TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

ITC70LT

Stefano Panarese - 165624

NESSUNO: A FRIEND-TO-FRIEND ANONYMOUS COMMUNICATION PROTOCOL

Master Thesis

Supervisor: Olaf Maennel
Ph.D.

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

ITC70LT

Stefano Panarese - 165624

NESSUNO: SÕBRALT-SÕBRALE ANONÜÜMNE KOMMUNIKATSIOONI PROTOKOLL

Magistritöö

Juhendaja: Olaf Maennel
Ph.D.

Tallinn 2018

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Stefano Panarese

May 6, 2018

Abstract

This thesis presents Nessuno, a distributed friend-to-friend anonymous communication protocol. This study starts with introducing the privacy and anonymity issues emerging from the contexts of those countries with a heavy employment of surveillance. In these contexts, the weaknesses of existing solutions to achieve anonymity in communications are noticeable. Inspired by some of the approaches used in other solutions such as Freenet, Retroshare, Bitmessage, PGP and mix networks, Nessuno adopts a flooding mechanism to forward the messages to the rest of the network and asymmetric encryption to secure the communication's integrity and confidentiality. The friend-to-friend policy prevents the user from establishing direct connection with untrusted nodes, hence carefully choosing trustworthy peers to directly connect with becomes vital. This thesis also provides guidelines for the implementation of a client for Nessuno and a basic proof-of-concept is available on GitHub for the community. The results coming from a theoretical performance test highlight the performance trade-off that induced by the adoption of a flooding mechanism, however, small changes in the protocol could lead to considerable improvements in its performance.

This thesis is written in English and is 36 pages long, including 8 chapters and 5 figures.

Annotatsioon

Nessuno: sõbralt-sõbrale anonüümne kommunikatsiooni protokoll

Antud lõputöö esitleb Nessunot, hajutatud sõbralt sõbrale anonüümne kommunikatsiooni protokoll. Antud lõputöö tutvustab privaatsus ja anonüümsusprobleeme nendes riikides kus rakendatakse tugevat töötajate järelvalvet. Selles kontekstis on eksisteerivate lahenduste anonüümsus märgatavalt nõrgem. Inspireeritud teistest lahendustest näiteks nagu Freenet, Retroshare, Bitmessage, PGP ja mix networks, siis Nessuno kasutab ülevoolavus mehhanismi, et edastada sõnumeid kogu võrgule ning asümmeetrilist krüpteeringut, et turvata kommunikatsiooni terviklikkust ja konfidentsiaalsust. Sõbralt sõbrale poliis väldib seda, et kasutajad saaksid otse ühendust võtta mitteusaldusväärsete klientidega, seega hoolikalt usaldusväärsete klientide valimine muutub oluliseks. Antud lõputöö estiab ka juhised kuidas Nessuno klienti üles seada ning algeline ideetõestus on saadaval GitHubis kogu turvalisuse kommuunile. Tulemused mis on saadud teoreetilistest jõudluse testidest näitavad jõudluse kompromisse, mis tekib ülevoolavuse mehhanimsit ning väiksed muudatused protokollis võivad oluliselt jõudluse näitajaid tõsta.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 36 leheküljel, 8 peatükki, 5 joonist.

List of abbreviations and terms

FBI	*Federal Bureau of Investigation*
NSA	*National Security Agency*
GCHQ	*Government Communications Headquarters*
BND	*Bundesnachrichtendienst*
P2P	*Peer to Peer*
CC	*Common Criteria*
ISP	*Internet Service Provider*
IP	*Internet Protocol*
TTL	*Time to Live*
F2F	*Friend to Friend*
RSS	*Rich Site Summary*
XXE	*XML External Entity*
SS7	*Signalling System No. 7*
XMPP	*Extensible Messaging and Presence Protocol*
XML	*Extensible Markup Language*
PGP	*Pretty Good Privacy*
DARPA	*Defense Advanced Research Projects Agency*
I2P	*Invisible Internet Project*
UDP	*User Datagram Protocol*
TCP	*Transmission Control Protocol*
NAT	*Network address translation*
TOFU	*Trust on first use*
SSH	*Secure Shell*
IV	*Initialization vector*
AES	*Advanced Encryption Standard*
CBC	*Cipher Block Chaining*
UDT	*UDP-based Data Transfer*
FIFO	*First in last out*

UI	*User Interface*
API	*Application programming interface*
CDN	*Content delivery network*

Table of contents

List of figures

1 Introduction

As the Internet evolves, new and more powerful ways to communicate digitally keep emerging. The digital era we now live in has unleashed a wide range of opportunities for people to connect to and communicate with each other, create various groups and networks, and share information and digital property such as pictures and music. These new ways of communicating are both tending the users' various – and increasingly diverse – needs better than ever before, but also posing as a potential threat to their privacy at the same time. The political significance of the Internet has led to states building architectures aimed to control this flow of communication, information and property with tools such as laws and regulations, and by establishing cybersecurity units in agencies such as FBI and NSA. These measures have essentially given life to the Internet censorship.

