedicated bidirectional channels for data transfer. If a user wants to send a message it is first transformed into the MIX-format and passed through the MIX cascade. If a user wants to send a smaller message than a given size, it is padded to avoid volume correlation. On the other hand, if he wants to send a large message or streaming data, the message is splitted into several time-slices and it is sent slice by slice with an appropriate Slice ID (SI). The idea of slicing was first introduced in the previously mentioned ISDN-MIXes. If there is nothing to send by a given client, a dummy message is sent. An adversary that wants to correlate a message with the specific sender is unable to do so as with equal probability it can be any of the active clients in a given moment and he is unable to tell the difference between the real message slice and a dummy message as they are both encrypted. As the channels are bidirectional, it is also responsible for handling the responses. Messages that are sent via JAP are also filtered in respect of potentially compromising content like cookies and JS. On top of that JAP also utilises Information Service (info-service) in order to measure anonymity. Each user has limited bandwidth and the number of concurrently used time slices. Each time he wants to send a slice he has to provide a ticket. The ticket-based authentication system uses the blind signature algorithm invented by David Chaum [26] in order to preserve users' privacy. This authentication mechanism aims to prevent flooding and DoS attacks.
- MIXes cascades - dedicated servers connected with each other through the Internet, although authors in the paper suggest that ideally they should be connected with dedicated high-speed connections instead, however it would be too complex and expensive. The chain of MIXes that mediates between the sender and the receiver is called MIX-cascade. Similarly to the original

MIXnet design each MIX node strips a layer of encryption, creates a batch of messages and reorders the messages and forwards the data to the next MIX. The last one in the cascade sends it to the cache-proxy. MIXes, similarly to JAP, also send dummy traffic to each other.

- Cache Proxy - an element responsible for sending the data from the last MIX to a specific web server that the user requested. It sends back the response through the same bidirectional channel, although in a reverse order. It also loads all objects embedded into the requested HTML page. This idea was first introduced in the Crowds project. Moreover, as the name suggests, it is responsible for caching the responses which helps to reduce the number of calls to the Internet.
- Information Service - a service providing information about MIXes' addresses and public keys, current traffic and MIXes availability and the provided level of anonymity which is calculated by taking into account the number of currently active users which is directly associated with the possibility of the intersection attack. Each user can set a threshold of anonymity he wishes to maintain and once the level falls below that threshold he will be warned.

The WebMIXes project underlined the importance of the usability perspective and the ease of configuration and use as well as the portability of the JAP software so that it could have been run on as many diverse devices as possible. It also provides a built-in integration with web proxies that helps to work behind restrictive proxies, firewalls, etc. in order to provide better accessibility. In JonDonym users also had the possibility of choosing Mix Cascades in order to avoid organizations that they do not trust. In the past it used to be one of the more recognised ACNs, although as it is not functioning today it cannot be considered within the comparative analysis.

3.4.5. Mixing and batching strategies

Several strategies of mixing and batching messages were introduced in various papers and projects. One of the first attempts to provide a formal framework of modeling mix networks based on the batching and mixing strategy was proposed in 2003 [27]. It was based on the 2002 paper [28] where certain types of mix networks were distinguished and categorised along with attacks on them. Those strategies included:

- threshold mixes: the mix collects a certain numbers of messages and then forwards them,
- timed mix: the mix forwards messages after a specific time period,
- threshold or timed mix: the mix either collects a certain numbers of messages and then forwards them or forwards messages after a specific time period,
- threshold and timed mix: the mix collects a certain numbers of messages and forwards them after a specific time period,
- threshold pool mix: when $n + f$ messages accumulate in the mix, where n is a threshold and f is a pool size, n randomly chosen messages are forwarded while f messages are kept in the pool for the next iteration,
- timed pool mix: every specific period of time messages are sent, however a fixed number of

random messages (a pool) are kept in the mix,

- timed dynamic-pool mix (Cottrell Mix): similar to the timed pool mix combined with threshold pool mix, however only the certain fraction of messages surplus, after the pool subtraction, are forwarded every specific period of time.

3.4.6. Message format

Over the years specific message formats for Mix-nets were proposed in order to improve security, for example, by hiding metadata and routing information which helps break linkability between incoming and outgoing messages or protecting against attacks like the tagging attack where the attacker modifies messages by embedding tags that he wants to detect in other parts of the network in order to get routing information. Minx [29] was one of the first propositions to address these issues. The format uses RSA for the standard Mix-net encryption along with anonymous reply blocks, AES with IGE mode that helps to detect message manipulation and, as the IGE mode propagates errors, destroys the modified message by changing the content into garbage. At the same time, the format imposed a low computational and communication overhead. Minx has a vulnerability which was discovered in a 2008 paper [30]. The paper, besides describing the vulnerability and a chosen ciphertext attack that exploits it, provides a fix for the Minx format that would address this issue by replacing explicit session key and next hop header fields by utilising hash functions. Yet another message format - Sphinx - was proposed in 2009 [31]. The goal was to provide the most compact and yet fully secure scheme. Sphinx addressed issues of the Minx design at the same time keeping all the desirable properties of it. One of the changes was using ElGamal instead of RSA. Sphinx also includes replay protection and formal proofs of security, which Minx lacked.

3.4.7. Topologies

There are several possible topologies in the Mix-nets that appeared throughout the years. These topologies were distinguished and analysed in terms of anonymity and overhead in 2010 [32]. The first topology that appeared in the original Chaum's 1981 paper was a cascade topology where mixes create a simple path where each node is connected only with its predecessor and successor (excluding the first and last nodes). Due to the unlinkability property of Mix-nets, the cascade can be fixed and we do not need permutations of paths as in the onion routing that will be described later. The drawback of this topology is its poor scalability and availability, as each mix in the path is a potential point of failure and there is a high risk of bottlenecks. Instead of a single cascade, there is also a possibility of a topology where there are many cascades - it is called multi-cascade topology. One of the alternate topologies is a fully connected network, where each node is connected with any other node in the network. It fixes the availability issue as there are many potential routes from one mix to the other. It does not fix the issue of scalability as adding a new node implicates the necessity of adding a connection with every other node in the network. The drawback of this topology is the fact that messages cannot be freely

mixed as there are several possible directions of a message from the node. Another possible topology, mostly used in the latest Mix-nets, is a stratified topology. Each node is assigned to a certain layer and it is connected to every other node from the previous layer and with every other node in the next layer; however, the direction of the messages in the network is fixed. With this topology, the scalability and availability are improved and all messages in the node can be mixed together regardless of their origin and destination, contrary to the free route design.

3.4.8. Loopix

The Mix-net-based systems development continues to this day, although it has definitely less popularity than onion routing-based solutions. One of the newer propositions in the MIX world is the Loopix system, proposed in the 2017 paper [15]. It utilised ideas from Stop-And-Go MIXes, creating a simplified version of it called Poisson mix. The messages do not use synchronous rounds as with some different Mix-net designs; they are asynchronously and non-deterministically delayed using exponential distribution. Messages arriving at a given entity within the network follow Poisson distribution. As a solution for a smaller user base and increased security, cover traffic is introduced between the entities in the network, including so-called loop traffic. Loop traffic, whose concept is similar to the 2003 paper [33], where a user sends messages to himself, helps to detect active attacks and blocking of messages and, along with drop cover traffic, aims to hide communication patterns. Loopix provides a modifiable trade-off between anonymity and usability by changing message delays and the real to cover traffic ratio. Loopix uses providers to manage access and store offline messages. These are semi-trusted nodes that users connect through. The Loopix mixes create a stratified topology, meaning that they are grouped in layers and each node is connected to every node from the previous layer and every node from the next layer. When a user wants to send a message, he randomly chooses a node from each layer. The messages sent in the Loopix network follow the Sphinx format. Loopix opened a new chapter in mix networks as it serves as a basis for other Mix-nets that are created to this day. Nym [16] being an example of such a network.

3.4.9. Nym

The Nym Network was described in 2021 [16]. The network is based on the Mix-nets architecture and heavily utilises concepts from the Loopix network and can be considered an implementation of Loopix, enhanced with economic incentives. Mixes in the network are rewarded with NYM tokens using a mechanism called "proof of mixing". The tokens provide economic incentives which are not present in other volunteer-based networks and aim to encourage more people to host mix nodes. One of the network's main goals is to introduce a global adversary-proof solution which is not present in solutions like Tor. Besides a decentralised Mix-net, Nym includes an anonymous credential cryptosystem. Nym network architecture consists of the below elements:

- mix nodes - nodes that create a network called Mix-net that relays, decrypts its specific layer of

encryption, reorders and delays messages asynchronously as in the Loopix network

- gateways - entry points for the Nym network. Users are able to choose whether they want to always use the same gateway or not. Similarly to the Loopix design, they are also able to store messages for users that are offline
- service providers - entities that provide certain services through the Nym network. They receive cover traffic that is indistinguishable from the real users' traffic, which enhances users' privacy
- Nym blockchain - decentralised blockchain structure, maintained by validators, that distributes network-wide information regarding public node reachability and public keys, configuration parameters and other publicly available data that is necessary for proper functioning of the Nym network
- nympool - shared pool of funds supporting the Nym network. Periodically validators algorithmically distribute rewards from the nympool to the nodes (validators, gateways, mixes) and to the service providers that redeem users' service credentials.
- validators - Validators maintain the Nym blockchain, including nympool, as well as issue bandwidth and service credentials that proves allowance of sending data through the Mixnet and the right to accessing a certain service, respectively.

The flow of users connecting to the service provider in the Nym network works as follows: A user deposits his NYM tokens - earned by operating a node, purchased, or received from a service provider - in the nympool and obtains credentials with the help of the validators. These credentials cannot be directly linked to their use thanks to blind signatures. With the bandwidth credential, the user can access the gateway of his choice. The gateway validates the token in the Nym blockchain and submits a commitment of the credential to the validators in order to avoid double-spending. The user and the gateway establish a shared secret for data encryption and a pseudonym associated with the credential that allows for tracking bandwidth consumption as well as for offline storage. From now on, until the bandwidth credential expiration or consumption, the user can utilise the Nym mixnet through the gateway. While the gateway has the knowledge of when the user is accessing the network, it is not aware of where his messages are sent. The message from the user is sent to the recipient's gateway, from which the recipient can pull the messages. The recipient is either a service provider, a validator, or another user. If the recipient is a service provider that provides a paid service, a service credential is needed to access the service. The logic behind handling service credentials is similar to the bandwidth credential's one.

3.4.10. Other designs

In recent years, other Mix-nets designs appeared. Katzenpost [34] being one such example, aims to fix certain issues associated with the Loopix design. It puts a great emphasis on implementing post-quantum cryptography algorithms into the mix network, as well as changing a point of failure in the form of a single service provider into scattered distribution across multiple providers, similarly to a

distributed hash table form. The project is still in development. Recently, the design was described in a paper "Echomix" [35], where Katzenpost is defined as the Echomix implementation.

Besides Loopix-based designs, there are also unrelated recent approaches to the topic of Mix-nets, including Vuvuzela [36] that aimed to minimise metadata related to the communication process, Riffle [37] that aimed to decrease bandwidth and computation costs for the sake of file sharing and microblogging, or cMix [38] that reduces public key cryptography-related latency and the overhead for the client by utilising precomputation. The cMix uses a threshold and timed mixing strategy, which may potentially still introduce significant latency, especially for small numbers of users. Moreover, fixed cascade topology suffers from the issues described in the topology subsection, for example, reduced availability. The paper suggests a multi-cascade approach as a potential answer for this issue.

3.5. DC-nets

3.5.1. Original design

DC-nets are anonymous communication networks that are based on the dining cryptographers problem that was described by David Chaum in his 1988 paper [10]. Despite the name, the problem is not related to the dining philosophers problem. It considers the problem of secure multiparty computation of the boolean XOR function. We can describe the problem as follows: assume three cryptographers are dining together. The waiter informs them that their meal has already been paid for by someone, but he does not reveal the identity of the payer. We assume that it can be either one of the dining cryptographers or the National Security Agency (NSA). In order to determine whether it was one of them who paid or the NSA, they have to perform a certain two-stage protocol. In the first stage, every pair of cryptographers establishes a one-bit secret. For three dining cryptographers, we have three such secrets: AB, AC, BC. In the second stage, each cryptographer that did not pay announces the result of an XOR operation on the secrets he shares with his two neighbours. On the other hand, if a certain cryptographer paid, he announces the opposite of this XOR operation. After three announcements are revealed, it can be determined whether it was one of the cryptographers who paid or was it NSA. It can be done by performing another XOR operation on these three bits. If the result is zero, or in other words, if the number of ones is even, it means that it was one of the cryptographers who paid for the meal. If the result is one, it means that NSA paid. There are several limitations to this problem:

- only one cryptographer can pay for the meal,
- each cryptographer is trustworthy, meaning he always presents information that is true,
- each cryptographer must share a secret with every other participant, meaning that will be in total $n * (n-1)/2$ connections, creating a complete graph.

The proposed protocol to solve the dining cryptographers problem can be easily converted to the anonymous communication of longer messages. While sending messages bit-by-bit might be problematic, it can be replaced with fixed-size blocks and the sender, instead of performing a negation of the XOR operation he performed on the secrets he shares with his neighbours, performs an XOR of

these secrets with his message. The message is broadcasted by design, however, certain improvements can be made in order to provide confidentiality and authenticity - public-key cryptography can be utilised where each participant can have a public key associated with a pseudonym. The sender can include a digital signature to his message in order to prove the authenticity of the message. Moreover, in order to provide confidentiality (or secrecy), he can encrypt this message with the public key of the recipient he wants his message to be transferred to. Each network member will receive the encrypted message; however, only the recipient - the owner of the private key - will be able to decrypt it. Chaum also proposes other potential improvements to the design, like slot reservation. The original DC-Nets design aimed to demonstrate the dining cryptographers issue and sketch potential implementations of them. However, they cannot be used in the original form in real use cases due to the issues associated with this simple design. Similarly to Chaum's Mix-Nets, his DC-Nets introduced a new branch of ACNs, creating a basis for the future work of other researchers.

3.5.2. Related designs

Among systems or propositions that were based on the DC-nets, that similarly to the Mix-net-based solutions were meant to address some issues from the original design, we can mention The Dining Cryptographers in the Disco protocol [39] and CliqueNet [40], a peer-to-peer, practical, self-organising, resilient, scalable and anonymous communication network that was proposed in late 2001 and aimed to fight surveillance from solutions like Carnivore [41]. CliqueNet evolved into the Herbivore [42] network; however, the goals remained mostly the same. Another example of DC-net based design with efficiency improvements and multicast focus was described in the Xor-Trees paper [43]. A notable paper about improvements to the DC-net design - among others in terms of jamming detection, efficient fault recovery and performance - was proposed in "Dining Cryptographers Revisited" [44]. Dissent (Dining-cryptographers Shuffled-Send Network) [45] aimed to address disruptions caused by DoS or Sybil attacks by introducing a slot distribution mechanism via accountable shuffling. Its follow-up paper titled "Dissent in Numbers: Making Strong Anonymity Scale" [46] addressed scalability issues with the original Dissent design through the introduction of client-server architecture, where each client trusts that at least one server is honest. Riposte [47] further enhances this concept with Private Information Retrieval via Distributed Point Function, which reduces the high bandwidth cost associated with DC-nets-based designs. At the time of writing this thesis, there is no widely deployed and fully operational DC-net-based solution.

3.6. Anonymous publication systems

Anonymous publication systems are a category of systems that have the specific purpose of anonymous publishing of content, which is usually joined with the goal of censorship-resistance. They can be either created on top of an existing anonymous communication network or have dedicated mechanisms. As these networks are not universal in terms of networking and have a specific purpose,

they will not be considered within the comparison; although they are worth mentioning as they had an important role in the development of ACNs that are known today. One of the first proposals of such a system was The Eternity Service [48]. It was a proposal of a storage system with long-term availability of files in it. It assumed utilising redundancy and scattering and emphasised the importance of the anonymity of servers that were serving content. For anonymous communication utilising Chaum's Mix-nets or DC-nets was proposed. Adam Back, a British cryptographer, implemented the Eternity Service [49] that was inspired by the original paper and its goal, although the technical design was different. Another proposal for anonymous WWW publishing was described in the paper "TAZ servers and the Rewebber network: Enabling anonymous publishing on the world wide web" [50]. The authors underlined the importance of anonymous publishers throughout the history of publication. They provided a more practical solution to the issue of anonymous publishing. The Rewebber acts as a proxy when a certain file is accessed and the user sees only the Rewebber's domain, the path is encrypted, therefore no information about the TAZ server location is revealed. The Rewebber forwards the request to one of the TAZ servers. The data in the system is stored on TAZ servers in a scattered way. The Rewebber network can provide availability in case of one of the TAZ servers' failure. A similar system was proposed by The Free Haven project [51]. It aimed to provide anonymity for publishers, readers, and senders, accountability without sacrificing anonymity, persistence and robustness of the storage, and flexibility of peers joining and leaving the network. One more example of a system in this category was the Publius [52]. Nine goals of the network were: censorship resistance, tamper evidence (unauthorised changes can be traced), anonymous source, updatability, deniability (third parties that participate in the publishing are not responsible for the content itself), fault tolerance, persistence, extensibility, and free availability. System was implemented and opened for a two-month trial, however it did not gain enough prominence to continue development. A further example is Tangler [53], a censorship-resistant system that employs a storage mechanism where newly published documents depend on the previously published documents, similarly to the blockchain where each block depends on the previous. Publication of new documents is associated with replication of previously published ones. This dependency is named by the authors as entanglement. Today, the most recognisable systems from this category are Freenet and GNUnet; although GNUnet greatly evolved compared to the original file storage functionality.

3.6.1. Freenet

Freenet was created in 1999 as a university project of Ian Clarke who studied at the University of Edinburgh. The paper describing the network was published in 2001 [54]. The goal was to create a fully decentralised, peer-to-peer system for censorship-resistant and private communication and publishing. Freenet was further developed and maintained by a group of internet freedom activists. It has gained great popularity, as it has been downloaded more than two million times since the release of the system. In March 2023, Locutus was renamed to Freenet, and the project that up to that point worked under the name Freenet was renamed to Hyphanet. It is worth mentioning that maintainers of the original Freenet

were not satisfied with the decision of rebranding. In March 2023, Locutus was renamed to Freenet, and the project that up to that point worked under the name Freenet was renamed to Hyphanet. It is worth mentioning that maintainers of the original Freenet were not satisfied with the decision of rebranding. For the sake of clarity, this paper will use the names Freenet and Hyphanet interchangeably and Locutus when talking about the new network. The main goal of Hyphanet is to provide a way to anonymously share files, browse and publish Hyphanet websites and chat on forums without censorship. Ian Clarke emphasised the importance of freedom of speech and the fact that if some institution has a possibility to control the information that people have access to, then the institution can influence peoples' opinion. Hyphanet therefore aims to allow two or more people to share information freely, without government or other entity's control. At the same time, Hyphanet assumes that it is impossible to have freedom of speech without being anonymous, as it is easier to punish those who proclaim things that are not appealing to the censor rather than preventing them to do so. Hyphanet also proposed solutions to the problem of trust in the anonymous system. The goal of Locutus [55] is slightly different than Hyphanet's one; however, it shares some similarities as well. As was described before, Locutus wants to address "modern challenges", which in essence means to provide an alternative to the Internet that we know today, which is a very ambitious goal. Ian Clarke, in his talks presenting Locutus, presents the history of the Internet along with the history of its predecessor - ARPANET. He especially emphasises the fact that ARPANET, due to the fact that it was designed in such a way that it should be resilient to potential nuclear attack, was decentralised. On the other hand, the Internet that we know today is not only highly centralised when it comes to the widely spread client-server architecture, but also it is mostly controlled by a few corporations that own most of the Internet services and infrastructure. It poses a threat of data exploitation or censorship and therefore can violate individual freedom. Locutus is proposed as a solution to these problems. It works as a shared computing platform. While Hyphanet can be compared to a shared disk, Locutus can be compared to a shared computer. Locutus can be reached using standard web browsers as well as with an API. It proposes interesting solutions regarding many problems of today's Internet, like users' privacy, spam, or DDoS attacks. Locutus is designed to be a platform on top of which developers can build highly interoperable, seamlessly integrated, scalable and cryptographically secured decentralised services that can be an alternative to the centralised ones that we know today.