The undisputed pioneer of Internet censorship and surveillance is China. The Chinese government has built a system in 1998 known as the "Great Firewall of China", by using firewalls at the national choke points of the Internet, to establish what are essentially its digital national borders [1]. Nowadays, this approach to Internet control has been legitimized – at least superficially – by establishing laws and rules to justify filtering and restricting of information flow [2].

In 2013, the Reporters Without Borders filed a report listing the "State enemies of the Internet" [3]. This list consists of countries whose governments apply intrusive mass surveillance on their citizens, with some of the top countries being Bahrain, China, Iran, Syria and Vietnam – mainly developing or newly industrialized countries. During the same year, however, Edward Snowden's revelations on NSA's mass surveillance practices came about. It led to raising the US citizens' as well as other Western peoples' awareness of their privacy rights, making online privacy issues one of the most discussed topics in the field of human rights as well as technology.

One of the fundamental human rights linked to this discussion is the freedom of speech, protected by many multilateral treaties and conventions, such as the Universal Declaration of Human Rights and the International Covenant on Civil and Political Rights. Internet censorship gives rise to potential threats to freedom of speech, by restricting what can be said and what information can be shared. In fact, the possibility to stay anonymous online is the only possible way to go around governmental restrictions on free speech. If the possibility to stay anonymous is weakened, one could end up as a victim of threats and prosecution due to sharing of an unpopular or even illegal opinions or information. The Syrian government, for example, frequently traces and tracks dissidents by intruding their private information, in order to prosecute them [4].

A close term with privacy, anonymity is also considered a fundamental right in today's society. In many democratic countries, the government has issued regulations to protect it in every facet along with digital networks [5]. However, in countries where the government has the interest and power to break the anonymity through lawful techniques (court orders and regulations) or unconventional solutions especially in the absence of a democratic authority, protecting the users' anonymity has become a challenging task. Telecommunication technologies have raised a considerable concern about the preservation of privacy [6], but the definition of anonymity differs in relation with the context and the reasons why it is needed.

Although the principle of privacy can be considered in a number of fields such as finance, health, education or public records, this study focuses on two aspects. First, the need of privacy and anonymity coming from political dissidents, journalists and researchers whose freedom of speech has been denied by an oppressive regime with high capabilities to wiretap the digital traffic in order to track and prosecute the subjects. Second, people living in a society that applies mass surveillance justified by protection against terrorism or other threats to the nation.

With this collision of human rights and technology, a new term depicting a sort of activism in cyberspace has been coined: hacktivism [7]. In the Syrian case, these hacktivists were journalists who managed to evade the government's surveillance through anonymous networks, all the while informing the rest of the world of the internal situation, and by this, drawing the international community's attention towards the Syrian problem. Hacktivists can exist through technologies, that allow freedom of speech by creating digital anonymous identities in order to stay in the blind spot of the surveillance measures. Their perception of what is actually secure and private is different from a company offering anonymity and security services, seeing as these companies can be legally forced to give out the data access permission to law enforcement agencies, disclosing all the information supposed to be private [8].

An example of this practice dates back to 2005 when Shi Tao, a Chinese journalist, was jailed for eight years after "leaking state secrets" [9]. The secrets that led the journalist to the arrest were about the 15th anniversary of the Tiananmen Square massacre where the Chinese soldiers opened fire on unarmed civils. The official death toll counted 241 deaths while the numbers that Tao gave to a human rights group was in the order of thousands [10]. What stood out from the news was the fundamental involvement of the company Yahoo that, under a court order, handed over Tao's account information that were subsequently used in the case [11].

The need of journalists from all over the world to protect the identity of their sources has become critical. According to the Ethical Journalism Network, a coalition of more than 60 groups of journalists, editors and press owners, before the digital era, protecting the source of confidential information was a professional matter rather than only ethical [12]. If a journalist promises to a source to grant him anonymity, the journalist has to honour the promise and protect the source, contrarily he would lose his trustworthiness together with other opportunities.