3.6.2. GNUnet

GNUnet started in 2001 and its main goal back then was to improve Freenet's file sharing. The initial name for the system was GNET as GNU had not yet approved the project back then. The principle of operation was described in the paper with the same name [56], describing a reliable, anonymous distributed backup system. After the approval a few months after the release, the project was renamed to GNUnet as was intended since inception. The communication mechanism in the GNet on the lowest layer was similar to SSH, although it was based on UDP, not TCP as it did not need

guarantees of proper order or packet loss and preferred to utilise lower-overhead solution. The project also described a reputation system that affected a node's ability to interact with the network in order to prevent various attacks and protect the network. Instead of a global credibility system, each node had an "opinion" on other nodes that it contacted that decreased with every quest and increased for the valid replies. Queries were encrypted and hashed by the client in order for intermediaries not to be able to decrypt the content. Hashcodes could have been combined with logical operators in order to increase the searchability over systems like Freenet. Users can decide whether to use keywords of shorter length and improve usability or of longer length and increase security. Data stored in the GNet's nodes was split into chunks and each chunk was by default encrypted and stored separately with hashes as chunks' ids. In an alternative configuration, the data can be shared in plaintext and encrypted only in transfer or, in the maximum-security configuration, XORed with a one-time pad which results in doubling traffic/storage requirements. The project greatly evolved through the years and now GNUnet is a framework for p2p networking in a decentralised and privacy-preserving manner. It aims to serve as an alternative for the traditional TCP/IP stack, creating many of the networking solutions from the ground up, addressing their major flaws from the point of view of privacy, freedom and decentralization and at the same time providing excellent plugin-based compatibility with existing infrastructure. As the anonymity is no longer the main focus of the project, it will not be considered within the comparison.

3.7. Onion routing

While Chaum's Mix-nets provided a satisfactory level of anonymity, most of the solutions based on them introduced significant latencies and could not be used for latency-vulnerable use cases. One of the first proposals that aimed to address this issue was the ISDN-MIXes network, although it was mostly focused on the ISDN network. A different and more universal approach was PipeNet [57], a protocol for low-latency anonymous communication. It was based on a network of nodes, identified by their public keys, that asynchronously communicated with each other over secure links, creating a mesh network topology. The main drawback of the network was that it assumed fully utilised links, which are expensive. Another proposition was the UDP-based Freedom System [58]. The primary goal of Freedom was to create a system with the strongest privacy, ease of use, and integration. Freedom utilised a circuit-based solution named the telescope encryption. The solution that turned out to be the most used for low-latency anonymity communication was onion routing.

3.7.1. Original design

In 1995 the U.S. Naval Research Lab (NRL) researchers started working on the first prototypes of onion routing. The first paper about the Onion Routing was released in 1996, however the final version of it was first published in 1997 and a year later, in 1998, it was published in IEEE Journal on Selected Areas in Communication [11]. Onion routing enables anonymous connections that are highly resistant to surveillance, tracking and traffic analysis. Onion routing provides a near real-time

bidirectional connection similar to the TCP socket connections. Regular Internet applications can access anonymous connections via proxies, which additionally can remove identifying information from the data stream and therefore make the communication anonymous. It is the major difference between the Mix-net and the Onion routing as the mixes are not connection-based. It has several implications, for example onion routing does not batch the traffic and the message reordering is highly limited, which may make it more vulnerable to certain traffic analysis attacks compared to the classic Mix-nets. Nonetheless, the versatility of onion routing joined by better user experience through smaller latencies made this solution a dominant one today. Applications that initialise communication make it through a few machines called onion routers. Onion routing was designed to be application independent, therefore a wide variety of applications could have been used. The onion routing network allows initiator and responder to have a connection between them. These connections are called anonymous socket connections, or simpler, anonymous connections. They hide who is connected to whom and why. If we would like to go one step further and provide anonymity to the entities themselves, then we have to remove identifying information from the data stream before we send it over the anonymous connection. Basic configuration of the onion routing topology assumes that there is an onion router that sits on the firewall of the sensitive site. There is an assumption that connections within the sensitive site are protected by, for example, physical security, therefore we can also call it a secure site. Sensitive sites may be only on the initiator's end or on both ends of the connection. Onion router at the edge of the sensitive site serves as an interface between machines within the secure site (behind the firewall) and the external network like the Internet. In order to avoid traffic analysis, the onion router sitting on the initiator's edge should route its traffic through other onion routers within the onion network. If both sites control onion routers then they can successfully hide their communication from eavesdroppers. On the other hand, if the responder, for example Web server, has an unprotected connection between him and the last onion router, then the data stream from the initiator must be anonymized. Otherwise analysis of the unprotected link may reveal the initiator's identity. Onion routers within the network are connected by permanent socket connections. Anonymous connections over the network are multiplexed over these permanent connections. For any given anonymous connection, the route in the path is precisely defined at the connection initiation, however each router in this path is only aware of the previous and the next hop in the route. The data transmitted through the onion network is modified at every onion router in order to prevent tracking of data and ensuring that compromised onion routers cannot collaborate by correlating the data stream they observe. Onion routing's connections by design have three phases: connection setup, data movement and connection teardown. In the first phase an initiating application establishes a socket connection to the proxy specific to an application on an onion router which intermediates its connection to the onion routing network. The proxy then determines a path through the onion network and creates a layered data structure known as an onion and sends it through the network. Each layer specifies the next hop in the route, when an onion router receives the onion it removes the layer specific to itself, identifies the next hop and forwards the remaining

onion to the next onion router. According to the design each onion router should pad the onion in order to maintain a fixed size. After sending the onion, the proxy on the initiator's site can send data through the established anonymous connection. The last onion router in the path forwards data to the proxy on the same machine, called the responder's proxy which forwards data to the responder. Each onion layer includes properties of the connection at each point in the path, for example cryptographic algorithms that will be used in the data movement phase or key seed material necessary for generating symmetric keys that will be used to encrypt the data sent forward (direction in which the onion travels) or backward (opposite to the onion travel's direction) along the anonymous connection. 1999 design assumes usage of different algorithms and keys for the data that moves backwards compared to the one moving forwards. Symmetric cryptography is used for this purpose as it has significantly less computational overhead than expensive public key cryptography, therefore it is more suitable for near real-time connections. As was mentioned, after establishing connection by sending the onion in the initialization phase the connection is ready to perform data movement phase. Sender's onion router will add a layer of encryption for each onion router in the path. Each onion router in the route will remove one layer of encryption and the last onion router will have a plaintext which it will pass to the receiver. For the response we use the reverse order of these layers. Namely, the last router in the path will add the first layer of encryption and the first router will add the last one. Then the sender will receive the response encrypted with multiple layers which then he can remove in order to obtain the plaintext. The 1999 paper describes that all information, including the onion from the initialization phase, network control information, as well as the data that is being sent in the data movement phase, is sent in the uniform-sized cells. All cells that the onion router receives within a fixed time interval are mixed together in order to reduce correlation of the network data. For the purpose of foiling external observers, padding and bandwidth-limiting can be used. Due to the fact that each onion and each data phase cell looks different to each-onion router it is significantly harder for the attackers to perform traffic analysis. Although the name "onion routing" may suggest that it has something to do with the third (network) layer of the ISO/OSI model, the operation logic occurs in the seventh (application) layer. It's an example of an overlay network so it refers only to the logical connections, not the physical ones. Moreover there is a significant flexibility when it comes to the protocols of the lower layers. Authors describe onion routing's anonymous connections as protocol independent. Onion routing turned out to be a breakthrough in the anonymous communication network field and many designs to this day implement this concept. The best known example of such a network is Tor - the most popular ACN today that will be described in a dedicated section. Onion routing introduced an idea of reply onions, which idea is similar to Mix-nets' reply blocks. It is the way to connect with the originator in other ways than using initial bidirectional connection with him, for example for the sake of asynchronous communication.

3.7.2. Tor

History

In the beginning of 2000s Roger Dingledine and Paul Syverson, shortly later joined by Nick Mathewson, worked on an implementation of onion routing that could be used for everyone, not just U.S. Navy. It was designed to work in a decentralised network, run by people and institutions with varied interests and levels of trust. Roger Dingledine named the project Tor, which was an abbreviation for The Onion Routing, in order to differentiate between other onion routing-related projects that were developed at that time. In 2002 Tor network was deployed and its code was released under a free and open software licence in order to provide more transparency to the project. In 2004 a paper "Tor: The Second-Generation Onion Router" [13] was published, which described how Tor operates. In the same year Electronic Frontier Foundation (EFF) began to fund the work on Tor. In 2006 Tor Project Inc, a non-profit organisation was created in order to maintain development of the Tor network. Nowadays Tor is funded from several sources, including volunteers, foundations and institutions. Tor stood the test of time and turned out to be an exceptionally useful, free and easily accessible tool against censorship, surveillance and much more. Today the Tor network has thousands of voluntary-runned relays, millions of users and it is undoubtedly the most popular anonymous communication network in use.

Modifications compared to the original onion routing design

Tor introduces a few modifications to the original onion routing design. Costly, slow and compromise-prone public key cryptography with long-living keys was replaced with short-living symmetric keys, achieving a perfect forward secrecy, as it will be impossible to decrypt captured traffic once the keys are deleted with the circuit teardown. Separate, dedicated proxies for every application were replaced with a universal SOCKS proxy that supports most TCP-based applications and for the HTTP and HTTPS Tor relies on Torbutton, an add-on for Firefox, and hardened version of the Mozilla Firefox, creating a Tor Browser Bundle. Ideas of mixing, padding and traffic shaping were rejected in favour of performance improvements and the quality of experience. Many TCP streams are multiplexed within a single circuit, which improves both efficiency and anonymity, as building many circuits can present a threat to anonymity, which will be discussed later. Tor proposed a leaky-pipe circuit topology, meaning that a client can communicate with all nodes in the circuit, which in theory may make traffic analysis more difficult as the data is not always going through the entire circuit. Congestion control was achieved thanks to end-to-end acknowledgements that detect congestion. Problematic state flooding was replaced with another mechanism - directory servers; certain nodes, considered as more trusted, provide a list of known routers and their state in order for the user to be able to build a circuit. Initially such a request was made in HTTP, however it was later exploited by censoring regimes that were able to block Tor network for users, therefore the connection between the user and the directory servers started to use HTTPS. More information about the history of blocking Tor across the world will be

described in a dedicated section. Tor nodes can advertise exit policies that describe hosts and ports to which a node is allowed to connect. End-to-end acknowledgements were introduced, enhancing security and preventing certain types of attack, i.e. tagging attack. Reply onions were replaced with a more sophisticated mechanism of hidden services that allows for a connection with a server whose location is not known. In order to connect with a hidden service, clients negotiate rendezvous points. Initially Tor did not try to hide the fact that a certain user is using the network - it changed over time when regimes started to censor Tor and it was necessary to fight back - the censorship resistance became another goal of the Tor network. Today Tor circumvents censorship with bridges that are not publicly listed relays, along with pluggable transports - a way of masking traffic.

Design and architecture overview

Tor does not protect against a strong global passive adversary and therefore it is not secure against end-to-end traffic correlation attacks. It can protect against local passive adversaries that observe a certain fraction of a network, against generation, modification, deletion and delay of traffic or against operating or compromising some number of the onion routers. The main reason for sacrificing stronger anonymity is to achieve a compromise between anonymity, usability, and efficiency. The project puts a great emphasis on deployability by inexpensive and easy setup, portability by supporting various operating systems, expandability for future improvements and simplicity for the sake of the ease of understanding and analysis. The main goal of these actions was to increase usability - the more people will use Tor, the larger anonymity set will be and therefore the harder it will be to identify a single user. Tor utilises a client-server architecture. Each onion router (OR) in the onion network runs a non-root process which allows it to maintain a TLS connectivity with every onion router in the network. Users run a process called an onion proxy (OP) which allows them to establish circuits in the network, manage user connections and fetch directories. Multiple TCP streams from onion proxies are multiplexed across a single circuit. Onion routers maintain two kinds of keys: short-term onion keys and long-term identity keys. Identity keys are used for TLS certificate signatures, signing a structure called a router descriptor, which is a summary of router's details like address, bandwidth, policies, keys etc. On top of that the directory servers use identity keys to sign directories. Onion keys on the other hand are used in circuit setup and symmetric ephemeral keys negotiation. TLS protocol also establishes a short-term key during onion routers' intercommunication. Short-term keys are rotated periodically and, in order to reduce a potential key compromisation impact, independently. Initially Tor design assumed random choosing of nodes in the circuit. This approach turned out to be problematic for various reasons. The first one was a fact that it created bottlenecks as each node had the same number of circuits as any other node and therefore the low-bandwidth ones were overloaded, the high-bandwidth ones were underutilised. The solution for this was to weight a node based on this bandwidth in a way that the high-bandwidth nodes get more circuits than the low-bandwidth ones, as well as to balance based on the node's capabilities as the nodes that can be only the middle node would be chosen three times less frequent compared to

a node that can be either entry, middle or exit node, which would also lead to bottlenecks. To prevent abuse by a potential attacker, who would advertise an impossibly high bandwidth in order to have a dangerously high probability of choosing him as a node in the circuit, Tor uses measured bandwidth instead of an advertised one. Another improvement that was made with nodes choosing was related to guard nodes. It is assumed that it is sufficient for the attacker to deanonymise user if the attacker is controlling both entry and the exit node. Considering the fact that the circuits are created rather frequently, the probability of deanonymisation in a longer period of time was exceptionally high. Guard nodes are an answer to this issue - instead of choosing three new nodes each time the circuit is created, the entry node is kept for a longer period of time. With this approach, the likelihood of denonymisation is not increasing as rapidly with time, unless the malicious node is chosen for the guard - then the probability is even higher. Moreover, in order to decrease the likelihood of controlling both entry and exit nodes at the same time, additional diversity requirements were introduced for nodes in the circuit that prevents the creation of a circuit with two routers with the same network range or family.

Cells, circuits and streams

Tor uses fixed-size cells of 512 bytes for communication over TLS connections between onion routers and onion proxies. Cell includes the identifier of a given connection, command field that specifies a cell's payload purpose and the payload itself. There are two kinds of cells: control cells, interpreted by the receiving node, and relay cells, responsible for end-to-end data stream. Relay cells have additional header fields compared to control cells, including: TCP stream identifier, end-to-end checksum, payload length and a specific relay command. Relay header is encrypted along with relay cell payload and decrypted along the way in the circuit. In Tor one circuit can be shared among multiple TCP streams. Moreover, in order to avoid correlation between the Onion Proxy and the stream, a new circuit is being established periodically in the background, when the previous circuit is being used. Circuits that do not have any open streams are closed. Every minute Onion Proxy considers switching the circuit to the new one. The Onion Proxy negotiates symmetric keys with every node in the circuit incrementally. It will first establish an encrypted connection with the first (guard) node, then through the first node with the middle relay and eventually, through the guard and the middle relay, with the third (exit) node. Establishing is done via certain cell commands and Diffie-Hellman key-exchange algorithm is utilised. It's important to mention that we have one-side entity authentication here, meaning that the Onion Proxy will know to which Onion Router it connects to, but not the other way around. Two symmetric keys are derived from each established key: one for the forward traffic and the second one for the backward one. After establishing the circuit, relay cells can be sent. After receiving such a cell the Onion Router checks the correlated circuit and decrypts the header and payload with the session's key. It checks the integrity of the message by verifying the digest. In order to gain computational optimization, hash computation is often omitted and the first two bytes of the hash are zero. After verifying the message integrity it either processes the message or checks the circID and OR for the

next step of the circuit and replaces circID. If the exit node receives an unrecognised relay it raises an error and tears down the circuit. When Onion Proxy receives the relay response cell it unwraps the relay header and the payload with the session keys that it shares with every Onion Router in the circuit, checking the integrity with every unwrapping. Initial Tor TLS handshake was a straightforward mechanism that included providing supported cryptographic algorithms and parameters by the initiator and choosing them by the responder as well as providing a two-element certificate chain that proved the authenticity of the Tor node. This mechanism had to be later modified due to censoring Tor using deep packet inspection in a way that would make it appear like the usual HTTPS web traffic. Streams utilise SOCKS protocol in order to be able to connect the Onion Proxy to the given address and port. OP either creates a new proxy or uses the newest created open circuit. From that circuit it chooses one node to be an exit node, depending on the exit policy - usually the last node. Due to several flaws with SOCKS in the initial Tor design that could have led for example to a DNS leak, usage of privacy-aware HTTP proxies like Privoxy was suggested. It was later addressed by the Tor Browser bundle. Similarly to closing TCP streams, Tor uses two-step handshake for normal closing, which is especially useful for half-closed connections, and one-step handshake for errors.

QoS

1. Congestion control: congestion control in Tor is achieved on several levels. Firstly, it utilises TCP along with its mechanisms of congestion control and in-order guarantees. Secondly, circuit-level throttling to control the bandwidth within the circuit, where each onion router keeps a track of two windows for incoming and outgoing traffic. Thirdly, an end-to-end stream-level throttling is introduced, similar to the circuit one, however utilising a single window.
2. Fairness and rate limiting: in order to limit bandwidth usage Tor uses a token bucket algorithm, that helps to achieve long-term average rate of bandwidth but at the same time allowing for infrequent sudden spikes. Moreover, streams can be categorised based on the frequency of cells' appearance in order to provide a better latency for interactive streams. The risk of an end-to-end attack in this case is not an additional threat as Tor is already vulnerable to these types of attacks.

Onion Services

Tor can not only provide anonymity for the sender, which was described above, but also for the server itself (responder anonymity). It can be achieved thanks to Tor's onion services, previously known as hidden services or location-hidden services. The server anonymity can have many benefits, which will be described in the use cases and application areas section. The anonymous bidirectional connections is established as follows: First the onion service creates a long-term key pair which will identify the service. Secondly, it chooses several onion routers as its introduction points (IPs), establishes a circuit with each of the IPs, providing each of them his public key, and advertises them in the lookup service which has a form of a distributed hash table - a hidden service directory (HSDir). It then signs the

advertisement, otherwise known as the hidden service description, with its private key. Optionally, a part of the descriptor that includes the introduction points can be encrypted with a key that is shared between the onion service and authorized users. With this approach, the unauthorised users not only will not be able to connect to the service, but also they will be unable to discover it in the first place. If a user wants to communicate with the onion service, looks up in the DHT for one of the introduction points of the hidden service, chooses an OR as his rendezvous point (RP), establishes a connection with both RP and IP and then announces the RP to the OS through the IP, along with a rendezvous cookie, first part of a DH handshake and optionally with additional information, encrypting everything with the onion service's public key. Once onion service receives this message it can connect with the rendezvous point, of course through intermediate nodes, and pass the rendezvous cookie, joined by the second part of the DH handshake and a hash of the session key they are now sharing. Despite the fact that the RP connects user's and onion service's circuits, it is not aware of the communication between them. Neither are onion service's IPs. Once the user sends the control cell along the circuit, the two-way anonymous communication can be performed. Such a separation of functionalities on multiple routers provides smear-resistance, while several introduction points provide robustness and higher availability. There is also a possibility of access control associated with optional authorisation tokens that a user can provide. From 2015, .onion domains are recognised as special-use domains by the IESG and they can obtain TLS certificates [59]. There are two companies that support issuing X.509 certificates for the .onion top level domain. First one is DigiCert, they support Extended Validation (EV) TLS certificates, wildcards (which is an exception as EV in general do not support wildcards [60]) and validity period up to 15 months. Second one is HARICA - they support Domain Validation (DV) certificates. While it may sound redundant as the onion services already seem to provide benefits of TLS in terms of end-to-end encryption and authentication, there can be several benefits from using TLS certificates. Firstly, mixing HTTP and HTTPS can cause sensitive information leakage [61]. Browsers enable additional security and privacy improvements when accessing HTTPS websites related to cookies and sensitive data. Moreover, certain software solutions require TLS by default and allow only the HTTPS. Additionally, users are taught to look for HTTPS when connecting to a website in order to make sure that it is a safe connection and changing their habits might be challenging. Furthermore, if the Tor client is located in a different place than the web server itself, the connection between the Tor client and web server needs to be additionally encrypted; in case of using HTTPS for the onion service the problem is solved.

Bridges

Tor Bridges are Tor relays that are not publicly listed like the standard nodes. They often offer additional functionalities like traffic obfuscation. Bridges were introduced as a response to the censorship of traditional nodes or directory authorities that are easy to block due to their public availability and it was foreseen by the Tor project in advance. The idea was to simply encourage users from less censored

parts of the world to offer themselves as private relays that will not be publicly listed for the people in censored countries. Different ways of distributing bridges:

1. From the Tor website: https://bridges.torproject.org/
2. Via e-mails from bridges@torproject.org upon request. Currently, it is required to send the request from a Gmail or Riseup email address.
3. Every user can set up a bridge and share it to whom he wants

Pluggable transports

Pluggable transports are modules that obfuscate traffic, so anyone monitoring the network activity of a user running Tor with a pluggable transport will see traffic that does not look like typical Tor usage. There are several kinds of pluggable transports (PTs):

1. Obfsproxy: first pluggable transport, created in 2012, encrypts the traffic so that no specific headers are recognizable. In this case censors can either block all unrecognisable traffic which will result in a massive number of false positive cases or to allow it, which is usually done.
2. Meek: pluggable transport created in 2014, based on domain fronting via Amazon Web Services (AWS), Microsoft Azure or Google Cloud Platform (GCP). The basic idea behind it is to route user's traffic to a cloud provider and from within the cloud reach the Tor network.
3. Snowflake: pluggable transport that works as a JavaScript code executed in the volunteers browsers. Client connects to the volunteer's Snowflake proxy via WebRTC that obfuscates his traffic, making it appear as, for example, a video call. The major advantage of this kind of pluggable transport is the simplicity of deployment without the need of dedicated software other than browser extension, although there are also dedicated standalone snowflake proxies aimed for long-term connections. The more people join as a snowflake proxy, the more difficult it will become to censor traffic without a large number of false positives.
4. WebTunnel: officially announced in 2024 is a new kind of bridge that aims to circumvent censorship in heavily-censored countries. It utilises a HTTPS proxy, similarly to HTTPT [62]. Contrary to obfsproxy it aims to mimic a real web server, rather than appearing as unrecognisable traffic.

3.7.3. I2P

History

The idea for I2P started in October 2001 when Lence James, known under a pseudonym 0x90, created IIP - the Invisible IRC Project as a way for instant communication with Freenet users regarding Freenet's issues as well as for Freenet keys exchange. In 2003 it was decided that IRC was not sufficient for the project to serve its purpose. Instead of such an approach an universal anonymizing layer for more variety of protocols was needed. From then, the IIP was also referred to as the InvisibleNet. In the same year an anonymous developer jrandom joined the project and his ambition was to expand

it. He wanted to redesign the system with inspiration from Tor and Freenet projects but at the same time many things specific to the I2P were introduced. Strong emphasis was put on protection from organisations with huge resources. In the same year jrandom took control over the project and it was renamed to the name that exists today - Invisible Internet Project, or I2P for short. Two years later, in 2005, another prominent anonymous developer joined the project - zzz, shortly after joined by yet another developer with a pseudonym Complication. In 2007 zzz with Complication took control over the project after jrandom unexpectedly quitted, which caused turbulence for the project development. Over the years several developers joined and left the project, many improvements were introduced. Project is to this date fully based on volunteers' work. The project used to accept donations but it was discontinued. Today the I2P is considered as one of the most popular anonymous communication networks, currently it has around 55 000 routers, according to the I2P metrics [63, 64]. Users' number may be slightly larger as users from restricted countries may not act as routers.

Design and architecture overview

I2P is an overlay, peer-to-peer network with full encryption and resilience against censorship, eavesdropping and recognition. Every user, unless it is unsafe for him, contributes to the network as a router, increasing the network anonymity, creating a peer to peer network. Also, being a router, or in other words participation in the I2P network, is not necessarily something that should be kept as a secret while information about connections with endpoints associated with specific applications called destinations are. I2P utilises tunnels, encryption (both router-to-router and end-to-end), provides forward secrecy for all connections and self-authenticated network addresses. The network consists, among others, of routers/peers communicating with each other, unidirectional tunnels (one for incoming traffic, one for outcoming traffic, contrary to the Tor's bidirectional tunnels) and a distributed hash table as an internal database for information distribution. These and other elements of the I2P architecture will be described in detail in this section. In order to connect to the I2P network user needs to install a personalizable Java client consisting of a static website template, email and BitTorrent client. As was mentioned before, it also works as a router. I2P offers a complete suite of network protocols and serves as a platform on top of which applications can be built. These applications include newsgroups, e-mail, messaging, file sharing, web browsing, blogging or distributed data storage. One of the major advantages of the I2P network it's its scalability. The more people use the network, the faster it gets as there are more routers in the network and more traffic is being passed by them. It's also getting more anonymous with an increasing number of users as there is much more traffic to pass with for a particular router. Due to the peer-to-peer nature, creating exit points would impose a threat of legal issues for the users, given illegal actions of some other user that would have them as the exit point. Users might not be willing to take such risk, which would definitely reduce the number of users willing to use the network. Due to this fact, along with security issues that are associated with exit points, the I2P does not support the exit points by default, however it is possible to create such a proxy (outproxy)

on a higher layer.

Routers and destinations

Router in I2P is an entity that participates in routing users' traffic. It has the same role as nodes in the Tor network, however, contrary to Tor, each user that uses the I2P is also routing traffic for others. While the fact of being a router or running the I2P software is currently not a secret, it is possible that in the future, when I2P will gain more popularity and the governments will try to censor it, that it will become necessary to add a masking layer to circumvent the censorship. When a user first connects to the network, he connects to special-purpose routers called reseed hosts, that provide an initial set of nodes for a new router to connect with. Destinations are cryptographic identifiers of applications within the network. The fact of a user reaching a given destination is a secret. Also, the user who is reaching the destination remains anonymous.

Network database

Network database, or netDb is a distributed hash table (DHT) database based on the Kademlia algorithm that is responsible for metadata sharing in the network. The database is maintained by a subset of routers called floodfill routers. They are normal routers with high bandwidth, constituting about 6% of the network. There are two types of metadata:

1. leaseSets: gives the instructions necessary for reaching the particular destinations. LeaseSets also include a number of leases, where each lease specifies a tunnel gateway that is necessary to reach a particular destination. Lease information includes: pair of public keys for message encryption, tunnel expiration time and recipient's inbound gateway. LeaseSets are distributed via outbound tunnels in order to avoid correlation between leaseSets and their router and to remain the router's identity. There is a possibility of private encrypted leaseSets for the purpose of reaching services that must not be publicly available. Such encrypted leaseSets are not published in the netDb and require appropriate decryption keys.
2. routerInfo: gives the instructions for reaching the specific router. These instructions include a unique ID in the form of a cryptographic public key, transport addresses and ports, timestamps and other options that can be utilised by the user, everything digitally signed and the signature is also attached. RouterInfo is distributed by the routers to the netDb directly, as this information should not be a secret.

Tunnels and garlic routing

A tunnel is a short-living, direct path between two points underlaid by several routers between, where layered encryption is utilised. Each router can decrypt (or "peel") the encryption layer and forward the message to the next router. The first router in the path, a tunnel starting point, is called a gateway. Tunnel ends with an endpoint. Tunnels are only one-way and for responses additional

tunnel is needed. That's why there are two types of tunnels in I2P that are required for two-way communication: inbound and outbound. Inbound tunnels, as the name suggests, bring messages to the tunnel initiator and outbound tunnels send messages from the tunnel initiator. Sender's outbound endpoint will pass the message to the recipient's inbound gateway given appropriate instruction to forward the message. Utilising two one-way tunnels instead of one two-way tunnel makes traffic analysis harder for the adversary. In order to build inbound and outbound tunnels a user needs to call the netDb and fetch a list of peers that can be used as hops in these tunnels. Then users can gradually, router after router, build the tunnel via construction messages. User sends the message to the first hop, then to the second hop with intermediation of the first router, then third one through first and second until the tunnel is successfully formed. When the sender wants to send a message he first needs to find the receiver's leaseSet in the netDb in order to retrieve receiver's inbound tunnel gateways. Sender then needs to choose one of his outbound tunnels and sends the message through this tunnel along with information to pass the message to one of the receiver's inbound tunnel gateways. To sum up, the path of the message is as follows: first it is sent via the sender's outbound tunnel, then it is forwarded, thanks to appropriate instructions, to the receiver's inbound tunnel. In case the sender would like to receive a response he will need to send his destination within the message. Sender can also optionally include his leaseSets in the message so that receiver would not need to look it up in the netDb. Despite layered encryption in tunnels that was described before, an additional end-to-end encryption needs to be used in order to protect messages that are leaving outbound tunnels and entering inbound ones. The end-to-end encryption is performed with public key cryptography. Messages can be grouped together and form what is called a garlic. Each message in the garlic can be named a garlic clove. Garlic cloves can be accumulated in the inbound gateway and then passed via the tunnel and split in the input tunnel endpoint. That mechanism is possible due to unidirectional tunnels and would not be possible in classic, onion routing bidirectional tunnels. It is often referred to as garlic routing. Garlic contains information about the tunnel which will be used for message sending which makes route establishment much faster. While I2P initially proposed the introduction of batching, mixing, throttling, padding and delaying traffic, it is currently not implemented as parallel support of two types of traffic can be problematic. If it will be in the future, the network should then be considered as a hybrid solution between Mix-Nets and Onion Routing. Routers are chosen for the tunnel based on various metrics, including their bandwidth, capacity measured by successfully built tunnels, which is the direct indicator of their reliability, family diversity, as well as on manually adjustable individual preferences.

I2P protocol stack

Invisible Internet Project proposes its protocol stack on top of which applications can be built. For analysis purposes references to OSI and TCP/IP stacks will be made. In theory TCP/IP transport layer and all lower layers are also a part of the stack, however the overview will be focused solely on the I2P-specific stack. Beginning from the bottom of I2P stack:

1. I2P Transport Layer: in TCP/IP and ISO/OSI stack this layer and each layer after that could be placed in the application layer, however this specific layer can also be treated as an overlay on the traditional transport layer. I2P Transport Layer is responsible for providing encrypted connections between I2P routers. These are not anonymous end-to-end connections but simple, not anonymous, hop-by-hop connections. In I2P Transport Layer there are two protocols that are based on UDP and TCP:
   (a) SSU2: Secure Semi-reliable UDP
   (b) NTCP2: NIO-based TCP

   In the future, this layer can also provide additional obfuscation that will make it harder for the censor to block I2P traffic.
2. I2P Tunnel Layer: layer that provides tunnel connections with full encryption. Protocol data units used within the I2P tunnel layer are tunnel messages and I2NP (I2P Network Protocol) messages. I2NP are messages used to pass information through multiple routers. Multiple I2NP messages layered encrypted along with instructions necessary for their delivery are combined into a larger tunnel message which has a fixed size of 1024 bytes. Each hop in the path removes (decrypts) its layer, reads an instruction and forwards the message to another router.
3. I2P Garlic Layer: layer providing anonymous and encrypted end-to-end message delivery. Similarly to the tunnel layer, the I2P Garlic Layer uses I2NP messages. They are wrapped in layers to ensure anonymity for the sender as well as for the receiver. Garlic messages expand standard onion messages by allowing the message to contain multiple cloves. A clove is a fully formed message with delivery instructions.

Mentioned layers are considered as I2P core layers. There are also additional layers that are considered non-core layers:

1. I2P Client Layer: layer providing ability to use I2P without direct router API usage. I2CP or I2P Client Protocol is a protocol of this layer. It provides asynchronous and secure communication by sending messages over the I2CP TCP socket between a client and a router.
2. I2P End-to-End Transport Layer: layer providing functionalities similar to TCP and UDP from the TCP/IP transport layer over the I2P network. Two prominent solutions that exist within this layer are:
   (a) Streaming library: functionality similar to TCP streams over I2P network.
   (b) Datagram library: messaging similar to UDP over I2P network.
3. I2P Application Interface Layer: layer providing libraries that make development on top of I2P easier. The protocols of this layer utilise the protocols of the I2P End-to-End Transport Layer. Example protocols:
   - SAMv3: supports multiple communication types, including reliable in-order TCP-like streams, UDP-like datagrams or unauthenticated raw ones with minimal overhead. It supports many programming languages as well. SAMv3 can be used for creating I2P-native

applications.

- I2PTunnel: for creating general-purpose TCP-based connection, enabling a seamless proxying of existing non-I2P applications into the I2P network without the need of writing I2P-specific code.

4. I2P Application Proxy Layer: layer, used for proxy systems. Example protocols: SOCKS, HTTP, IRC.
5. I2P Application Layer: common name for the applications that can be built on top of the I2P stack.

I2P applications

There are many I2P applications, some of them included within the I2P install bundle. Those include for example:

- Naming library and address book: web-of-trust-based, local naming system included within the bundle. Specific long I2P addresses can be added as a human-readable entry in the address book.
- I2PSnark: BitTorrent client working within the I2P network. Although it is not the only BitTorrent client available, it is the one that is shipped within the install bundle. In general it is impossible to connect to non-I2P torrents from within the I2P and vice versa.
- I2Pmail/susimail: internal and external email service working within I2P.

There is also a possibility of creating custom applications and distributing them as I2P plugins which can then be easily integrated with other users' I2P routers. Users can access the I2P router API for example via a simple JSONRPC2 interface. The I2P network allows services to be hosted anonymously in the I2P network. These destinations, for example websites, called Eepsites, are subject to the i2p pseudo-top level domain. Domain name is resolved into an I2P peer key by a proxy named EepProxy. Service providers can use solutions like I2PTunnel to share basic client-server applications in I2P, it also includes support for SOCKS proxies. On the other hand, if an application is peer-to-peer, requires a more sophisticated approach, enhanced anonymity or performance optimisation, they can write I2P-specific code and achieve tailored solutions. There is also a possibility of embedding I2P into an application.

3.7.4. Lokinet

Lokinet is a relatively recent decentralised anonymous network that utilises onion routing architecture. The network was described in the 2018 paper [14]. It is one of the first networks that introduced economic incentives for the node runners. Lokinet introduces Low Latency Anonymous Routing Protocol (LLARP) for communication within the network. It is said to be a hybrid solution between Tor and I2P. In fact, the design is more similar to the Tor one, including client-server architecture; however, Lokinet, contrary to Tor, does not use semi-centralised directory authorities. Instead, it relies on DHT, similarly to I2P, although LLARP's DHT is built from blockchain. Moreover, LLARP is not restricted to TCP-only traffic like Tor is. Lokinet has formal support for exit nodes,

similarly to Tor and contrary to I2P, which requires dedicated outproxies. Similarly to Tor and I2P, Lokinet supports services within the network - SNapps - that work analogically to Tor's onion services or the services within I2P. Lokinet also introduces a message storage concept, for users that are temporarily offline, that is not present in Tor nor in I2P. The biggest difference, however, is the placement in the ISO/OSI model; while Tor and I2P are built on top of the transport layer, Lokinet is built within the network layer and can support any IP-based protocol. Lokinet's infrastructure underlines a popular instant messaging app - Session, although Session is temporarily switched from utilising Lokinet directly for the sake of simplified Onion Requests due to compatibility issues on various platforms [65], however, as the infrastructure is the same, Session user base directly increases the anonymity set of Lokinet. Moreover, the return to Lokinet is planned when the compatibility issues are solved. The return is needed as low latency properties of Lokinet will allow for anonymous voice and video calls, while the current simple Onion Requests mechanism is not able to provide such functionality.

3.7.5. Other designs

There are also newer propositions that aim to address issues with the original onion routing design or with the Tor network, as it is considered a direct successor to the onion routing design. Examples of such networks include HORNET (High-speed onion routing at the Network Layer) [66], TARANET (Traffic-Analysis Resistant Anonymity at the Network Layer) [67] and Dovetail [68]. They introduced potential improvements in terms of latency or traffic analysis; although requiring changes in the network layer, they would be difficult to perform.

3.8. Summary

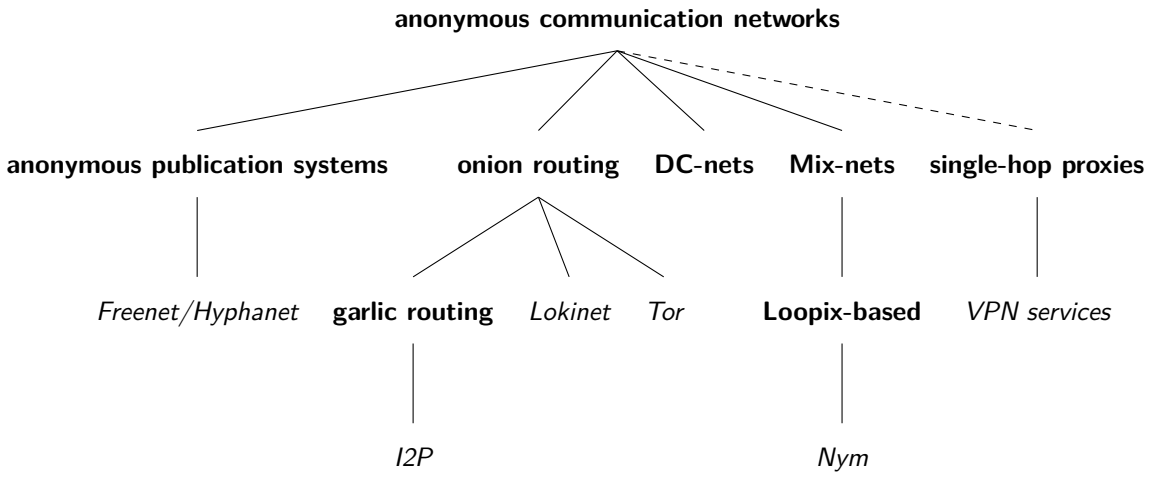

fig. 3.1. Classification of anonymous communication networks

The classification of anonymous communication networks is shown in 3.1. The tree structure presents the main categories of ACNs that have been developed over the years, highlighting the technical diversity.

At the top of the diagram are all anonymous communication networks. The next level distinguishes five main families: anonymous publication systems, onion routing, Mix-nets, and single-hop proxies. Among these, single-hop proxies are shown with a dashed line. This is to underline that, unlike other branches, single-hop proxies do not provide anonymity by design but only by policy. Because of this, they cannot be considered proper ACNs in the strict sense.