Nowadays this job has to adapt to the new context in which protecting a source is not only a matter of choice but instead, it has become a challenge. After the confidential information that Snowden released, the impression is that the NSA has a counterpart in many other countries like in U.K. where the Government Communications Headquarters (GCHQ) is alleged to use a system called Tempora to intercept every communication data passing through their U.K. network up to 30 days [13]. Likewise, the German's federal intelligence agency (BND) is accused by the whistle-blower in an interview to the Spiegel to be sharing users' information in exchange of monitoring tools [14]. As reported by The Citizen Lab, a group of researchers based in Canada, a software called FinFisher is used to monitor users in 25 countries all over the world and used by the governments to target dissidents from the opposition group [15].

As the public's awareness towards privacy issues keeps rising, and as more whistle-blowers keep exposing governmental surveillance practices, there is a growing need for a truly anonymous way to communicate. Studies have identified a variety of solutions applicable to many different situations and anonymity needs [16]. It has to be considered, however, that in many cases they cannot be applied in a context where the offensive side is represented by a government, as they can demand access to private or public devices containing its citizens' data or force the ISP to disclose further information.

Considering a state-level adversary model with high capabilities (specified in the chapter 1.2.1), the current solutions available for communicating anonymously present a variety of issues that prevent them to be used in a high-risk environment. Some technologies like Signal are legit from the point of view of the cryptographic stability but its architecture is centralized. As a centralized architecture can be vulnerable to the pressure of a government, any technologies relying on a central entity will suffer from the same weakness. Applying a peer-to-peer (P2P) approach is considerably better as a distributed system does not present a single point of failure and allows a better control in terms of security and anonymity. However, after analysing a wide range of anonymous P2P solutions such as Bitmessage, Freenet, Gnutella, Retroshare not only limited to communication but also content and file sharing, it can be concluded that despite claiming security and anonymity, some technologies made contradictory choices in the implementation that favour a wider range of functionality which opens the system to more vulnerabilities as the attack surface increases. Some other projects, like Bitmessage that takes advantage of the Tor hidden services, present a tight coupling with a specific technology that against a state-level adversary becomes unreliable.

This study will present "Nessuno", an anonymous communication protocol that provides peer-to-peer messaging in a non-trusted environment, hiding sender, receiver and content of the message. Nessuno adopts the same concepts employed in other P2P anonymous solutions like the friend-to-friend approach used in Retroshare but aimed to protect the user's identity from a state-level threat. In a friend-to-friend network any user can invite a known user into the network who does not have knowledge of all the other users, moreover he will not be able to make direct communication with any of the other users except the one that invited him in. This private approach to the P2P network aims to give a further layer of invisibility to the users as long as only trusted “friends” are involved.

In order to achieve the user’s identity protection, the protocol makes use of a *flooding* mechanism (see the chapter 3.4) and cryptographic solutions involving asymmetric encryption and signature. The first is used to congest the network with messages, hence preventing an attacker to link a message to a sender or a receiver. The cryptographic protection allows the message to be readable by the intended recipient only and not be tampered by any other subject in the path.

From a technical viewpoint, the goal of Nessuno is to allow a small group of individuals to safely communicate in a closed network despite living under the pressuring presence of a state-level adversary interested in preventing or altering the communication between the users.

The “Related Work” section analyses the existing protocols and systems for anonymous communication, highlighting the underlying reasons for their failure in the example contexts. In the “Description of the Protocol” section, Nessuno is presented alongside with the core concepts which make it an ideal protocol to use to protect the user identity when communicating under surveillance. The “Implementation” chapter contains the guidelines for the development of a client that implements the protocol introduced in this paper. The “Performance” chapter shows an estimation of the performance of the Nessuno when the number of users participating changes under different network conditions. The “Conclusions” section contains a summary of what the solution presented in the paper and lastly, the “Future Work” section gives an insight of the future challenges that Nessuno will tackle.

1.1 Definitions

When dealing with privacy and anonymity it is critical to give them a definition as the concepts involved can be misunderstood. This research will derive these definitions from the Common Criteria for Information Technology Security Evaluation standard (CC) [17], hence the focus will be held in the technological environment with a significant attention to the Internet.

1.1.1 Privacy

The concept of privacy can be spotted in the Universal Declaration of Human Rights as one of the human rights in two different articles:

- **Art. 12**: No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence […]
- **Art. 19**: Everyone has the right to freedom of opinion and expression; this right includes freedom to hold opinions without interference and to seek, receive and impart information and ideas through any media and regardless of frontiers.