Anonymous publication systems (such as Freenet/Hyphanet) are also present in the classification. While they can provide strong anonymity, their main purpose is very narrow and is strictly based on anonymous publishing and distributed, censorship-resistant storage, and they cannot serve as general-purpose universal networks. For this reason, anonymous publication systems are not included in the comparative analysis in the following chapters.

Another important family included in the classification are DC-nets. While DC-nets have strong theoretical anonymity properties, they are not widely used in practice today due to scalability, performance limitations, and significant bandwidth overhead. However, their unique approach has inspired many later designs and improvements in the field of anonymous communication.

The two other branches represent the main technical directions in ACN development. Mix-nets (and their modern descendant, Loopix-based networks like Nym) and onion routing (including classic Tor, Lokinet, and garlic routing solutions such as I2P) are the most influential approaches, each with their own properties and typical use cases. These designs are the most important for understanding practical anonymity online today.

Because the goal of this work is to compare fully functioning and practically relevant ACNs, the following chapters focus on four representative and currently active networks: Tor, I2P, Lokinet, and Nym. These systems illustrate the main architectures and techniques found in modern anonymous communication, and the analysis will show how their design choices affect their strengths, weaknesses, and best application areas.

4. RELATED WORK

Existing literature on the analysis of anonymous communication networks includes legacy systems such as JAP and does not compare more recent network designs like Lokinet or Nym [69, 70, 71, 72, 73, 74]. As ACNs evolve, older measurement studies become less representative of the current performance and other properties.

Furthermore, much of the research concentrates on traffic classification techniques [71, 74, 73, 72], rather than practical usage in various use cases or application areas. While some studies provide comparative analyses of anonymous networks, they often lack a clearly defined goal or context for their comparison [75]. Other works address usage aspects [70, 69], but these are typically outdated, covering deprecated ACNs, and do not consider the latest developments in the field. Moreover, mentioned papers provide only general guidance rather than conducting a comparative analysis based on specific use case scenarios.

5. USE CASES AND APPLICATION AREAS

5.1. Identifying use cases and application areas

1. Internet censorship circumvention: in restricted countries ACNs can be used to circumvent Internet censorship, in order to be able to browse it freely. Among the most prominent examples of censoring countries are China and its Great Firewall [76], Iran and its extensive history of Internet censorship, along with other Middle East countries. Censorship in the Arab World especially intensified during the Arab Spring where Tor played a crucial role as a tool that was used to bypass Internet restrictions [77]. There are also less-known but equally severe examples of the Internet censorships, for example from Africa [78]. Freedom House is an organisation that performs periodic measurements of Internet censorship across the world [79]. One may think that in order for the user to reach the website he would simply need to use an ACN and visit it, circumventing the censorship. While it may be true in some cases, in others it is a little more complicated. There may be some built-in security features that are noncompliant with ACNs nature. For example: If someone is accessing a website in the clearnet from the ACN, he would from time to time change his circuit which was described before. This implies the fact that the exit node will change and therefore the location from which the user visits the website. While in the ACNs such a location change is normal, it may seem suspicious for a web service as it may be associated with a hacker attack. The example of location issues described above was a real case of Facebook and an issue that legitimate users faced while reaching it via the Tor network. They could also count how many people use Facebook via Tor. In April 2016, less than two years after the first introduction of the onion service which took place in October 2014, the number of users per month passed one million [80]. Not only Facebook did create an onion service but they also used single hops. As it is not a secret what is the real address of Facebook's servers, they reduced the number of nodes (from the server side) necessary for the end-to-end user-server communication, reducing delays [81]. Additionally, Facebook provided an SSL certificate for their onion address. The benefits of such action were provided in the onion services section.
2. Surveillance protection in web browsing: ACN can protect its users against monitoring their web activities. An external observer, i.e. an ISP, will only see that a specific user is using the ACN, unless the additional obfuscation is provided. In that case, the ISP can see various masking traffic, unrelated to the real actions of the user. While protection against censorship is challenging, protection against surveillance is even more difficult. In case of censorship the feedback loop is straightforward: certain websites are either reachable or blocked. Surveillance, on the other hand, is much harder to detect, and users are usually unaware that they are being monitored.
3. Unbiased research: Getting unbiased results is crucial when it comes to research in order to ensure access to diverse, accurate and unmanipulated information, free from personalisation or

censorship. ACNs help by masking user identity and location, preventing search engines from tailoring results based on user's past behavior, geography, or profile data, therefore promoting objectivity and neutrality in search outcomes.

4. VPN service alternative: certain ACNs, in theory, could be considered as an alternative to popular VPN services. However, as was described before, there is a major difference between these two: VPN providers in most cases provide so called privacy by policy, meaning that a user has to trust some entity with his privacy and hope that they will stick to their policies, while ACNs' privacy by design approach ensures that if any entity within the network or outside of it would like to reveal the user's identity, it is not able to do that. Another benefit is the fact that many ACNs are open-source and free. In the case of VPN providers they are usually paid services and if a user from a restricted country is purchasing such service from within that country it may draw the attention of the censoring government [82]. On the other hand, if the VPN service is free and at the same time it is not funded by volunteers and various organizations - it is highly likely that such a company sells the data of its users as running infrastructure costs money.
5. Remote access to host: hidden services can be used for remote machine access. Not only that but thanks to NAT punching the port forwarding can be avoided, which would not be possible in case of using clearnet alternatives that require opening an appropriate port. In other words, a hidden service can be set on any machine behind a firewall in order to reach it regardless of firewalls.
6. Internet of Things (IoT): The IoT refers to the concept of connecting everyday physical objects to the internet so that they can collect, share, and exchange data. These objects can be anything from household appliances and wearable devices to industrial machinery and cars. Each device can communicate with other devices or systems over the internet or other networks. IoT security is crucial, especially when it comes to critical infrastructure. What is even more concerning is the fact that IoT manufacturers often underestimate the role of security in their devices, which results in lack of data encryption or insufficient updates. Considering the fact that these devices are often exposed to the Internet without proper authentication or confidentiality we can draw a conclusion that the problem is in fact exceptionally serious. Utilising ACNs with IoT can provide enhanced security as devices are not exposed to the internet, the fact of communication between the devices or between devices and computing centers can be hidden.
7. Running cryptocurrency nodes: ACNs can be used to operate various cryptocurrencies nodes. Nodes can communicate within the network, protecting their real IP address. Thanks to such an operation, it is impossible to censor these nodes nor to shut them down as their location is not known. Moreover utilisation of hidden services can make the node setup easier and safer as no ports on a firewall need to be forwarded. Examples of cryptocurrencies that support running nodes over ACNs are Bitcoin, the most popular cryptocurrency, or Monero, widely recognised as the most private of the cryptocurrencies. Their nodes can be run within clearnet, the Tor

network, the I2P network or any combination of them. In order for the nodes that run within ACNs to connect to the clearnet nodes, they need to use bridge nodes that support both ways of communication. Moreover, users that utilise wallets over ACNs can anonymously connect to ACN-specific nodes and the fact of the communication between these parties is hidden as well as their identities behind the onion addresses.

8. Web-based cryptocurrency wallets: Web-based cryptocurrency wallets can leverage hidden services to enhance privacy and security for their users thanks to the security properties that come with them. An additional benefit from using hidden service for such use cases is that there is no risk from the exit node as the traffic never exits from the network. Malicious exit node could, for example, manipulate the cryptocurrency address in such a way that the cryptocurrency would be transferred to the attacker instead of the legit receiver. Of course such malicious activity of the exit node would be most likely recognised by the ACNs' security mechanisms, however it is better to prevent even the tiny risk of such an event, especially considering the fact of irreversibility of the cryptocurrency transactions and potential losses.
9. Emails: Email is a fundamental communication medium which can largely benefit from the utilisation of ACNs, especially in case of activists, people under surveillance or whistleblowers.
10. Whistleblowing: ACNs are a perfect tool for journalists to communicate with their sources safely and anonymously. An example of a solution that utilises ACNs for this use case is SecureDrop, which is a free and widely recognised platform for such communication. It utilises the Tor network in order to provide anonymity of the source, which is one of the most fundamental aspects of the free press, as was recognised by The European Convention of Human Rights (ECHR). Thanks to the anonymity by design it is impossible for journalists to reveal the whistleblowers identity, no matter how much pressure the government or other entities put. Another widely recognised platform for whistleblowing is WikiLeaks. It's a non-profit organisation and publisher of leaked documents founded in 2006. WikiLeaks releases vast numbers of documents exposing human rights violations and abuses by various entities, including governments. To protect whistleblowers' anonymity, they use a Tor onion service as a secure submission platform. While anonymity is also preserved when accessing regular websites with Tor, onion services provide an additional layer of security desired by WikiLeaks' creators: these sites are inaccessible without Tor, reducing the risk of exposure if submissions occur outside of the Tor network. GlobaLeaks is a free, open-source and internationalised software that enables creation of whistleblowing initiatives. The GlobaLeaks project started in 2010 and the first version was released in 2011. It uses Tor onion services for the anonymous submissions. GlobaLeaks is a framework that was adopted by more than 10000 projects [83]. The use cases of the projects implementing GlobaLeaks include independent media, activists, governments, public agencies and many more. One of the examples is WildLeaks [84] - the world's first whistleblowing initiative for environmental and animal-abuse crimes.
11. Securely accessing websites: There are several risks that can pose a threat to the people while

accessing regular websites. For example malicious local DNS servers can redirect users to other IP addresses than expected. Another possibility is from the certificate authority site where fraudulent digital certificates can be used to perform Man-in-the-middle attacks. Such events occurred in the past, Turkish CA TURKTRUST being one such example [85]. BGP hijacking is yet another issue that can be used against users accessing the websites, which is a takeover of IP addresses by corrupting BGP protocol used in the Internet. In order to address such issues there is a simple solution: to create a hidden service for the website. There would not be a risk from the malicious DNS servers. Even though certificate authorities are not present, an onion service still provides end-to-end authentication and encryption. Hidden services are end-to-end authenticated and encrypted.

12. One-to-one file transfer: Secure file transfer may be beneficial in numerous situations, for example a doctor securely sending patient's data to him or an organisation and its employees sending sensitive information between each other internally. The advantage is that there is no need of trusting any third party service provider like DropBox or WeeTransfer and you do not have to physically pass sensitive files stored on data storage devices. An example of such a solution in Tor is OnionShare. A user that wants to send a file creates an onion URL which allows the other user to fetch the file and we can optionally enable URL expiration so that it would be impossible to download the file again after it is downloaded once. This guarantees that the file was not downloaded by any malicious middleman. Moreover the service can additionally require a private key to access the file. In I2P it can be achieved with numerous communicators with file sharing capabilities or eepsites secured with private key that disallow unauthorised access.
13. One-to-many file publishing: users can distribute files over ACNs including documents, media or software without the censorship. One example from history where such a use case was beneficial is the Zyprexa scandal from 2006. Documents were published online alleging that the pharmacy company Eli Lilly was aware of the severe side effects associated with its antipsychotic drug Zyprexa and continued to sell it regardless. Initially a pharmacy giant was able to quickly remove the documents from the websites where they were hosted, until it was moved to a hidden service. It was no longer possible for the company to remove uncomfortable documents and people had the chance to read them.
14. Anonymous container registries: container registries can have several benefits from utilising ACNs, hidden services in particular; it can prevent censorship of the container images, protects against registry spoofing attacks, increasing security of pulling images, prevents tracking of contributors in case they are not comfortable with revealing their identities and on top of that, hidden services can provide end-to-end encryption and authentication.
15. Asynchronous messaging: Users can communicate asynchronously in an anonymous fashion in a privacy-preserving manner. As the communication is asynchronous it can accept longer delays but it requires the possibility of replying.

16. Software updates: utilising ACNs with software updates can prevent numerous attacks as it is not possible to tell who is performing an update. On top of that utilising hidden services can provide end-to-end authentication and encryption for the users that are performing the software updates.
17. Instant messaging: Users can chat synchronously in a privacy-preserving manner, however, contrary to the asynchronous communication, latency is crucial in this use case.
18. Statistics: collecting usage data in privacy-preserving fashion is crucial. ACNs can unlink the user from the usage data he is sending, unless the data contains identifiable information that should also be removed. Utilising ACNs can allow for the large-scale analysis while preserving users' privacy.
19. Gaming: utilising ACNs in gaming can help to circumvent the censorship of the certain game, protect users from revealing their private data which leakage may lead to doxxing, or protect IP addresses of players to avoid DoS attacks. Moreover games that are not widely considered harmful, such as chess, can be forbidden in certain countries [86]; therefore protecting the game server in this scenario can also be justified.
20. Censorship and denial of service-resistant archives: location hiding helps to prevent DoS attacks, including DDoS. It can occur when the service content is not necessarily in accordance with the regime government and therefore is exposed to DDOS attacks from their site. Instead of attacking a single server, an attacker would be forced to attack at least several introduction points. One of the first propositions of such systems that were censorship-proof by design were Eternity Service and Free Haven described before, however the most popular archive today is The Web Archive which decided to create a hidden service for its archive as the website was targeted with numerous attacks.
21. Voice calls: voice calls can benefit from utilising ACNs to preserve users' privacy, hiding not only the identity of both parts of the communication but also the fact that they are communicating.
22. Video calls: video calls are one step further compared to the voice calls, involving the visual aspect besides the voice. Utilising ACNs here can be beneficial for the same reason as with the voice calls.
23. Source code hosting: Code repositories hosted within the ACNs can benefit from the censorship resistance, while people fetching the repositories can benefit from end-to-end encryption, authentication and the fact of hiding communication parties, as well as protecting against targeted attacks for users accessing certain repositories.
24. Distributed data storage systems: highly-available distributed data storage systems can further enhance their censorship-resistant features by utilising ACNs. It would also protect the publishers of these files as well as the people fetching them.
25. Protecting privacy in web browsing: while it is not a secret that companies sell the usage data of their users, especially while web browsing, which highly violates their privacy. As the users' data is highly valuable for advertisers, the process of collecting the data is evolving. Utilising ACNs

can help to protect the privacy while web browsing.

26. E-voting systems: Anonymity of the voters is one of the most important characteristics of democratic elections. It is crucial to ensure unlinkability of a given vote to its voter and using ACNs can help with it.

5.2. Categorising use cases

Above use cases can be grouped into several groups based on the requirements they have in terms of the ACNs. The anonymity is a basic requirement for these use cases, but the strength of provided anonymity can be evaluated within criteria. Each group will be declared in terms of what kind of anonymity is needed for this group: sender, receiver (server) or both. Following groups were distinguished:

5.2.1. Low-latency intra-network communication

Use cases and application areas for which the lowest possible latency is crucial. It includes instant messaging, gaming, voice, and video calls. As these use cases are focused on two-way communication along with the fact that giving up one-end anonymity may impose additional threats from the traffic analysis perspective with low-latency communication, the comparison will be based only on solutions providing two-way anonymity, meaning that the communication has to be performed within the network. While both sides of communication are often known to each other, the importance of such a property lies in the lack of exposing metadata - who is speaking to whom. Criteria defined for this group:

1. low latency (weight 0.25): if one-way transmission delay is too high, the applications mentioned above cannot be accomplished. Therefore, the most significant weight was assigned to this criterion. According to ITU-T recommendations [87] maximum one-way latency for international calls is 400 ms, ideally 150 ms or less.
2. low jitter (weight 0.25): great variation between subsequent packet arrivals can significantly reduce the perceived connection quality [88], and similarly to latency, making use cases from this group not feasible to deploy. It is the reason why jitter was assigned a stronger weight. The lower the value for jitter, the better the connection quality is; the value should be less than 30 ms.
3. throughput (weight 0.2): sufficient throughput is needed for transmitting video and audio frames, especially those of high quality. One of the least demanding voice codecs is G.729 which requires only 8 kbit/s of throughput [89]. When it comes to video calls, Skype required [90] a minimum 128 Kbit/s and recommended throughput for high quality video calls was 500 Kbit/s. The more throughput, the higher the quality of the video and audio frame can be. As it is also an important property for the use case scenarios, it was assigned a stronger weight.
4. both-end anonymity strength (weight 0.15): while the stronger anonymity, the better it is for the ACNs users, according to the literature it is not possible to provide low latency, high bandwidth

and strong anonymity at the same time [91]. Choosing stronger anonymity over the latency or bandwidth would make use cases from this use case group irrelevant, therefore anonymity was assigned a smaller weight than the previous criteria, although the anonymity is obviously still relevant and was not omitted.

5. UDP support with low packet loss (weight 0.15): usually low-latency applications and protocols rely on the UDP protocol as TCP's overhead is considered too impractical. Examples of such protocols include QUIC and RTP. Even if TCP could have provided a decent performance, running a service via ACN should not require major architectural changes for popular solutions used today. While TCP handles packet loss underneath, UDP does not. However, packet loss is more acceptable than high latency or jitter [88], it is still an important property that should be considered within this category. It was assigned a weight according to its importance in this use cases category. Considering all of these aspects, it was assigned a weight of 0.15.

5.2.2. Highest anonymity and latency-tolerant

Use cases and application areas for which the latency is not crucial and can be traded for potential anonymity benefits. Those include emails, whistleblowing, asynchronous messaging, statistics and e-voting systems. As the use cases are mainly focused on the sender anonymity, only this type of anonymity will be considered within this category. Criterias defined for this group:

1. sender anonymity strength (weight 0.5): as low latency, high bandwidth and strong anonymity cannot be combined at once [91], and considering the fact that currently there are no low-latency networks providing strong anonymity (they could have been potentially based on DC-nets) in this use case group the anonymity can be chosen with the cost of larger delays, providing strongest metadata protection and therefore can be considered as the main focus within the category, receiving the largest criteria weight. It also allows for the network to be more resilient against traffic analysis attacks, which will be described in the next chapter, that can be performed by a powerful global passive adversary - potentially the one that a whistleblower wants to reveal information about.
2. reliable delivery (weight 0.25): it is crucial for the user to determine whether his message has been delivered due to the importance of the information that is often associated with this use case group. The lack of a mechanism that addresses this issue, such as an acknowledgement, is a significant drawback that was present in early Mix-net ACNs [92]; therefore, this criteria was assigned a weight of 0.25.
3. message persistence (weight 0.25): asynchronous communication may often involve parties that are temporarily offline. It is therefore a decent feature for the network to have the built-in possibility of temporarily storing messages; otherwise, the message has to be sent to an always-online entity within the ACN, possibly an offline entity outside of the ACN, or a service provider must create such a mechanism on top of the network, if there is such a possibility. Lack of these

possibilities also imposes security issues as described in the literature [93]. Due to these factors, it was assigned a weight of 0.25.

5.2.3. Web browsing-based

Under this category, there are use cases that are strictly related to web browsing: internet censorship circumvention, surveillance protection in web browsing, unbiased research, VPN service alternative, web-based cryptocurrency wallets, securely accessing websites, and protecting privacy in web browsing. Within this category, mainly sender's anonymity will be considered in the comparison, as it is a category highly focused on the users. Criteria defined for this group:

1. clearnet support (weight 0.25): currently there are over 1.5 billion websites on the world wide web and less than 200 million of them are active [94]. In Tor, the most popular anonymous communication network, there are around 800 thousand unique onion addresses [95, 96] and not every onion address is associated with a website as there are many other use cases for them. Therefore, the vast majority of websites are not reachable as hidden services, requiring for this specific use case group to allow users for exit traffic. That is why this criterion was assigned the most significant weight of 0.25.
2. censorship resistance (weight 0.25): as many countries impose censorship on ACNs, Tor in particular as the most popular network [76, 78], it is crucial for the network to address censoring in order for the users in these countries to be able to use the ACNs freely for web browsing-based use cases. It was for that reason considered as the second most important feature in this use case scenario, receiving a notable weight.
3. latency (weight 0.2): as web browsing is an interactive activity, it requires little latency. While the lower the value, the better it is for the user, the anonymity must be preserved. It implicates a trade-off between those properties as described in literature [13]. Due to the significance of this property in terms of comfortable web browsing, this criterion was assigned a weight of 0.2.
4. Sender anonymity strength (weight 0.2): as this use case group does not aim for the strongest anonymity that cannot be achieved without additional latency, a trade-off must be accepted. For that reason, anonymity strength is considered as a criterion with significant weight, but not more significant than the latency.
5. web browsing optimisation (weight 0.1): ACN optimisation in terms of web browsing such as ease of setup, browser fingerprinting resistance, HTTPS support, or others. While it may bring additional benefits for utilisation of a given ACN in web browsing, it is not a critical functionality, and therefore was assigned the lowest weight.

5.2.4. File sharing-based

This category involves use cases and application areas related to file sharing, including: one-to-many file publishing, one-to-one file transfer, anonymous container registries, anonymous software

updates, source code hosting, and distributed data storage systems. Within this use case group, both-end anonymity, including mutual anonymity (parties of the communication are not known to each other), will be considered as in these use cases, anonymity is needed for both sides of communication. While in theory this category could also compare anonymous publication systems, they can fulfill only the minority of the mentioned use cases; therefore, they would not be taken under consideration. Criteria defined for this group:

1. download speed (weight 0.3): download speed is critical for transferring large files. While the higher the download speed the better, the high speeds that are known from non-ACN-based file sharing are not achievable in ACNs. Due to its intuitively large significance in this group, it was assigned a weight of 0.3.
2. both-end anonymity strength (weight 0.3): Due to the importance of anonymity not only for the receiver/downloader but also sender/publisher as described in literature [50, 51], it was assigned a weight of 0.3.
3. volume correlation resistance (weight 0.3): while it can be considered as a part of anonymity strength, a separate criterion was distinguished due to its importance in this group of usage scenarios with appropriate weight. Large volume transfers create an identifiable pattern that can be easily distinguished by an attacker if there are no additional protections against it [13]. [**?**]ile sharing protocols support (weight 0.1): there are numerous file sharing protocols that optimise the process, for example BitTorrent. The ability to deploy them in an ACN without major modifications is considered as a desirable property, although not a critical one, and therefore assigned a weight of 0.1.

5.2.5. Infrastructure security and resilience-based

In this category, the highest importance is associated with the security and resilience of the underlying infrastructure. It includes remote access to hosts, Internet of Things (IoT), running cryptocurrency nodes, as well as censorship and denial-of-service resistant archives. As these use cases are primarily focused on server-side anonymity, only ACNs providing such functionality will be considered. Criteria defined for this group:

1. resilience to active attacks (weight 0.3): these long-running services need to have high resistance against active attacks provided by the ACN, especially disruptive DoS ones that may cause decreased availability of the service, like it was in the case of an Internet Archive [97]. The ACN must be highly resilient to these types of attacks; therefore, it was assigned a notable weight.
2. authentication and authorization mechanisms (weight 0.3): one of the benefits of utilising ACNs in this use case group is a possibility of omitting exposing an infrastructure to the clearnet. As hidden services across ACNs are stored in DHT structure and can possibly be enumerated [98], additional mechanisms for preventing direct access to hidden services are desired - otherwise, these mechanisms need to be implemented on the application side.

3. server anonymity strength (weight 0.2): in this use case group, providing server anonymity is crucial. Hiding location may positively correlate with the mentioned resilience to active attacks mentioned earlier. The anonymity can also serve for protecting critical infrastructure from discovery [99].
4. academic research and maturity (weight 0.2): academic research directly influences the resilience of the network as it investigates various aspects, potential threats and attacks on a given ACN and gives more confidence with regards to its safety. Maturity is directly correlated with smaller chances of existing bugs as many of them were discovered through the years of existing a given ACN. The reasons for this are similar to Kerckhoffs' principle [100]. The more a system is studied and tested openly, the less it relies on secrecy to stay secure. Instead, security comes from technical aspects, examined by many experts and used over time, which helps find and fix problems. So, academic research and maturity both make an ACN safer because they lead to more thorough testing and fewer hidden bugs. It is therefore assigned with a significant weight of 0.2.

6. THREATS, ATTACKS AND LIMITATIONS

6.1. Attacks

6.1.1. Passive

Passive attacks in ACNs are mostly related to traffic analysis. Detecting this category of attack is almost impossible.

1. Timing correlation: One of the most effective attacks against low latency ACNs. Given an adversary who can watch traffic entering and leaving the network, he can easily distinguish who sent the message due to the timing correlation. In order to mitigate this kind of attack, a non-deterministic strategy needs to be included. In Mix-nets, batching with various strategies is deployed. Addressing the timing correlation is always associated with additional latency and therefore smaller usability in lower-latency usage scenarios. While there are propositions of optional inclusion of non-trivial delays and batching strategies in I2P, they are not yet implemented. As in I2P each user acts as a router it will impose additional effort on the attacker to distinguish the traffic originating from the user from the traffic he is passing through, however it is not a complete mitigation of this type of an attack. Tor and Lokinet are the most vulnerable to the timing correlation attack, however the more users the network has, and therefore the larger anonymity set is, the harder it is to perform this attack.
2. Size correlation: Attack in which an adversary correlates the message with its sender due to its size. Lokinet is the most vulnerable to this type of attack as it does not propose any mitigation. Tor utilises fixed-size messages, however it is only a partial mitigation of the issue, especially when a user is sending large messages, as the sequential chunks clearly form a larger message and the pattern can be therefore observed. In I2P the chunks can be initially combined into larger messages and then, after leaving the sender's outbound tunnel, split and delivered through multiple receiver's inbound tunnels, which can make the size correlation harder to perform. Nym provides the best defense against this attack, providing both padded Sphinx messages along with cover traffic which makes observing message size patterns impossible.
3. Intersection attack/statistical disclosure/traffic pattern observation: Identifying users by correlating online presence over time. When an adversary can observe one endpoint of the communication as well as the fact of a user entering the network, the fact of intensified message number on the receiving side can be associated with a certain user that is most likely a sender. This attack is especially feasible when an adversary can observe the network for a long period of time. In theory, when a user is constantly connected to the ACN, it would not be distinguishable, however this is not an acceptable assumption to make. The more users and routers are in the network, the more complex the attack is to be performed. Masking traffic and non-public routers, for example

pluggable transports and bridges from Tor network, can also mask a fact of connecting to the network. The best solution to this issue, deployed by Nym, is to include a cover traffic - as every endpoint is constantly receiving it, an adversary is not able to tell whether a certain entity is receiving larger traffic than usual. Batching and mixing is yet another potential solution to this issue.

4. Browser fingerprinting: browser-related attack that involves collection of user's data that can be later used to identify him. Tor solves this issue by providing a custom, hardened browser. While other ACNs are vulnerable to this issue, it is less performable in hidden services-focused ACNs like I2P.
5. Unencrypted content observation: an attack associated with malicious exit nodes that can observe unencrypted traffic of users that are reaching clearnet resources over unsafe protocols such as HTTP. The solution for this attack is to either only use end-to-end encrypted protocols, for example, HTTPS-based websites, or not to leave the ACN at all and focus on the hidden services.

6.1.2. Active

1. Key exposure: while brute force-based attacks on cryptographic keys is not feasible in acceptable time with the current technology, the key can be acquired by other means. In case of keys that are not periodically rotated, and therefore ACNs that do not provide perfect forward secrecy, it may be potentially used to decrypt an encryption layer from the past. In most of the ACNs, the keys are ephemeral, and therefore leaked keys can only be used to unwrap one layer of encryption for a short period of time.
2. Tagging attack: attack based on modifying a message in a way that it would be distinguishable later in the message path. Message integrity checking in ACNs is crucial when it comes to defending against this type of attack and most of them include the mechanisms for providing integrity.
3. Content modification: an attack that is one step further compared to the unencrypted content observation passive attack described above. In this case the malicious exit node modifies content before passing it through. The mitigation is the same as described with the unencrypted content observation attack.
4. Sybil attack: an attack in which an adversary creates a large number of routers and tries to, for example, deanonymise users or perform other attacks such as DoS. I2P is particularly vulnerable to this attack due to its peer-to-peer architecture and the lack of effective mitigation mechanisms. In Tor the attack needs to be performed over a long period of time as a sudden spike in the number of nodes can be easily noticed and addressed. Lokinet and Nym consider themselves as more resistant to this type of attack due to economic incentives for the node runners. Nym also includes penalisation of the operators that register multiple nodes.
5. N - 1 attack: an attack aimed for Mix-nets in which an adversary floods the mix with his n - 1

messages, leaving space for one non-adversary message which he aims to target. Nym addresses this issue by using cover traffic and stratified topology.

6. Partitioning attack and eclipse attack: an attack in which an adversary tries to isolate a single target (eclipse attack) or a larger network segment (partitioning attack), forcing victims to connect to the routers controlled by the adversary. The attack involves interference into the mechanisms of discovering routers within the network, for example netDb in I2P or directory servers in Tor. ACNs such as I2P with more decentralised nature during new user joining, can also suffer from a specific type of the eclipse attack called the bootstrap attack in which an attacker provides a list of his malicious routers instead of the legitimate ones.
7. Denial of Service (DoS): attack that aims to disrupt an ACN. The attack has its distributed version - DDoS, that is performed from many places. The larger the network is, the more costly the attack becomes. In peer-to-peer ACNs like I2P naive DDoS may potentially have the opposite effect as each new user will also act as a router and speed-up the network, unless the attacker modifies his nodes to only consume from the network and do not contribute. There are many forms of DoS, often addressing ACN-specific technical solutions, for example Sniper attack [101] on Tor.
8. Compulsion attack: non-technical type of attack in which a router runner is forced to give up control over his router in order for the adversary to control the traffic passing through it. This attack is not useful against the networks that have perfect forward secrecy, meaning the frequent key rotation, as gaining control over nodes is not useful for decrypting previous messages. While this attack may have been more dangerous if the node runner ran all nodes in a given path, most of the ACNs provide diversification mechanisms for the nodes in order to avoid them from being under the same jurisdiction et cetera.
9. Cryptographic attacks: while today's ACNs use algorithms considered safe, meaning that they cannot be brute forced in an acceptable time, the highest danger is associated with quantum computers and their ability to break currently safe cryptographic algorithms. While it is not a threat that is present today, Tor and I2P are already in the process of replacing their currently used algorithms with post-quantum-based ones.
10. Bugs: attacks that exploit code vulnerabilities are mostly related to less mature ACNs with less research backup.
11. ACN-specific attacks: there are many ACN-specific attacks that are usually an effect of omitting a certain aspect during projecting of the network. Contrary to bugs, in this example the ACN works as design but the issue lies with the design itself. The solution is also more costly than a resolution of a bug as it involves changes in the architecture. An example can be guard discovery attack [102] on Tor.

6.2. Economic sustainability

Economic sustainability is a critical challenge in ACNs. The approach to the economic issues related to ACNs varies from network to network. Peer-to-peer solutions, like I2P, benefit from the decentralized nature. Each I2P user that utilises the network contributes to it for the others, enhancing scalability and anonymity. The development team behind I2P does not accept direct donations; they encourage supporting application developers instead. One of the examples is StormyCloud that provides a free outproxy service for I2P.

Tor has an entirely altruistic volunteer-based model in terms of running nodes. This approach has turned out to be working well for over two decades of Tor history. Tor underlines the importance of diversity among their nodes, and the approach they have chosen seems to align with this goal.

Lokinet focuses on economic incentives for node runners, rather than volunteer-based networks. While this approach can attract a broad set of participants, which it currently does, it introduces several potential challenges. Firstly, the node runners may choose the cheapest infrastructure providers in order to maximise their profit, which may lead to lesser diversity of nodes within the network. Secondly, the token-based reward that Lokinet proposes may be volatile. The value of the token they provide is strictly speculative and may vary significantly over time, potentially discouraging node runners when the price goes down. The development team does not accept direct donations; similarly, to the node runners, they benefit from the tokens.

The Nym network has a similar approach to Lokinet in terms of incentives for node runners. While this can encourage more node runners, it shares the same risks, including token volatility and centralisation issues. Contrary to other ACNs, Nym requires a monthly fee for the network usage that is funding the development behind Nym.

6.3. Censorship arms race

Efforts to circumvent Internet censorship using ACNs are continually met with countermoves by restrictive governments, creating an ongoing technological arms race. Tor is in particular an object of such actions [103] as it is the most popular ACN today. Governments try to block ACNs by targeting their management mechanisms, public relay nodes for client-server ACNs, websites from which you can download a given ACN, or by using deep packet inspection (DPI) to detect ACN-specific traffic. In response, the Tor Project has developed solutions such as bridge relays and pluggable transports to help users access the network even under heavy censorship. While censors have access to significant resources and increasingly sophisticated techniques, the Tor Project adapts by introducing new technical measures, making censorship circumvention a continuous and evolving challenge. Other ACNs have not yet deployed such advanced censorship circumvention techniques; although it is often a part of their roadmap.

7. MULTI-CRITERIA COMPARATIVE ANALYSIS

7.1. Literature-based comparison

Table 7.1: Literature-based comparison of ACNs

	Tor	**I2P**	**Lokinet**	**Nym**
Inception year	2002	2003*	2018	2019
Access	Free	Free	Free*	Paid
Anonymisation technique	Onion Routing	Garlic Routing	Onion Routing	Mix-net
Architecture	client-server	peer-to-peer	client-server	client-server
Supported network and transport protocols	TCP	TCP and UDP	All IP-based	All IP-based
Clearnet support	High	Low	Medium	High
Traffic analysis resistance	Low	Medium	Low	High
Sybil attack resistance	Medium	Low	High	High
Network management	Directory Authorities	DHT	DHT/blockchain	Gateways, blockchain
Topology	Free route	Free route	Free route	Stratified
Intermediate hops number (one way)	3 or 6	0–14	3–17	5
Hidden services support	Yes	Yes	Yes	No
Users number	2,000,000 daily	More than 55,000 daily	More than 1,000,000 monthly	No information
Routers number	10,700	55,000	2,100	700

The table 7.1 summarises the comparison of currently functioning ACNs based on the literature.

Tor was initially deployed in 2002, hence it is the oldest working ACN in this comparison. It is considered a direct successor to the original Onion Routing project, utilising the same anonymisation technique. It is fully free for the end users but relies on the volunteers for donating and running dedicated relays as well as on the financial support from various organisations. Tor supports only TCP-based applications, with a strong emphasis on web applications. Access to the clearnet is fully supported and it is one of the main goals of the network. Sadly, certain websites block traffic originating from the Tor network, most likely due to abuses. On the other hand, other websites, like Facebook or Reddit, have

dedicated hidden services to distinguish traffic coming from within the Tor network. Tor is vulnerable to various traffic analysis attacks due to its low latency with the lack of timing analysis attack defenses or traffic volume obfuscation techniques. It does however protect against browser fingerprinting due to the dedicated Tor Browser. While in theory a Sybil attack could be easily performed in the Tor network, such an anomaly in the number of relays would be quickly spotted and blocked by the centralised Directory Authorities. Circuits in Tor follow free route topology. There are either three hops, for the clearnet traffic, or six hops, for the hidden services-related traffic. These lengths could be in theory changed in appropriate files but it is not easily accessible nor advised by the Tor developers. While the number of Tor users is hard to precisely measure, the estimations from the dedicated metrics subpage vary around 2 000 000 daily users, with a recent spike to 8 000 000 users, although the number came back to the usual 2 000 000 and the reason for such a spike was not publicly given. The routers' number is roughly constant, adding up to around 10 700 relays with 8 800 publicly-listed ones and around 1 900 bridges.

In theory, I2P can be considered an older network, dating back to 2001 with the Invisible IRC project. However, the major architectural change and move towards universal networking rather than an IRC-specific one was performed in 2003; therefore, it is considered as an inception year in this comparison. Utilising the I2P network is free, and each user acts as a router for other users, unless it is unsafe for him - meaning that he is connecting to the I2P network from a restricted country. The categorisation of restricted countries is performed according to the Freedom House research. I2P uses a slightly modified version of Onion Routing. Thanks to unidirectional tunnels instead of bidirectional ones, it is possible to bundle multiple messages within a single transport via a given tunnel. After leaving the sender's outbound tunnel, the messages can then be distributed to appropriate recipients' inbound tunnels. The modified version was named Garlic Routing. The vast majority of messages are sent separately and are not combined; however, in order to distinguish differences in this anonymisation technique, it is considered as a distinct one. Moreover, the concept of unidirectional tunnels makes I2P only a semi-circuit-switched network with elements of a packet-switched network, as an end-to-end message path is never established. The length of a tunnel is modifiable. By default, the unidirectional tunnels have 3 hops, and there are usually two or three tunnels for a given purpose. The length of any unidirectional tunnel can be set to any value from 0 to 7, with a possibility of probabilistic variations of the length to further distinguish traffic patterns. Even with a tunnel length of 0, the user still has the property of plausible deniability, as it is difficult to distinguish whether a certain traffic is designated for this user or is he just routing it for another user. In total, there can be as little as 0 hops to as much as 14 hops in one-way communication. I2P follows free route topology with remarks mentioned earlier. Multiple unidirectional tunnels with variable lengths and plausible deniability make the traffic analysis more difficult for an external observer, increasing the traffic analysis resistance of the network. It is not, however, fully resistant due to the same reasons as other Onion Routing-based low-latency ACNs. I2P supports both TCP and UDP traffic; however, each type of traffic requires a dedicated tunnel. Tunnels

in the java-based I2P do not allow for a simple UDP-based tunnel; I2PD, a C++ implementation of the I2P router, provides such functionality. Java-based I2P allows only for a UDP-based Streamr tunnel. The I2P is mainly focused on the traffic within the network. In order for traffic to leave a network, an outproxy service needs to be established for a given application. For example, there is a built-in outproxy service for HTTP-based traffic provided by StormyCloud, allowing anonymous web browsing for I2P users. As each type of traffic needs a dedicated outproxy to leave a network and the outproxies are not always fully reliable, the clearnet support is considered as low. Due to its decentralisation, I2P is highly vulnerable to Sybil attacks, as there is almost no mechanism for controlling an unnaturally large number of nodes - a powerful attacker that would be able to launch tens of thousands of nodes would be able to take over the network. There is, however, a very basic anti-Sybil mechanism with checking IP closeness. Currently, there are around 55 000 routers within the I2P network and the number of users might be slightly larger, as users from restricted countries do not participate in routing messages.