In agreement with the CC standard, Alan Westin also defines privacy as "*the claim of individuals, groups or institutions to determine for themselves when, how, and to what extent information about them is communicated to others.*" [18]. The concept of privacy can be further defined according to the sphere of influence it resides.

The context relevant for this study belongs to the *privacy in communications* which deals with the protection of the privacy of telephone, email and other channels to establish a communication. This definition is different than the information privacy, which is not part of the scope of this research, that instead covers the situations involving the processing of personal data and "the individual's claim to control the circulation of data about himself" [19].

As stated in the CC standard, privacy concerns *"user protection against discovery and misuse of identity by other users."* [17] and its requirements are represented by:

- Anonymity
- Pseudonymity
- Unlinkability
- Unobservability

1.1.2 Anonymity

It is clear that according to the definition of privacy above, anonymity is referenced as one of its primary properties. This conception is also described as the property that "*ensures that a user may use a resource or service without disclosing the user's identity. The requirements for anonymity provide protection of the user identity. […] Anonymity requires that other users or subjects are unable to determine the identity of a user bound to a subject or operation.*" [20]. Therefore, the definition of anonymity adopted for this study will be the protection of the identity of the user in any step of the communication process.

1.1.3 Pseudonymity

When considering communication is crucial to identify the user in a different way than his real-world identity. Using a pseudonym is a solution that fits the problem and allow the user to interact with other pseudonyms that shield other users. In this way, the real identities are protected and an unrelated digital identity is created (see Unlinkability below). As stated by the CC, this property "e*nsures that a user may use a resource or service without disclosing g its user identity, but can still be accountable for that use.*".

1.1.4 Unlinkability

The unlinkability property is a precondition for the anonymity and the safeguard of the user's real-world identity. It concerns the absence of information from an attacker's point of view to determine whether a certain activity can be linked to the user or not.

1.1.5 Unobservability

According to the CC standard, unobservability is achieved when the attacker "*cannot determine whether an operation is being performed.*". Some more authors demand for a differentiation with *undetectability* [20] affirming that a process can be unobservable but can still be detected. This study will follow the CC standard definition.

1.2 Assumptions

Since the need for an anonymous protocol is present in different context as explained above, it becomes critical to provide a formal assumption regarding the users of the system and the adversaries that this protocol is intended to resist to.

1.2.1 The adversary

The definition of the adversary is vital for designing an anonymous protocol aimed to resist to it. As every project involving anonymity and censorship-resistance the adversary is highly-skilled, strongly-funded and resourceful whose aim is to intercept and wiretap people's communication, localize the user or deny the communication.
The adversary can be either external or internal to the anonymous communication network. It can be active or passive. In case of an active adversary, it can be referred as *attacker* while, if it applies a passive approach (eavesdropping) the adversary can be defined as *observer*. The adversary can also have a global access or local access depending on the spectre of network under its control.
The primary profile outlined in this paper involves:

- **An attacker with internal limited scope.** The extend of the adversary's capabilities are assumed to allow it to penetrate the network and gain the trust of a limited portion of the network.
- **An observer with global scope.** It is assumed that the adversary can intercept and observe all the traffic from the national (or possibly global) Internet network. The realistic case considers an adversary that can control an Internet Service Provider (ISP) for surveillance purposes.
- **An attacker with limited active scope.** The adversary is assumed to be able to create, tamper or drop Internet traffic. The capabilities however, are limited to distinct IP addresses. This study does not contemplate an adversary that can prevent the Internet access at a national level despite being a realistic risk.
- **An attacker with limited physical scope.** In performing an attack, the adversary is assumed to make use of coercion techniques only if the user has been identified.
- Ability to locate the user given his IP. It is assumed that the adversary is able to locate the user thanks to the control over an Internet Service Provider (ISP). However, this eventuality is possible only after his IP has been found

- **Computational power.** The computational power available for the adversary is considered to be greater than the aggregated power of the entire network. Despite the greater power, the adversary is not able to break the cryptographic concepts that lie behind the protocol.

1.2.2 The users

Further assumptions must be given about the profile of the potential user of the anonymous protocol.