Lokinet is a newer approach, existing since 2018 both when it comes to the code release and the whitepaper. In theory, the usage of Lokinet is free; however, the free exit nodes are usually non-functioning, which makes Lokinet unusable for clearnet usage. However, there is a possibility of purchasing access to private exit nodes or hosting custom ones, although it might be problematic from the anonymity point of view as it may link the user who wishes to remain anonymous directly to the exit node that he is using. Lokinet also utilises Onion Routing as an anonymisation technique, however on a lower layer than Tor and I2P do. While Tor supports TCP-based applications and I2P supports both TCP and UDP, Lokinet supports all IP-based protocols, including TCP and UDP but also, for example, ICMP. Hidden services hosted within Lokinet - SNApps, may therefore utilise a wide variety of traffic protocols. Lokinet utilises client-server architecture, free route topology and suffers from the exact same vulnerabilities in terms of traffic analysis as Tor. When it comes to the resistance of a Sybil attack it is more resistant than both Tor and I2P, despite the fact that it also uses decentralised DHT structure, in the form of blockchain, for the network management that in theory prones for the attack. The resistance is associated with a fact of cryptoeconomics - each time a new node is introduced to the network it needs to lock a fixed amount of tokens. With each new node the token pool shrinks, effectively increasing the price of the token and making the attack increasingly expensive with every new node. While the exact number of Lokinet users is difficult to estimate, the Session application, which is the most popular application utilising Lokinet infrastructure, has more than one million users a month - it can be therefore considered as the lower boundary for the number of users. Number of nodes varies around 2100.

The youngest ACN in the comparison is the Nym network which had its first presentation in late 2019, the paper describing it was written two years later, in 2021. The NymVPN, a service that enabled the utilisation of the Nym Mix-net was officially launched in March 2025 and it is available as a paid service. Contrary to previous ACNs Nym is not Onion Routing-based - it utilises Mix-nets with client-server architecture. It is heavily based on another Mix-net design - Loopix and many technical

solutions are derived from there, including stratified topology and traffic analysis resistance mechanisms like cover traffic and Poisson mixing. Nym has fixed-size 5-hop routing. Similarly to Lokinet, Nym provides economic incentives for node runners which not only encourages more people to run router nodes but also enhances Sybil attack resistance. Another similarity is using blockchain for the network management, which in Nym is also joined by gateways in order to charge users for the network usage. Nym provides a decent built-in clearnet support and does not support hidden services. There is no data regarding Nym's number of users and the number of nodes is around 700.

7.2. Experiment-based comparison

Table 7.2: Experiment-based comparison of ACNs

	Tor	**I2P**	**Lokinet**	**Nym**
RTT - clearnet [ms]	240	Not measured	Not measured	3152.92
Jitter - clearnet [ms]	20.05	Not measured	Not measured	1221.12
RTT - hidden services [ms]	393.94	813.27	352.13	N/A
Jitter - hidden services [ms]	43.97	54.25	26.6	N/A
Throughput - hidden services [Kbit/s]	2240	775	3504	N/A
Download speed - hidden services [Kbyte/s]	280	591	438	N/A

Table 7.2 shows results of measurements of ACNs for both clearnet and hidden services traffic. Latency and jitter were measured by a simple Python client-server application, which was accessed through a specific ACN that was measured. In each trial, the measurement was repeated 50 times, and the result was the average across these repeats. Trials were repeated multiple times as well, as a specific measurement may have been taken during the periodic routing path change or suffered from temporary bottlenecks. Throughput initially was measured by iperf3, but as it turned out, it did not give correct results for ACNs and therefore it cannot be considered a suitable tool for measuring throughput in them - it showed a far too low value compared to a simple file download; therefore, the throughput was measured by the file download, which is more accurate, although not ideal. There is, however, a dedicated metric of file download speed which can be different from throughput in case ACN allows an alternative way of downloading files than simple client-server communication. There were also attempts to perform dedicated measurements for UDP; however, as Tor does not support UDP, measuring Nym alone makes no sense in terms of comparing ACNs. There was a similar issue with UDP for hidden services - Tor does not support UDP, and while I2P in theory does, there is no basic UDP tunnel in Java-based I2P; there is one in C++-based I2Pd, although there were difficulties in terms of compatibility with

applications and packet loss, and the measurements were unstable - therefore the measurements were discouraged, leaving Lokinet the only ACN with full UDP support for hidden services. Measurements were performed for both clearnet and hidden services where there was such a possibility. As the Nym network does not support hidden services, it was only included in the clearnet measurements. Tor was included in both clearnet and hidden services. I2P requires a dedicated outproxy for each type of traffic. There were attempts to measure latency and jitter by a similar client-server web-based application using web sockets, but the I2P HTTP outproxy did not handle the websockets properly, while Tor and Nym did not have such issues. There were also attempts to measure it with external tools, but the results were unstable and varied significantly between each trial. Moreover, the outproxy suffered from significant downtimes. Considering all of these aspects, the clearnet measurements for I2P were abandoned as it is clearly not the right network for such traffic. While Lokinet in theory also supports browsing clearnet, the freely available exit nodes were not functioning in any of the attempts to perform measurements. There is a possibility of creating a private exit node or purchasing access to one; however, from the anonymity perspective, it is definitely not desired as the exit node can be directly correlated to an owner and/or vendor and potentially deanonymize the user. Certain Lokinet exit node vendors allow for cryptocurrency payments which may protect users as well as short-living exit nodes but it was not purchased for the sake of measurements due to large Bitcoin fees at the time of performing them and lack of possibility of paying not in crypto, which is understandable considering the risk. Despite cryptocurrency choice, the vendor required information about a purchaser which potentially may lead to a deanonymisation. Such inconveniences and potential threats are considered as not desired and therefore measurements for clearnet in Lokinet were not performed. While Nym is also a paid service, one does not pay for the certain entry/exit node, only for the network access, with many layers of privacy protection as described in the Nym section, therefore the risk is considered significantly lower than Lokinet's. Certain measurements are not present in the results table. For example, packet loss was checked with a simple ping command for Lokinet and Nym. The median value across the measurements was 0%. The result is not present as the comparison is not fair - Lokinet was tested in terms of hidden services, while Nym in terms of clearnet. Another measurement that did not appear was Nym's throughput measurement for two reasons; firstly, there were issues with measuring it via the mechanism that was used for other ACNs and it was evaluated with online tests that were not performed for other ACNs. Secondly, placing this measurement in the hidden services table would not be accurate as it was performed for the clearnet connection; although, considering that connection to the inter-network service provider would involve the same number of hops, the measurements should vary significantly. The results in the tests showed very low throughput, around tens of Kbytes/s, which was expected due to the network architecture.

Results of the measurements were mostly expected based on the literature, although there were certain unexpected aspects. In terms of latency and jitter on clearnet Tor performed surprisingly well, achieving 240 ms of RTT, or 120 ms of latency and 20 ms of jitter. Nym was measured with 3152 ms of RTT, or 1576 ms of latency, and 1221 ms of jitter - significant values but they were expected due

to the Mix-net architecture. With hidden services the lowest RTT and jitter were measured for Lokinet - 352 ms of RTT (176 ms of one-way latency) with 26.5 ms of jitter. Tor was not much slower - it had 394 ms of RTT (197 ms of one-way latency) and 44 ms of jitter. I2P turned out to be the slowest for hidden services in terms of RTT and jitter with 813 ms of RTT (406.5 ms of one-way latency) and 54 ms of jitter. In terms of throughput for hidden services, Lokinet also was measured with the highest value of 3504 Kbit/s, Tor with slightly lower value of 2240 Kbit/s, and I2P with the lowest value of 775 Kbit/s. In the matter of download speed, Tor and Lokinet were assigned with the same value as in throughput and converted to KByte/s, 280 KByte/s and 438 KByte/s respectively, as throughput was measured by a client-server file download and these networks do not have any techniques of optimising file downloading. There is however a significant difference in terms of file download for I2P - due to built-in I2P snark that supports BitTorrent protocol. The download speed is faster than in other ACNs, despite the fact of lower single-link throughput. The median value was 591 KByte/s with 21 peers. It shows the strength in terms of download speed of peer-to-peer file sharing protocols.

7.3. Comparative analysis in terms of use cases and application areas

Within this section ACNs will be evaluated in terms of using them in various use case groups. The scores will then be properly justified. ACNs will be, in general, evaluated with values from 1 to 10, with the possibility to rate 0 in case a certain trait is not present in the network at all. There is also a possibility of scores with halves. The scale is not linear.

7.3.1. Low-latency inter-network communication

Table 7.3: Comparison of ACNs in terms of low-latency inter-network communication use cases

	Tor	**I2P**	**Lokinet**	**Nym**
Low latency (0.25)	7.5	4	8	1.5
Low jitter (0.25)	6	5	8	1.5
High throughput (0.2)	7	4	8	1.5
Both-end anonymity strength (0.15)	5	5	5	8
UDP support with low packet loss (0.15)	0	5	10	10
Weighted sum	5.525	4.55	7.85	3.75

In the table 7.3 Tor, I2P, Lokinet, and Nym were evaluated in terms of using them in use cases requiring the lowest latency in inter-network communication. Lokinet scored 8 in terms of latency as it had the lowest value of RTT. Tor's RTT was not much lower, and therefore it scored 7.5. I2P's RTT was significant - therefore it was assigned a score of 4. However, the highest latency was with the Nym

network, which was assigned 1.5. Half a point indicates that the latency could have been larger as Nym has relatively low latency compared to other Mix-nets. In terms of jitter, Lokinet once again achieved the highest score of 8. In fact, it was the only score that fits in the advised maximum 30 ms of jitter. Tor had a bit higher value of jitter, although higher than the desired 30 ms; therefore it scored 6 out of 10. Although I2P's latency was high, its jitter is relatively low in comparison; however, it was still higher than Tor's, so it was assigned a score of 4. The highest jitter was, of course, associated with the Nym network, which is a desirable property from the anonymity point of view but not necessarily in this use case scenario group. Throughput of Tor, I2P, and Lokinet was sufficient in terms of basic requirements within this group. Lokinet achieved the highest throughput and scored a value of 8, Tor had a slightly lower value and achieved 7 out of 10, I2P had drastically lower throughput than Tor and Lokinet, although still sufficient for basic requirements; therefore it was assigned a score of 4. Nym had the lowest throughput, which is directly associated with its architecture, and was assigned a score of 1.5. In terms of anonymity strength, Tor, I2P and Lokinet scored equal value of 5. While there are some differences between them, they all share onion routing architecture or its variations. On the one hand Tor can be considered as more anonymous as it has a larger anonymity set. On the other hand I2P has better plausible deniability as an observer is not able to distinguish whether a user received traffic for himself or just passed it through. Lokinet can provide anonymity on lower layers. As this paper does not aim to provide an anonymity measure for them, they will be assigned with the same weight as the biggest threat to all of these networks is traffic analysis to which they are all vulnerable. Nym is more resistant to traffic analysis, although it was rated with 8 out of 10 with one point subtracted due to the potentially low anonymity set or at least not publicly known and the second one subtracted due to the current lack of possibility of modifying parameters in a way that would allow more anonymity, as described in the Nym whitepaper. In terms of UDP support, only Lokinet and Nym provide full support, and for that reason, they scored 10 out of 10. I2P provides limited support, with the issues described earlier; as a consequence, it scored 5. Tor does not support UDP at all, and therefore was assigned a 0. The most suitable ACN for the low-latency use case group turned out to be Lokinet due to its consistently low latency and jitter, high throughput, and full support for UDP with low packet loss.

7.3.2. Highest anonymity latency-tolerant

Table 7.4 compares Tor, I2P, Lokinet, and Nym for scenarios where anonymity for the sender is the most important and latency is acceptable. Sender anonymity is the main factor here. Nym got the highest score (8) because its Mix-net design is better at hiding users from traffic analysis. Tor, I2P, and Lokinet have similar weaknesses due to their Onion Routing-based methods, so they all scored 5. More justification was provided within the previous category. Reliable delivery matters a lot when messages are critical. All the networks scored perfectly (10) because they reliably deliver messages. Message persistence, or keeping messages available until received, is important for asynchronous communication.

Table 7.4: Comparison of ACNs in terms of highest anonymity latency-tolerant use cases

	Tor	**I2P**	**Lokinet**	**Nym**
Sender anonymity strength (0.5)	5	5	5	8
Reliable delivery (0.25)	10	10	10	10
Message persistence (0.25)	0	0	10	10
Weighted sum	5	5	7.5	9

Lokinet and Nym already include this feature, so they got 10. Tor and I2P don't have this built-in, needing extra setup; therefore, they got 0. The most suitable ACN for the highest anonymity latency-tolerant use case group turned out to be Nym due to its strong anonymity provided by its Mix-net architecture, reliable message delivery, and built-in message persistence. Nym's design effectively mitigates traffic analysis threats, which are critical in sensitive communications.

7.3.3. Web browsing-based

Table 7.5: Comparison of ACNs in terms of web browsing-based use cases

	Tor	**I2P**	**Lokinet**	**Nym**
Clearnet support (0.25)	9	4	4	10
Censorship resistence (0.25)	9	5	1	1
Low latency (0.2)	8.5	4	8	1.5
Sender anonymity strength (0.2)	5	5	5	8
Web browsing optimisation (0.1)	10	3	6	6
Weighted sum	8.625	4.55	4.85	5.325

Table 7.5 compares Tor, I2P, Lokinet, and Nym for web browsing scenarios.

Clearnet support is the most important aspect here. Nym scored highest (10) because it seamlessly supports browsing the regular internet. Tor is also good (9), with one point subtracted due to blocking exit nodes as described in the chapter about threats. I2P and Lokinet are limited in accessing regular sites, so both scored lower (4); their issues with clearnet support were described. Censorship resistance is important for unrestricted browsing. Tor excels at this (9) because it has many methods to get around censorship, including bridges and pluggable transports. I2P provides moderate resistance (5) as it is more difficult to ban constantly changing peers in the peer-to-peer design compared to long-running servers in client-server design. Lokinet and Nym don't have strong anti-censorship capabilities,

although both acknowledged the need for such a mechanism; therefore both scored 1. Low latency is crucial for comfortable browsing. Tor did very well (8.5), the score was based on both the latency in clearnet and for hidden services. Second one was Lokinet (8). I2P's latency was noticeably higher, scoring 4, and Nym's latency was very high, earning just 1.5. Sender anonymity is important for privacy. Nym has stronger anonymity (8) due to its Mix-net architecture. Tor, I2P, and Lokinet have moderate anonymity (5) because of their onion routing methods as described earlier. Web browsing optimization, like easy setup and fingerprint protection, strongly favours Tor (10). Lokinet and Nym have some optimizations, mostly in terms of ease of setup (6), while I2P lacks significant optimisations and with difficulties for setting up clearnet support, scoring 3. The most suitable ACN for web browsing-based use cases turned out to be Tor, due to its excellent balance of clearnet support, censorship resistance, low latency, great latency/anonymity trade-off and robust optimisations tailored specifically for web browsing.

7.3.4. File sharing-based

Table 7.6: Comparison of ACNs in terms of file sharing-based use cases

	Tor	**I2P**	**Lokinet**
Download speed (0.3)	6	8	7
Both-end anonymity strength (0.3)	5	5	5
Volume correlation resistance (0.3)	3	6	1
File sharing protocols support (0.1)	0	10	0
Weighted sum	4.2	6.7	3.9

Table 7.6 evaluates Tor, I2P, and Lokinet for file-sharing purposes. Nym is not evaluated as it does not provide mutual anonymity for the server and receiver; in other words, hidden services are required for this use case group. Download speed matters greatly here. I2P got the best score (8), Lokinet scored decently (7), while Tor, less suited for large file transfers, got 6. Both-end anonymity, or anonymity for senders and receivers, is similar across Tor, I2P, and Lokinet. All scored equally (5) because of vulnerabilities to traffic analysis. Volume correlation resistance, important in large transfers, is best in I2P (6) due to garlic routing capabilities, many tunnels, plausible deniability, and significant possibilities in terms of changing tunnels' properties, obfuscating the volume pattern. Moreover, utilising peer-to-peer file sharing protocols can further help to obscure traffic patterns as smaller chunks are sent through the network from various sources. Tor has moderate protection (3) like fixed-size cells, and Lokinet lacks effective protection, scoring lowest (1). Support for file-sharing protocols clearly favors I2P (10) because of its support, for example, for BitTorrent with various clients (not only built-in

I2PSnark), but also supports less used protocols like Kad network clients or Gnutella clients. Tor and Lokinet don't have built-in support, and Tor even discourages using BitTorrent over Tor as it is not anonymous; both therefore scored 0. The most suitable ACN for file-sharing-based use cases turned out to be I2P, due to its high download speeds, mostly enabled by peer-to-peer protocols, strong volume correlation resistance through garlic routing and other architectural characteristics, and built-in support for common file-sharing protocols.

7.3.5. Infrastructure security and resilience-based

Table 7.7: Comparison of ACNs in terms of infrastructure security and resilience-based use cases

	Tor	**I2P**	**Lokinet**
Resilience to active attacks (0.3)	8	6	7
Authentication and authorisation mechanisms (0.3)	10	7	0
Server anonymity strength (0.2)	5	5	5
Academic research and maturity (0.2)	9	6	2
Weighted sum	8.2	6.1	3.5

Table 7.7 compares Tor, I2P and Lokinet for infrastructure protection and resilience. Nym is not evaluated as it does not provide a possibility of server anonymity, which is needed for this use case group.

Resilience to active attacks, like denial-of-service, is strongest in Tor (8), benefiting from its central management and history. Lokinet, due to blockchain-based incentives, has decent resilience especially for Sybil attacks; therefore, it scored 7, and I2P, with its decentralized design, scored slightly lower - 6. While I2P is slightly more resilient to DDoS attacks, it is vastly more prone to Sybil attacks as described in previous chapters. Authentication and authorization mechanisms are strongest in Tor (10) because of its established security options. I2P has reasonable mechanisms of encrypted lease sets (7). Lokinet currently doesn't have integrated security features, scoring 0. Server anonymity strength is similar across all networks, scoring moderately (5) due to shared vulnerabilities from onion routing. Academic research and maturity strongly benefit Tor (9) since it is extensively studied and tested. I2P has moderate research backing and maturity, scoring 6. Lokinet, being new with less research, got the lowest score (2). The most suitable ACN for infrastructure security and resilience-based use cases turned out to be Tor, due to its resilience to active attacks, robust authentication and authorisation mechanisms, and extensive academic research and maturity.

8. FUTURE DIRECTIONS

8.1. Common

Certain desired future directions are common for each existing ACN.

1. Increase number of users: Popularity of a given ACN directly influences the size of the anonymity set and therefore the provided level of anonymity. The more users the network has, the more diverse they are, the harder it is to target one specific user as he blends in the crowd. In order for the popularity to increase, an awareness needs to be spread. One way to do so is to perform talks and interviews. While people do care about their privacy, the awareness about privacy by design solutions leaves room for improvement.
2. Post-quantum cryptography: Quantum computers pose a threat for ACNs that do not utilise quantum computers-proof cryptography algorithms. The process of introducing them in ACNs is already ongoing, for example I2P and Nym publicly stated switching to PQC as a part of their roadmap.
3. Fighting censorship: One of the most important things that ACNs need to address is progressive blocking by restrictive countries. In order to address this issue in Tor, bridges were created but with time they are also getting blocked in one way or another which was described before in this section. This leads to an unfair arm race with several countries that have an extensive amount of money they can spend on ACN censorship. Current bridge distribution mechanisms seem to be insufficient and pluggable transports were introduced that will further enhance censorship circumvention capabilities of Tor.

   While other ACNs like Lokinet and Nym face similar blocking, they have not yet implemented mitigation techniques. They can utilise similar mechanisms of bridges and pluggable transports as Tor does or introduce custom censorship circumvention solutions. While I2P seems to be the least censored ACN, most likely thanks to the peer-to-peer architecture and large number of routers in general, it can still be censored via DPI. Maybe censoring regimes consider I2P as a niche solution and therefore unworthy of censoring, although once the network will get more recognition, the more aggressive blocking will likely be imposed. I2P should therefore also propose censorship circumvention techniques.

8.2. Tor

1. Tor Browser development: One of the major features of anonymous communication networks is the number of its users as it directly influences anonymity. Besides the number of them, users should be as diverse as possible. In order to encourage more people to use Tor, the software needs to be as user friendly as possible. A large progress has been made in this field over the years, resulting in easy to use Tor Browser. Sadly it is not yet available on iOS devices. Alternatives

like Orbot need to be utilised, however it lacks the security properties of the Tor Browser.

2. Fighting with blocking exit nodes on websites: Certain websites block traffic that originates from Tor or discriminate against Tor users by other means. An example of a partial discrimination in this field is present in Wikipedia where Tor users can view articles but cannot edit them, despite the fact if they are logged into their accounts or not.
3. Decreasing delays: While performance of relays in terms of committed bandwidth is dependent on the relays' runners, the delays can be improved on the Tor project side. Potentially ongoing rewrite to Rust may help with it as Rust can better utilise concurrency. In theory cryptography could also be improved - it should be examined whether precomputing approaches as described with cMix would improve Tor's performance. Tor could also include UDP support or move down one layer and work within the Network layer. As UDP is getting increasingly popular with QUIC, it can be beneficial in terms of end-user delay.

8.3. I2P

1. Enhancing Sybil attacks resistance: For a long time I2P did not have any solutions for addressing Sybil attacks. The work on it is ongoing and there is a mechanism for detecting such an attack via IP closeness. It is sadly not yet sufficient for a full mitigation of this attack. One of the proposals was to include Proof of Work mechanisms for creating a new identity, making the attack more expensive to perform.
2. Decreasing latency: As was shown with the latency measurements, I2P suffers from a significant latency. I2P included speeding up cryptography as a part of their roadmap, which will have a positive impact on latency. As a part of speeding up they can consider precomputation proposals as in Chaum's cMix. Another aspect that may influence latency is the complexity of the I2P protocol stack - a simplification of it may be considered if decreasing latency by other means do not help.
3. Improving UDP tunnels support: Mainline I2P does not support basic UDP tunnels yet - it is only possible with I2PD - a C++ implementation of an I2P router. Even then the integration is not perfect - it was noticed during the measurements that while iperf3 worked fine with TCP tunnels, it did not work with UDP ones.

8.4. Lokinet

1. Improving documentation: Documentation of Lokinet is relatively modest compared to Tor or I2P. The Lokinet whitepaper puts a significant focus on the token itself, leaving many ACN-related topics undescribed. Even such a basic aspect as the number of hops is not properly explained in the documentation.
2. Decreasing latency: Lokinet is heavily focused on latency. It should be examined whether the number of hops is not too large. While Lokinet has indeed lower latency than Tor, the difference

is not that significant, as was shown in the measurements results. Potentially there is also room for improvement within the used cryptography.

3. Increasing number of nodes: Despite the fact that the number of nodes in Lokinet is not the lowest, it still should be increased. Currently running nodes is associated with a significant cost. While it may be beneficial in terms of Sybil attack resistance, it may discourage some node runners that would like to voluntarily run nodes rather than utilise a for-profit token scheme.

8.5. Nym

1. Increasing number of nodes: As Nym is the youngest network and it was just recently officially launched, it has the lower number of nodes within the network. While the placement of nodes seems to be diversified, the number could have been increased to make the network safer in general.
2. Modifying provided level on anonymity: While both Nym and Loopix paper describe modifying network parameters in a way that would allow for an adjustment of provided anonymity level and latency, it is not modifiable yet.
3. Improve documentation: Despite the fact that the Nym supports a communication fully inside the network it is not yet clear fully how to set up such a communication and there were documentation changes recently. It would be beneficial if for example a whistleblowing platform could have easily set up a service allowing users to access it without leaving the network and at the same time increasing the users' anonymity as the service would receive cover traffic. Nym could have also provided a simple list of available services within the network, if the service provider wishes for his service to be publicly available.

9. EDUCATIONAL DEMONSTRATOR

9.1. Scope of the laboratory

This laboratory is designed to help students understand how anonymous communication networks (ACNs) work in practice, what their main features are, and where they can be used. The lab covers both the theory and practical exercises to show differences between various anonymity solutions, starting from basic proxies and moving to more advanced systems. Students will learn about:

- The difference between anonymity by policy and anonymity by design, and why this matters.
- The basics of Mix-nets and onion routing, and how they affect anonymity.
- The most popular ACNs today: Tor, I2P, Lokinet, and Nym—how they work, what they are good at, and what problems they are facing.
- The real use cases for each of these systems, including web browsing, hidden services, and file sharing.

9.2. Theoretical introduction

Anonymous communication networks are systems built to hide who is talking to whom. Firstly deployed were simple solutions like proxies or remailers, where you had to trust the operator not to reveal your identity. This is called **anonymity by policy**—you are only anonymous if the operator sticks to the rules. History shows this is not enough, as providers sometimes keep logs or are forced to give up user data. These solutions, therefore, cannot be treated as truly anonymous communication networks.

Anonymity by design is a different approach: the network is built so that not even the operator can find out who the users are. This is achieved due to cryptography, usually joined by layered encryption, and careful design. This is the basis for modern ACNs.

The first idea for a strong anonymous network came from David Chaum, who described Mix-nets in the 1980s. A Mix-net network consists of mixes - dedicated servers that receive messages, batch, shuffle (mix), and encrypt them, so that it is hard to link the sender to the receiver. Their main drawback is that they are slow and not good for latency-vulnerable use cases like web browsing. Later, onion routing was invented for lower latency, thanks to its circuit-switched nature - sending data through an established bidirectional circuit, making it possible to browse the web comfortably while staying anonymous; however, this introduced a significant drawback in terms of traffic analysis vulnerability. Nonetheless, onion routing turned out to be a dominant strategy and is widely used today. Mix-nets seem to have been forgotten for some time; however, recently they are appearing more often in the literature, mostly thanks to the Loopix design that allows using Mix-net architecture with relatively low latency, although still higher than the one introduced by the onion routing-based solutions.

The four main ACNs used today are:

1. **Tor**: it is the most popular ACN. It uses onion routing and thousands of volunteer relays. Tor is

good for anonymous web browsing and running hidden websites that can optionally be protected with authorisation mechanisms. Tor is a perfect ACN for accessing normal websites (clearnet) but can be slow or blocked in some countries - that is why they introduced censorship circumvention mechanisms in terms of non-public relays or pluggable transports that can optionally obfuscate the traffic, making it more difficult to classify and block.

2. **I2P**: it is a fully peer-to-peer network using garlic routing, which is a variation of the onion routing. I2P utilises unidirectional tunnels instead of the bidirectional circuits, and users have a lot of freedom in terms of customising tunnels' parameters: they can change the number of inbound/outbound tunnels for specific purposes, make them longer or shorter or even choose variation of tunnel length. All of these aspects makes traffic analysis more difficult as the users have the plausible deniability - it is hard to tell whether a certain user was receving message for himself or was he just passing it through. I2P is good for internal hidden services and especially good for anonymous file sharing due to peer-to-peer file sharing protocols support and volume obfuscation techniques described earlier, but it is not the best tool for accessing the clearnet; although there are outproxy services for HTTP and HTTPS, however they are not always reachable.
3. **Lokinet**: it works at the network layer and acts like an anonymous VPN for any program, not just browsers. It can in theory work with clearnet, the free exit nodes are often non-functioning but there is a possibility of renting dedicated exit nodes from private sellers or setting up custom exit nodes. The main advantage due to the lower-layer design is smaller latency and jitter.
4. **Nym**: It is a new project based on previously mentioned Loopix. It is made to be resistant to powerful attackers and is best for people who need strong anonymity even if the network is slower. It is accessible as a mode in the NymVPN service, making this network the only officially paid network in this list.

9.3. Course of the laboratory

During the lab, students will go through five main practical exercises:

1. **Anonymity by policy vs. anonymity by design**: Students will see how single-hop proxies work, check the logs, and notice the weaknesses. Then, they will switch to an ACN setup and compare the difference.
2. **Mix-nets and Onion Routing**: Students will send messages using both methods and look at the results. They will see how Mix-nets provide better protection against traffic analysis but with the cost of a larger delay.
3. **Browsing clearnet with ACNs**: Students will try to reach regular websites over ACNs, comparing support of the clearnet for each ACN.
4. **Hidden Services**: Students will set up a Tor hidden service, first without, then with authorisation keys. They'll see how this allows access to web pages and SSH services securely.

5. File sharing with ACNs Students will measure download speed for a big file shared over Tor via OnionShare and I2P via I2PSnark. They will compare which is faster and which hides traffic patterns better.

9.4. Laboratory setup

Laboratory should be performed on a Debian-based Linux distribution, ideally on Debian 12/Bookworm. Each workstation should have Python, Docker, Firefox, any non-Firefox browser (i.e. DuckDuckGo browser, Opera or Brave), OnionShare, I2P router, and Lokinet with GUI installed. I2P router and Lokinet should be turned on in order to establish necessary tunnels and paths in advance. I2P's bandwidth parameters should be changed in both router client and I2PSnark client in order to avoid throttling.

9.5. Practical exercises

9.5.1. Anonymity by policy and anonymity by design

Learning outcomes: After finishing this exercise, a student should be able to tell a difference between those approaches, along with the drawbacks and potential threats of anonymity by policy design.

1. Clone the repository from https://github.com/Karakean/Master-Thesis The files related to the educational demonstrator will be within the laboratory/ directory. It will be the working directory for exercise 1 and 2.
2. Run routers as Docker containers:

```
docker compose up --build
```

3. Wait for the routers to start and then run client:

```
python3 client.py
```

4. Select "single hop proxy" mode and send "Hello" message.
5. Enter the single hop proxy:

```
docker exec -it single-hop-proxy /bin/bash
```

6. Read the output.log file.
7. **Question**: What do you see? Do you see any issues with this design? Do you know any services that use this design?
8. Exit the client, re-run it with "anonymous communication network" mode and send the same message as before.
9. Enter the entry node and read the output.log file similarly to how you did it with a single hop proxy.
10. Repeat for the middle node and the exit node.

11. **Question**: What is the difference compared to the single hop proxy? What has improved and what can be worse? Why is the single hop proxy approach known as anonymity by policy and ACN-based approach as anonymity by design?

9.5.2. Mix-nets and onion routing

Learning outcomes: After finishing this exercise a student should be able to tell a difference between ACNs based on Mix-nets and Onion Routing, along with advantages and disadvantages of each design. A student should also be able to name a few use cases for each design.

1. Run routers analogically to the previous exercise.
2. Select "anonymous communication network" mode and send 10 messages with numbers from 1 to 10.
3. Enter each node and read output files.
4. **Question**: Consider an attacker who is watching a destination which receives messages from an exit node along with all users that are sending messages to the anonymous communication network. Do you see any issues with current design with regards to such an attacker?
5. Change ROUTER_MODE from "onion-routing" to "mix-net" in the .env file.
6. Re-run the docker containers.
7. Once again select "anonymous communication network" mode and send 5 messages with numbers from 1 to 5.
8. Enter each node and read output files.
9. **Question**: What do you see? Justify such a state.
10. Send further 5 messages from 5 to 10.
11. Enter each node and read output files.
12. **Question**: What do you see? Justify such a state.
13. **Question**: What are the main differences between the Mix-nets and Onion Routing? Which design will be more suitable in case of asynchronous communication or whistleblowing and which will be more suitable in latency-vulnerable use cases such as web browsing or instant messaging?

9.5.3. Browsing clearnet with ACNs

Learning outcomes: After finishing this exercise a student should be able to configure ACNs for browsing clearnet. A student should be able to point out a preferable ACN for this application area.

1. Download and install Tor Browser https://www.torproject.org/download/
2. Turn on Tor Browser and enter https://example.com/
3. **Question**: Relays from which countries are used within your circuit that is used for this website? Were the actions necessary to visit the website complex and/or difficult?
4. Open Firefox browser.
5. Open the *application menu* and go to *settings*.

6. Scroll down to the *Network Settings* and open them.
7. In the *Configure Proxy Access to the Internet* section choose radio button *Manual proxy configuration*
8. In the *HTTP Proxy* textbox write 127.0.0.1 with port 4444.
9. Click the checkbox *Also use this proxy for HTTPS*.
10. In the section *No proxy for* write localhost, 127.0.0.1
11. Save the changes.
12. Go to the address *about:config*.
13. Ensure that *media.peerConnection.ice.proxy_only* property is set to *true*.
14. Change property *keyword.enabled* to *false*.
15. Restart Firefox browser.
16. Try to reach https://example.com/ from the Firefox browser. Try to refresh it a few times.
17. **Question**: Is it reachable? If not - what may be the reasons behind it? Were the actions necessary to visit the website complex and/or difficult?
18. Open the Lokinet GUI and make sure the state is *Connected to Lokinet*. If not, click the power button.
19. In order to make sure that the connection to Lokinet is established run:

```
ping directory.loki
```

    If it works, it means that the Lokinet works properly.
20. **Question**: Can ping work over Tor or I2P?
21. From the GUI turn on the VPN mode.
22. **Question**: Were the actions necessary to turn on the VPN mode complex and/or difficult? Does it work? If not - what may be the reasons behind it? What might help?
23. **Question**: Based on this exercise - which ACN do you find the most suitable for clearnet web browsing?

9.5.4. Hidden services

Learning outcomes: After finishing this exercise a student should be able to configure Hidden Services within the Tor network. A student should be able to name their advantages and potential use cases.

Initial Tor setup

In order to create Hidden Services you first need to install necessary packages.

1. First update packages. Run:

```
apt update
```

2. Install necessary packages:

```
apt install -y apt-transport-https gnupg
```

3. Get current debian release. Run:

```
lsb_release -c | awk '{print $2}'
```

4. Create apt sources list for Tor:

```
tee /etc/apt/sources.list.d/tor.list > /dev/null <<EOF
  deb      [signed-by=/usr/share/keyrings/deb.torproject.org-keyring.gpg]
       https://deb.torproject.org/torproject.org <DISTRIBUTION> main
  deb-src [signed-by=/usr/share/keyrings/deb.torproject.org-keyring.gpg]
       https://deb.torproject.org/torproject.org <DISTRIBUTION> main
EOF
```

   Where <DISTRIBUTION> should be replaced with the previous step's output.

5. Verify Tor packages:

```
wget -qO- https://deb.torproject.org/torproject.org/
    A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89.asc | gpg --dearmor | tee /usr/share/keyrings
    /deb.torproject.org-keyring.gpg >/dev/null
```

6. Now install Tor packages:

```
apt update && apt install tor deb.torproject.org-keyring
```

HTTP

1. Create a directory with a student ID number as a name. Create index.html with visible ID inside the web page body.
2. Run python-based web server:

```
python3 -m http.server 80 &
```

3. Create necessary directory for the Hidden Service:

```
mkdir /var/lib/tor/laboratory_onion_service_website
```

4. Set proper permissions and ownership of the tor directory:

```
chown -R debian-tor /var/lib/tor/
chmod 700 -R /var/lib/tor
```

5. Modify Tor configuration under /etc/tor/torrc. Add these two lines:

```
HiddenServiceDir /var/lib/tor/laboratory_onion_service_website/
HiddenServicePort 80 127.0.0.1:80
```

   The first line specifies the directory where the Hidden Service configuration will be stored. The second line maps a hidden service port to a local port on the localhost interface.

6. In order for the changes to apply, Tor needs to be restarted:

```
systemctl restart tor
```

7. If everything went successfully then onion address should be visible after running this command:

```
cat /var/lib/tor/laboratory_onion_service_website/hostname
```

8. If something went wrong and the hostname file does not exist, you can find logs with below command:

```
journalctl -e -u tor@default
```

9. Reach onion address via Tor Browser. Due to the NAT/hole punching it is not necessary to open ports on a firewall.
10. **Question**: What properties do Hidden Services have? What use cases for them can you name?

HTTP with authorisation key

In this section you will add additional authorization for the web service.

1. Install necessary packages:

```
apt update && apt install -y openssl basez
```

2. Generate private key:

```
openssl genpkey -algorithm x25519 -out
    /tmp/website_key.prv.pem
cat /tmp/website_key.prv.pem | grep -v "␣PRIVATE␣KEY" |
    base64pem -d | tail --bytes=32 | base32 | sed 's/=//g' >
    /tmp/website_key.prv.key
```

3. Extract public key from the private key:

```
openssl pkey -in /tmp/website_key.prv.pem -pubout | grep -v "␣
    PUBLIC␣KEY" | base64pem -d | tail --bytes=32 | base32 |
    sed 's/=//g' > /tmp/website_key.pub.key
```

4. Create a directory for authorized clients:

```
mkdir /var/lib/tor/laboratory_onion_service_website/authorized_clients/
```

5. Within the newly created directory create <ID>.auth file, where <ID> should be replaced with your student ID number. Edit this file adding this one line:

```
descriptor:x25519:<base32-encoded-public-key>
```

Where <base32-encoded-public-key> is the public key that was created earlier.

6. Once again, set proper permissions and ownership for newly created files:

```
chown -R debian-tor /var/lib/tor/
chmod 700 -R /var/lib/tor/
```

7. Restart Tor service:

```
systemctl restart tor
```

8. Now try to refresh the Tor Browser. You should be prompted for a key - enter the private key that you created before.
9. There is an alternative way of reaching a Tor Hidden Service. It allows you to omit entering a private key via checkbox. First you need to create an appropriate directory for storing authorisation credentials:

```
mkdir -p /var/lib/tor/onion_auth
```

10. Then you need to edit configuration file at /etc/tor/torrc and add following line:

```
ClientOnionAuthDir /var/lib/tor/onion_auth
```

Keep in mind that configurations should be separated with an empty line - therefore make an empty line under the hidden service lines you have created before and then paste the ClientOnionAuthDir line.

11. Within the newly created authorisation directory create a file named *<ID>_website.auth_private*, where <ID> is your student ID number.
12. Edit this file and add a following line, replacing marked sections with proper values:

```
<56-char-onion-addr-without-.onion-part>:descriptor:x25519:<x25519
    private key in base32>
```

13. Set proper ownership and permissions:

```
chown -R debian-tor /var/lib/tor/
chmod 700 -R /var/lib/tor/
```

14. Restart Tor in order to apply changes:

```
sudo systemctl restart tor
```

15. Reach the website via torsocks and curl:

```
torsocks curl http://<your_onion_address.onion>
```

16. **Question**: What use cases do you see for Hidden Services protected with authorisation keys?

SSH

Last stage of this exercise should be done in pairs. One student from the pair will host a Hidden SSH Service and the second one will be a client connecting to it. Steps from 1 to 4 should be performed by the student that will host the Hidden Service.

1. Install SSH server:

```
apt update && apt install -y openssh-server
```

2. Start SSH server:

```
systemctl start ssh
```

3. Now edit configuration file under /etc/tor/torrc, adding below lines:

```
HiddenServiceDir /var/lib/tor/laboratory_onion_service_ssh/
HiddenServicePort 22 127.0.0.1:22
```

   Remember about leaving an empty line between existing configurations.
4. Perform analogical steps for protecting a Hidden Service with authorisation as you did in the previous stage (steps 2-7).
5. The student that will connect to the SSH service needs to perform steps 9-14 analogically as in the previous stage.
6. Client should connect to the SSH server by running the command below:

```
torsocks ssh <USER>@<ONION_ADDRESS.onion>
```

   Where <USER> can be replaced either with root or other existing user. If you choose a root user, remember to check on the server side if the /etc/ssh/sshd_config file contains the line PermitRootLogin with the value set to *yes*.
7. **Question**: What are the benefits of using SSH over Tor?