- **Access to the Internet.** It is assumed that every user who uses the system possesses the tool and the possibility to access an Internet connection. It is also assumed that the user is using a computer that has not been compromised as if this precondition would be broken it would make the anonymity core concept inefficient.
- **Different level of trust.** Every user connects willingly to at least one user (called *friend*) in order to access (or create) a network. The friend is assumed to be trusted by the user as this is the fundamental requirement to keep the user safe and hidden by his trusted direct connection to his friends. Every other user who is part of the network and is not a friend of the user is considered not trusted. Nevertheless, the user is willing to have an indirect connection (through his friends) in order to engage a communication with any user of the network whether he is a friend or not. Moreover, it is not assumed that any of the users is trusted by every other user in the network.
- **First contact is out of the system.** In order to access the system a user needs to be directly invited. The invitation is assumed to be carried out offline (physically meeting the user) or through another system that is reciprocally considered trustworthy.
- **Confidential content.** Every message that transits in the network is assumed to be strictly confidential.

2 Related Work

One of the earliest researches trying to satisfy the need for anonymous communications is found in 1981 with the paper from David Chaum “Untraceable electronic mail, return addresses, and digital pseudonym” [21]. Until now, the research has found a wide-range of solutions concerning different shapes of the need of privacy and anonymity. This chapter analyses the most important technologies that inspired and shaped the backbone of the protocol that will be outlined in the following chapters.

2.1 File sharing P2P solutions

The huge demand for a distributed system for file sharing was answered by the successful service called Napster [22]. The system, however, was relying on a central server to index and link the files to their location among the peers. This led to Napster being sued by a record company after the users started using the service to share mp3 files containing music under the company's copyrights [23]. In July 2001, Napster was forced to shut down to comply with the injunction that included a bond of 5 million dollars [24].

Originally developed by the IT company Nullsoft in 2000, Gnutella was the first peer-to-peer network that allowed file sharing without the need of a server, therefore resistant to surveillance and court orders.
In its peer-to-peer network, users connect to each other to share files and information, making all the users equally important for the network. When a user needs a file, it will issue a query message to its neighbours with a TTL (Time-to-leave) header set. The peer that receives the query message forwards it and decreases the TTL by one and forwards it in the same way until the TTL reaches zero. This method used by Gnutella to deliver the message to all the peers in the TTL distance, is called *flooding*.
What ended up Gnutella’s downfalls were scalability and user-traceability. The protocol was meant to be used by a big number of peers but the flaws in its design lead it to take up the entire bandwidth of a 56Kbs Internet connection for 10 queries per second [25].

Gnutella's flaws in its design could also leak information that could be used for tracking users as the TTL (Time-to-Live) header is set to 6 if the sender's is the initiator of the query [26].
Freenet [27], was introduced as a privacy-focused alternative to Gnutella. It works by storing an encrypted snippet of the document to be shared anonymously by each peer. Freenet creates a self-contained network, which makes it isolated from the regular Internet. Inside of this kind of network users can create content without any censorship or limitation, and access it anonymously. Freenet introduced the concept of Friend-to-Friend (F2F) rather than peer-to-peer (P2P) as it can be set to connect only to trusted peers (friends) creating a further layer of isolation. This makes Freenet very hard for the law enforcement to block (although not by default).

Despite having won the SUMA-award in 2015 for "surveillance and censorship-proof Internet solution" [28], it still has some flaws in the design that can allow a content retriever (a peer who wants to access the content) to be traced back "even if a single request message has been issued by the retriever" [29]. This is due to the two-hop routing lookup implemented in Freenet which allows a node to see the location of its neighbour's neighbours.
RetroShare, another communication and file-sharing solution, follows the footsteps of Freenet by utilizing the same F2F concept in its backbone architecture to create an isolated network. Its design allows multiple services to run in the network: text chat, voice/video calls, email, social network and file sharing also via RSS feeds called channels. However, the wide-range of features that the system can provide, increases the attacker's attack surface threatening anonymity and security of Retroshare's users. In 2016 Elttam [30], an IT security company, shared a security review of Retroshare's codebase highlighting critical vulnerabilities such as XXE Injection and remote heap overflow [31].

2.2 Chat protocols

2.2.1 Signal and MTProto

Signal (formerly known as TextSecure) and MTProto are cryptographic protocols that allow encrypted communication between users in an instant messaging application. It

incorporates end-to-end encryption to provide security and confidentiality, and a central server to manage traffic and authentication. Without digging into the solid cryptography approach used in both Signal and MTProto, the centralized structure of the systems is the kind that cannot provide any sort of anonymity for the users when the attacker is a state or its agency. It is not common for security agencies such as the NSA to require companies to provide them with a backdoor to their system. According to the journalist Glenn Greenwald, the NSA routinely receives devices before they are exported to international customers in order to implant backdoor surveillance tools [32]. Also, the fact that companies can publicly deny any cooperation, does not mean that we know what is happening behind the curtains.
The biggest weakness of these protocols is the fact that their client implementations (WhatsApp, Telegram, Signal as the most famous) rely on mobile numbers as authentication opening the system to the drawbacks that affect SS7. Signalling System No. 7 (SS7) is the standard protocol used in telecommunication networks for transferring calls, messages and other information. Law enforcement agencies can access the SS7 network to intercept calls, messages and location of a given telephone number as part of surveillance. This system also presents critical vulnerabilities as demonstrated by the researcher Karsten Nohl for the CBS in 2016 when he tracked the congressman Ted Lieu just by his phone number [33].
The impact of this vulnerabilities is not limited to the user’s privacy but it also threats his security as if the SS7 network is not reliable, the two-factor authentication that represents an important defence against hackers, becomes easier to exploit.

2.2.2 XMPP

Extensible Messaging and Presence Protocol (XMPP) is an XML-based communications protocol that relies to a client-server architecture but with a decentralised model. This means that every user can run an XMPP server and there are no authoritative central servers like in the above presented examples. Sometimes, this creates a sense of security and anonymity which can be misleading when the server has not been set up in a secure way. As this is still a centralized protocol, the scenario presented in this research would expose it to all the weaknesses that a decentralised protocol would solve.

2.2.3 Ricochet

Ricochet is a peer-to-peer instant messaging software that provides complete anonymity by using end-to-end encrypted and metadata free communications.
At the core of Ricochet are Tor hidden services. Every node runs a Tor hidden service and the traffic is routed in the Tor network and it never leaves it.
Although it provides a high level of anonymity as IPs cannot be spoofed, Ricochet relies too heavily on Tor hidden services. This opens the system to every security flaw that might hide in the hidden service itself [34].

2.2.4 Bitmessage

Bitmessage is a very interesting project that provides a P2P encrypted protocol used in their own P2P network [35]. A user can send a message to an address identified as an alphanumeric string, after which the message is replicated in the entire network, and finally encrypted in a way that only allows the actual recipient to decrypt it. This mechanism provides anonymity for both the sender and the receiver who will remain hidden until somebody (the recipient) decrypts the message. Therefore, every node will try to decrypt the message and will fail if the message was not addressed to them.
Considering the size and the computational power of the network, the system also implements a proof-of-work system to prevent flooding.
The proof-of-work system was formalised by Ari Juels and Markus Jakobsson in their paper in 1999 [36] but was first introduced in anarticle [37] by Cynthia Dwork and Moni Naor in 1993. The system is based on a task (e.g. creating a hash) which is hard to compute in terms of time and cost but easy for others to verify. In the case of Bitmessage, it uses the proof-of-work to prevent denial-of-service attacks that need an extraordinary computational power that cannot be afforded by most attackers.

Bitmessage can be extremely powerful for big groups of people that can take advantage of the large network to achieve anonymity when communicating. However, in case of a smaller group, users might not want to share their connection and messages with the whole network. The benefits of Bitmessage's computational power could be reconsidered as this lead to receiving packets from an untrusted source.
Good evidence of this danger was found on the 13th February 2018, when the Bitmessage group disclosed a remote code execution vulnerability that could de-anonymize the users behind a node [38].

2.2.5 PGP

Pretty Good Privacy (PGP) is a program used for encryption of data communication that provides authentication and privacy through cryptographic primitives that take advantage of the asymmetric public/private key pair.
Developed in 1991 [39] by Phil Zimmermann, it can be adopted for signing and encrypting simple texts, emails or files and it is designed for high-latency communications.
Despite being described as close to military-grade encryption by the cryptographer Bruce Schneier in 1995 [40], PGP has been proven to be vulnerable to a wide range of practical and theoretical attacks such as man-in-the-middle attacks or cryptanalysis attacks [41]. Besides, the two main issues of PGP are:

- **Missing forward secrecy**. It makes sense for an attacker to collect all the encrypted messages as compromising the private key will allow the entire record of messages to be readable.
- **Leaking of metadata**. A PGP encrypted message still leaks information that allow a well-resourced attacker to trace the sender of the message and its recipient as demonstrated by the senior researcher Nicholas Weaver from the International Computer Science Institute in a pitch [42] at the security conference Usenix Enigma.

2.3 Anonymous routing systems

2.3.1 Mix networks

The concept of mix network (mixnet) was coined in the 1981 by Chaum [43]. A mixnet is a system made of nodes (mixes) that, based on permutation and cryptography, can receive a certain number of messages in input before sending the encoded batch of messages as new input to the other mixes in the network. This combined process hides the sender and mix the messages in input to achieve the unlinkability between the incoming and outgoing messages. High-latency mixnets like Chaum's are more suitable for email communications, as the delay introduced by each mix denies timing attacks. On the other hand, low-latency networks are needed for browsing and instant messaging.