9.5.5. File sharing with ACNs

Learning outcomes: After finishing this exercise, a student should be able to point out differences between Tor and I2P in terms of file sharing. A student should point out a preferable ACN for this use case.

1. Lab instructor will provide a link to OnionShare with a file of size 500MB. Measure elapsed time for downloading this file.
2. Knowing the file size and elapsed time, calculate the download speed for the Tor network.
3. Lab instructor will provide a magnet link for I2P snark for the same file. Measure elapsed time for downloading this file. Do not remove the file or turn off the I2P once downloaded - make sure the torrent stays as *seeding*.
4. Calculate the download speed for the I2P network.
5. **Question**: Which network was faster? Why? Which network provides a better obfuscation when downloading large files and why?

10. SUMMARY

The main result of this work is that there is no universal anonymous communication network that is optimal for every use case scenario. The choice of the most appropriate ACN always depends on the intended application. For example, Tor typically offers the best experience for web browsing, while networks such as I2P or Nym may be more suitable for file sharing or scenarios requiring the highest levels of anonymity. The theoretical analysis and experimental results in this thesis clearly show that the design, configuration, and even the limitations of each ACN must be considered in relation to specific requirements and use cases. The educational demonstrator developed as part of this work enables students to observe in practice the theoretical differences between the major ACN designs. It also exposes the key vulnerabilities of the most popular networks and demonstrates how alternative designs can mitigate these weaknesses. This practical approach helps explain why certain ACNs are more effective for particular tasks and less so for others. Overall, this thesis provides a solid foundation for anyone interested in learning about or researching anonymous communication networks.

Future research in the field of anonymous communication networks can follow several promising directions. One important area is to monitor and analyse new designs as they emerge, such as Katzenpost and other next-generation Mix-nets. Another valuable direction would be to carry out more sophisticated measurements of ACN performance and anonymity guarantees under different network configurations, including varying the number of hops or relays and studying their impact on both usability and security.

BIBLIOGRAPHY

[1] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity - A proposal for terminology," in *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings* (H. Federrath, ed.), vol. 2009 of *Lecture Notes in Computer Science*, pp. 1–9, Springer, 2000.

[2] D. Cole, "We kill people based on metadata," *The New York Review of Books*.

[3] European Data Protection Supervisor, "Data protection." `https://www.edps.europa.eu/data-protection/data-protection_en`. Accessed: 2025-04-20.

[4] M. Eddy, "7 vpn services found recording user logs despite 'no-log' pledge," *PCMag*, July 2020. Accessed: 2025-04-20.

[5] S. Crocker, "Host software." RFC 1, Apr. 1969.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, p. 149–160, August 2001.

[7] P. Maymounkov and D. Eres, "Kademlia: A peer-to-peer information system based on the xor metric," in *Lecture Notes in Computer Science 2429*, vol. 2429, 04 2002.

[8] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, p. 84–90, February 1981.

[9] M. M. Helmers, "E-mail discussion groups and academic culture," *CMC Magazine*, September 1997. Accessed: 2025-04-20.

[10] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, p. 65–75, January 1988.

[11] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous connections and onion routing," *IEEE Journal on Selected Areas in Communications*, vol. 16, p. 482–494, May 1998.

[12] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Communications of the ACM*, vol. 42, p. 39–41, February 1999.

[13] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[14] K. Jefferys, S. Harman, J. Ross, and P. McLean, "Loki: Private transactions, decentralised communication," Tech. Rep. Version 3, Loki Project, July 2018. Accessed: 2025-04-20.

[15] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, "The loopix anonymity system," in *26th USENIX Security Symposium (USENIX Security 17)*, (Vancouver, BC), pp. 1199–1216, USENIX Association, August 2017.

[16] C. Diaz, H. Halpin, and A. Kiayias, "The nym network: The next generation of privacy infrastructure," Tech. Rep. 1.0, Nym Technologies SA, February 2021. Accessed: 2025-04-20.

[17] J. Boyan, "The anonymizer: Protecting user privacy on the web," *CMC Magazine*, September 1997. Accessed: 2025-04-20.

[18] P. Baran, "On distributed communications: Ix. security, secrecy, and tamper-free considerations," Research Memorandum RM-3765-PR, RAND Corporation, Santa Monica, CA, 1964. Accessed: 2025-04-20.

[19] C. Gulcu and G. Tsudik, "Mixing e-mail with babel," in *Proceedings of Internet Society Symposium on Network and Distributed Systems Security*, NDSS-96, p. 2–16, IEEE Comput. Soc. Press, 1996.

[20] D. Kesdogan, J. Egner, and R. Büschkes, "Stop-and-go MIXes: Providing probabilistic anonymity in an open system," in *Proceedings of Information Hiding Workshop (IH 1998)*, Springer-Verlag, LNCS 1525, 1998.

[21] A. Pfitzmann, B. Pfitzmann, and M. Waidner, *ISDN-Mixes: Untraceable Communication with Very Small Bandwidth Overhead*, p. 451–463. Springer Berlin Heidelberg, 1991.

[22] O. Berthold, H. Federrath, and S. Köpsell, *Web MIXes: A System for Anonymous and Unobservable Internet Access*, p. 115–129. Springer Berlin Heidelberg, 2001.

[23] M. Rennhard and B. Plattner, "Introducing morphmix: peer-to-peer based anonymous internet usage with collusion detection," in *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, CCS02, p. 91–102, ACM, November 2002.

[24] M. J. Freedman and R. Morris, "Tarzan: a peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM conference on Computer and communications security*, CCS02, ACM, November 2002.

[25] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: design of a type iii anonymous remailer protocol," in *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, SECPRI-03, p. 2–15, IEEE Comput. Soc, 2003.

[26] D. Chaum, *Blind Signature System*, pp. 153–153. Boston, MA: Springer US, 1984.

[27] C. Diaz and A. Serjantov, "Generalising mixes," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)* (R. Dingledine, ed.), pp. 18–31, Springer-Verlag, LNCS 2760, March 2003.

[28] A. Serjantov, R. Dingledine, and P. Syverson, *From a Trickle to a Flood: Active Attacks on Several Mix Types*, p. 36–52. Springer Berlin Heidelberg, December 2002.

[29] G. Danezis and B. Laurie, "Minx: a simple and efficient anonymous packet format," in *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, CCS04, p. 59–65, ACM, October 2004.

[30] E. Shimshock, M. Staats, and N. Hopper, *Breaking and Provably Fixing Minx*, p. 99–114. Springer Berlin Heidelberg, July 2008.

[31] G. Danezis and I. Goldberg, "Sphinx: A compact and provably secure mix format," in *2009 30th IEEE Symposium on Security and Privacy*, p. 269–282, IEEE, May 2009.

[32] C. Diaz, S. J. Murdoch, and C. Troncoso, *Impact of Network Topology on Anonymity and Overhead in Low-Latency Anonymity Networks*, p. 184–201. Springer Berlin Heidelberg, 2010.

[33] G. Danezis and L. Sassaman, "Heartbeat traffic to counter (n-1) attacks," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2003)*, October 2003.

[34] Katzenpost Project, "Katzenpost: Anonymous communication for everyone," 2025. Accessed: 2025-04-20.

[35] E. J. Infeld, D. Stainton, L. Ryge, and T. Hacker, "Echomix: A strong anonymity system with messaging," *arXiv preprint arXiv:2501.02933*, 2025. Accessed: 2025-06-01.

[36] J. van den Hooff, D. Lazar, M. Zaharia, and N. Zeldovich, "Vuvuzela: scalable private messaging resistant to traffic analysis," in *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, p. 137–152, ACM, October 2015.

[37] A. Kwon, D. Lazar, S. Devadas, and B. Ford, "Riffle: An efficient communication system with strong anonymity," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, p. 115–134, December 2015.

[38] D. Chaum, D. Das, F. Javani, A. Kate, A. Krasnova, J. De Ruiter, and A. T. Sherman, "cmix: Mixing with minimal real-time asymmetric cryptographic operations," in *Applied Cryptography and Network Security* (D. Gollmann, A. Miyaji, and H. Kikuchi, eds.), pp. 557–578, Springer International Publishing, 2017.

[39] M. Waidner and B. Pfitzmann, *The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability*, p. 690–690. Springer Berlin Heidelberg, November 1990.

[40] E. G. Sirer, M. Polte, and M. Robson, "Cliquenet: A self-organizing, scalable, peer-to-peer anonymous communication substrate," Technical Report TR2001, Cornell University, Ithaca, NY, 2001. Accessed: 2025-04-20.

[41] Electronic Privacy Information Center, "Carnivore surveillance system." `https://epic.org/privacy/carnivore/`. Accessed: 2025-04-20.

[42] S. Goel, M. Robson, M. Polte, and E. G. Sirer, "Herbivore: A Scalable and Efficient Protocol for Anonymous Communication," Tech. Rep. 2003-1890, Cornell University, Ithaca, NY, February 2003.

[43] S. Dolev and R. Ostrobsky, "Xor-trees for efficient anonymous multicast and reception," *ACM Transactions on Information and System Security*, vol. 3, p. 63–84, May 2000.

[44] P. Golle and A. Juels, *Dining Cryptographers Revisited*, p. 456–473. Springer Berlin Heidelberg, 2004.

[45] H. Corrigan-Gibbs and B. Ford, "Dissent: accountable anonymous group messaging," in *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, p. 340–350, ACM, October 2010.

[46] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale," in *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 179–192, October 2012. Accessed: 2025-04-20.

[47] H. Corrigan-Gibbs, D. Boneh, and D. Mazières, "Riposte: An anonymous messaging system handling millions of users," in *2015 IEEE Symposium on Security and Privacy*, pp. 321–338, 2015.

[48] R. Anderson, "The eternity service," in *Proceedings of Pragocrypt '96*, 1996.

[49] A. Back, "The eternity service," *Phrack Magazine*, vol. 7, September 1997. Accessed: 2025-04-20.

[50] I. Goldberg and D. Wagner, "TAZ servers and the rewebber network: Enabling anonymous publishing on the world wide web," *First Monday*, vol. 3, August 1998.

[51] R. Dingledine, M. J. Freedman, and D. Molnar, "The free haven project: Distributed anonymous storage service," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability* (H. Federrath, ed.), Springer-Verlag, LNCS 2009, July 2000.

[52] M. Waldman, A. Rubin, and L. Cranor, "Publius: A robust, tamper-evident, censorship-resistant and source-anonymous web publishing system," in *Proceedings of the 9th USENIX Security Symposium*, pp. 59–72, August 2000.

[53] M. Waldman and D. Mazières, "Tangler: a censorship-resistant publishing system based on document entanglements," in *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001)*, pp. 126–135, November 2001.

[54] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability* (H. Federrath, ed.), vol. 2009 of *Lecture Notes in Computer Science*, pp. 46–66, Springer-Verlag, LNCS 2009, July 2000.

[55] Freenet Project Inc., "Freenet manual." `https://freenet.org/resources/manual/`, 2024. Accessed: 2025-04-20.

[56] K. Bennett, T. Stef, C. Grothoff, T. Horozov, and I. Patrascu, "The gnet whitepaper," *unknown*, June 2002.

[57] W. Dai, "Pipenet 1.0." Post to Cypherpunks mailing list, January 1998.

[58] P. Boucher, A. Shostack, and I. Goldberg, "Freedom systems 2.0 architecture," white paper, Zero Knowledge Systems, Inc., December 2000.

[59] J. Appelbaum and A. Muffett, "The ".onion" special-use domain name." RFC 7686, October 2015. Accessed: 2025-04-20.

[60] DigiCert, "Ordering a .onion certificate from digicert," December 2015. Accessed: 2025-04-20.

[61] Hellenic Academic Research Institutions Certification Authority (HARICA), "Dv certificates for onion websites," April 2021. Accessed: 2025-04-20.

[62] S. Frolov and E. Wustrow, "HTTPT: A Probe-Resistant proxy," in *10th USENIX Workshop on Free and Open Communications on the Internet (FOCI 20)*, USENIX Association, August 2020.

[63] N. P. Hoang, P. Kintis, M. Antonakakis, and M. Polychronakis, "I2p metrics portal." `https://i2p-metrics.np-tokumei.net/overview`. Accessed: 2025-04-20.

[64] N. P. Hoang, P. Kintis, M. Antonakakis, and M. Polychronakis, "An empirical study of the i2p anonymity network and its censorship resistance," in *Proceedings of the Internet Measurement Conference 2018*, IMC '18, (New York, NY, USA), pp. 379–392, ACM, 2018.

[65] A. Linton, "Onion requests: Session's new message routing solution." `https://getsession.org/onion-requests-session-new-message-routing-solution`, January 2020. Accessed: 2025-04-20.

[66] C. Chen, D. E. Asoni, D. Barrera, G. Danezis, and A. Perrig, "Hornet: High-speed onion routing at the network layer," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS'15, p. 1441–1454, ACM, October 2015.

[67] C. Chen, D. E. Asoni, A. Perrig, D. Barrera, G. Danezis, and C. Troncoso, "Taranet: Traffic-analysis resistant anonymity at the network layer," in *2018 IEEE European Symposium on Security and Privacy (EuroSamp;P)*, p. 137–152, IEEE, April 2018.

[68] J. Sankey and M. Wright, *Dovetail: Stronger Anonymity in Next-Generation Internet Routing*, p. 283–303. Springer International Publishing, 2014.

[69] T. Ries, A. Panchenko, R. State, and T. Engel, "Comparison of Low-Latency Anonymous Communication Systems: Practical Usage and Performance," in *Proceedings of the Ninth Australasian Information Security Conference (AISC 2011)*, CRPIT, Vol. 116, (Perth, Australia), pp. 77–86, Australian Computer Society, 2011. Available via ORBilu repository.

[70] S. Winkler and S. Zeadally, "An analysis of tools for online anonymity," *International Journal of Pervasive Computing and Communications*, vol. 11, p. 436–453, Nov. 2015.

[71] A. Montieri, D. Ciuonzo, G. Aceto, and A. Pescape, "Anonymity services tor, i2p, jondonym: Classifying in the dark (web)," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, p. 662–675, May 2020.

[72] K. Shahbar, *Analysis of Multilayer-Encryption Anonymity Networks*. PhD thesis, Dalhousie University, October 2017.

[73] K. Shahbar and A. N. Zincir-Heywood, "Packet momentum for identificationof anonymity networks," *Journal of Cyber Security and Mobility*, vol. 6, no. 1, p. 27–56, 2017.

[74] A. Montieri, D. Ciuonzo, G. Bovenzi, V. Persico, and A. Pescape, "A dive into the dark web: Hierarchical traffic classification of anonymity tools," *IEEE Transactions on Network Science and Engineering*, vol. 7, p. 1043–1054, July 2020.

[75] A. Ali, M. Khan, M. Saddique, U. Pirzada, M. Zohaib, I. Ahmad, and N. Debnath, "Tor vs i2p: A comparative study," in *Proceedings of the 2016 IEEE International Conference on Industrial Technology (ICIT)*, March 2016.

[76] N. P. Hoang, J. Dalek, M. Crete-Nishihata, N. Christin, V. Yegneswaran, M. Polychronakis, and N. Feamster, "GFWeb: Measuring the great firewall's web censorship at scale," in *Proceedings of the 33rd USENIX Security Symposium (Sec '24)*, USENIX Association, August 2024.

[77] I. Zahorsky, "Tor, anonymity, and the arab spring: An interview with jacob appelbaum," *Peace and Conflict Monitor*, August 2011. Accessed: 2025-04-20.

[78] B. Okunoye, "Censored continent: Understanding the use of tools during internet censorship in africa: Cameroon, nigeria, uganda and zimbabwe as case studies," Tech. Rep. 2020-07-001, OTF, The Tor Project, July 2020. Accessed: 2025-04-20.

[79] Freedom House, "Freedom on the net: Country scores," 2025. Accessed: 2025-04-20.

[80] Facebook, "Title of the note," Year of Publication. Accessed: 2025-04-20.

[81] M. Braga, "Why facebook is making it easier to log on with tor–and other companies should, too," *Fast Company*, November 2014. Accessed: 2025-04-20.

[82] I. Borogan and A. Soldatov, "Russia's bankers become secret policemen," *Center for European Policy Analysis (CEPA)*, November 2023. Accessed: 2025-04-20.

[83] GlobaLeaks Project, "Globaleaks: Free and open-source whistleblowing software," 2025. Accessed: 2025-04-20.

[84] Earth League International, "Wildleaks: The world's first whistleblowing initiative dedicated to environmental crime," 2025. Accessed: 2025-04-20.

[85] Microsoft Corporation, "Microsoft security advisory 2798897: Fraudulent digital certificates could allow spoofing," January 2013. Accessed: 2025-04-20.

[86] BBC News, "Taliban suspends chess over gambling concerns," *BBC News*.

[87] International Telecommunication Union, "Itu-t recommendation g.114: One-way transmission time," May 2003. Accessed: 2025-04-20.

[88] P. K. Sharma, S. Chaudhary, N. Hassija, M. Maity, and S. Chakravarty, "The road not taken: Re-thinking the feasibility of voice calling over tor," 2020.

[89] International Telecommunication Union, "Itu-t recommendation g.729: Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (cs-acelp)," March 1996. Accessed: 2025-04-20.

[90] Microsoft Corporation, "How much bandwidth does skype need?," 2025. Accessed: 2025-04-20.

[91] D. Das, S. Meiser, E. Mohammadi, and A. Kate, "Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two," in *2018 IEEE Symposium on Security and Privacy (SP)*, p. 108–126, IEEE, May 2018.

[92] R. Dingledine, M. J. Freedman, D. Hopwood, and D. Molnar, "A Reputation System to Increase MIX-net Reliability," in *Proceedings of Information Hiding Workshop (IH 2001)* (I. S. Moskowitz, ed.), pp. 126–141, Springer-Verlag, LNCS 2137, April 2001.

[93] N. Gelernter, A. Herzberg, and H. Leibowitz, *Two Cents for Strong Anonymity: The Anonymous Post-office Protocol*, p. 390–412. Springer International Publishing, 2018.

[94] Internet Live Stats, "Total number of websites," 2025. Accessed: 2025-04-20.

[95] The Tor Project, "Tor metrics," 2025. Accessed: 2025-04-20.

[96] K. Loesing, S. J. Murdoch, and R. Dingledine, "A case study on measuring statistical data in the Tor anonymity network," in *Proceedings of the Workshop on Ethics in Computer Security Research (WECSR 2010)*, LNCS, Springer, January 2010.

[97] B. Kahle, "Learning from cyberattacks." `https://blog.archive.org/2024/11/14/learning-from-cyberattacks/`, November 2024. Accessed: 2025-04-20.

[98] G. Owen and N. Savage, "Empirical analysis of tor hidden services," *IET Information Security*, vol. 10, p. 113–118, May 2016.

[99] The Tor Project, "Who uses tor?," 2019. Accessed: 2025-04-20.

[100] F. A. P. Petitcolas, *Kerckhoffs' Principle*, p. 675–675. Springer US, 2011.

[101] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann, "The sniper attack: Anonymously deanonymizing and disabling the Tor network," in *Proceedings of the Network and Distributed Security Symposium - NDSS '14*, IEEE, February 2014.

[102] L. Oldenburg, G. Acar, and C. Diaz, "From "onion not found" to guard discovery," *Proceedings on Privacy Enhancing Technologies*, vol. 2022, p. 522–543, November 2021.

[103] R. Dingledine, "The tor censorship arms race: The next chapter." Presentation at DEF CON 27, August 2019. Accessed: 2025-04-20.

LIST OF FIGURES

LIST OF TABLES