2.3.2 Onion routing

Onion routing was developed first in 1990 by the U.S. Naval Research Laboratory, then by DARPA (Defense Advanced Research Projects Agency) and finally patented by the Navy in 1998. This mechanism was designed to protect the intelligence communications and prevent the traffic to be intercepted [44].
While the mix networks security benefits come from mixing of the content among the mixes, onion routing preselects the nodes in the network through which the message will be routed. The chain of selected nodes is called circuit.
When a client sends a request to a server, the message is encrypted with the public keys of the nodes in the circuit starting from the last node before the destination. Every layer of encryption contains the information about what node should be next (or the server destination if the circuit is over).
Every time the final encrypted message reaches a node, a layer of encryption related to the specific node is read and the rest is sent to the next destination. When the server sends data back, the nodes will travel through the circuit back to the client. As every node is aware of the previous and the next one, the weakness of this system is represented by the first and last node, leaking respectively the sender (client) and the receiver (server). A global observer able to see the traffic from to the first node and from the last node can correlate the incoming and outgoing traffic and trace the user.

2.3.3 Tor

The Tor project can be considered the second generation of onion routing. Developed in 2004, it is designed to create a network of relays aimed to route anonymous web traffic [45]. Tor applies the principle of confidentiality on the onion routing by negotiating a symmetric key with each node. In the first phase, the client generates the keys to be used with each node in the circuit randomly chosen from a public list of volunteer servers. Then, the client sends the message with the routing information to the first node that decrypts the first layer containing the identity of the next node (as in the onion routing) and the symmetric encryption key that will be used in the communication. At this point the communication can start and the messages are encrypted with a symmetric key achieving low-latency, vital to support browsing and instant messaging.
Tor can provide protection against traffic analysis attacks as the communication is encrypted until the last node. However, the communication between the exit node (last node) and the destination of the message is clear and it can potentially break the

sender's anonymity if the message contains identifying information. The risk of being de-anonymised is the highest when the attacker controls one of the nodes [46]. Recently, a new algorithm to select a trustworthy path of nodes has been developed by researchers at the Naval Research Laboratory and Joan Feigenbaum from Yale University to avoid man-in-the-middle attacks [47].
An important aspect of Tor is represented by its 'hidden services'. A note can advertise itself to some other nodes that will act as entry points for the server. The network will then connect to any of the entry points and negotiate a node that will be used as a meeting point in the node paths of the client and the hidden server. An attack against hidden services aimed to locate them has been developed by Lasse Øverlier and Paul Syverson in 2006. The assumption of the attack is that the adversary controls the first node that the server selects to build the anonymous path because the clear traffic that the node will receive will compromise the anonymity of the server. The relative countermeasure was proposed by the same authors few months later with the introduction of 'valet nodes'. Valet nodes act as an additional protection layer that guards the entry points preventing their direct communication with the hidden service. However, in 2016 the Tor project announced that the implementation requires a considerable amount of work, therefore the solution has not been adopted until more volunteer developers will be found [48].

2.3.4 I2P

I2P (Invisible Internet Project) is a project started in 2003 as a censorship-resistant P2P network similar to Tor. I2P was designed to provide the same functionalities as Tor (P2P content sharing, end-to-end encryption and routing) but protecting the users by keeping them inside the I2P network as opposed to Tor designed to be used as a proxy to anonymously access the Internet. Like Tor's hidden services, every user in the network has two isolated paths (chain of nodes) for incoming messages and outgoing messages to preserve the nodes' anonymity. However hidden services can be accessed from the regular Internet where I2P content is available only inside its own network.
In 2013 researchers have tested I2P security finding vulnerabilities that can eventually de-anonymize the user. Once in control of a portion of the network, an attacker could implement an Eclipse attack by blocking a resource to the node in that portion [49]. Hasib Vhora and Girish Khilari proposed a solution to prevent an Eclipse attack in 2015 with a structured overlay network [50].

3 Protocol Design

3.1 Introduction to Nessuno

Nessuno is a communication protocol through which a user can connect to other users in order to exchange messages in a way that nobody, except the sender and the receiver, can identify who sent the message, who received it and its content.
The aim of this protocol is to provide its user with a tool designed for censorship and surveillance resistance. Every user is represented by a 'node' and the group of all the interconnected nodes is defined as a 'relay'. The messages that are sent and received by the participants are in form of text and in any moment, new participants can join the relay by connecting to one of the nodes in the relay. A 'conversation' is defined as a set of one or more messages exchanged between two nodes that will be preserved and stored by each participant.

The core properties that together make up the definition of secure and anonymous chat protocol that Nessuno intends to be are:

- Confidentiality. The content of the message is shared only with the recipient it was addressed to.
- End-to-end encryption. The message's content is encrypted, hence not accessible until it reaches the receiver.
- Consistency. The content of the message has not been changed.
- Anonymity. The IP addresses of both the sender and the receiver is hidden.
- Metadata free. The source and destination of a message cannot be discovered.
- Decentralization. The protocol does not rely on a central system but is distributed among all the users.
- Non-Traceability. The messages are replicated in all nodes making it difficult to track the conversation flow (See 'Flooding').
- Origin authenticity. The identity of the sender can be proven by the receiver.
- Forward Secrecy. The confidentiality of past messages is preserved even if the encryption has been compromised.

Nessuno implements the concept of friend-to-friend network (F2F) as seen in Retroshare in order to connect directly only with trusted users. With Retroshare however, closing the connections to just real-world trusted users will critically limit the content available for the file sharing. Considering that Nessuno is a protocol primarily meant for communications, it does not receive the same impact as in Retroshare.
The most important feature at the core of Nessuno is being a metadata-free protocol. Nessuno does not send information about location or any other data related to the message, to the sender or to the receiver in its messages and the mechanism of flooding prevents the network to leak this information easily.

3.2 Relay

The relay consists of all the nodes that are able to send messages to each other.
A node can join a relay by connecting with one or more nodes that are already in the relay by sending a 'joining' packet. If the other node accepts the joining packet, the new node creates new private and public keys and sends the public key to the connected node to share with the relay. A node can leave the relay by sending a 'leave' packet to the relay. Alternatively, a node is considered out of the relay after some time of inactivity. (described in 'Keep alive' in the 'Implementation' Section).

3.3 Joining the relay

Joining to an existing relay is a process that needs to be carried out simultaneously by two users, one who is already part of the network and one who wants to join it. Prior to joining the relay, the two users must exchange their IP addresses. This procedure is vital as the two users need to reach from each other at the same time in order to implement the hole punching (see chapter 3.5). The assumption made is that the two users know in the real-world so they can find a third-party channel or communicate in person.

3.4 Flooding

Nessuno's routing mechanism is based on a flooding algorithm.
When a node receives a packet, it forwards it to all the nodes directly connected to it (except the one from which the message is coming).

This allows advantages like a certain level of ‘confusion’ in the relay that hides the identity of both the sender and the receiver and the flow of the messages.
Since the flooding naturally uses all the possible paths, it will also take the shortest one. Flooding the relay in this way also affects dramatically scalability, but since the beginning, Nessuno protocol assumes that anonymity and security have to be achieved as first priority, even when it needs a compromise with performance and scalability.
Not all the packets are forwarded but only the packets containing:

- The public key of the new nodes joining the relay
- A new public key from a node that renegotiate its key pair
- A message to a node
- Any other information that should be broadcasted to the entire relay

A consistent drawback of this approach can be the formation of loops in the routing system. Nodes that have already received a packet that needs to be forwarded, should not forward it again. Nessuno solves this problem by storing the information about each packet received in each node. The information stored are the ID of the packet and the time when the packet was received. When the packet is received, the node checks whether the ID is present in the table and in case it is found, it compares the current time with the time of the reception and if the validity period has expired, it forwards the packet. Otherwise the node drops it.

3.5 Choice of transport layer protocol

The decision on implementing Nessuno over UDP or TCP requires an analysis of the benefits and trade-offs that both implementation bring.
Although Nessuno is a protocol that implements privacy over performance, it has to be considered that if the performances are uncontrolled or the connection is unreliable, the communication becomes unreachable. Given the flooding logic involved, it is expected that any node can experience high latency due to the generated congestion in the network and some packets can be lost, therefore, it can be concluded that the protocol needs a congestion and retransmission mechanism to make the connection reliable.
On the other hand, peer-to-peer connections happening in a real-world scenario would be shielded by a Network Address Translation (NAT) that would drop the incoming

packets in absence of an established connection. Solutions involving the use of a server to initiate the connection cannot be taken into consideration as for the distributed nature of the protocol and the vulnerabilities that a server would introduce. The most suitable choice for this problem is the implementation the hole punching. Hole punching is a NAT traversal mechanism in which the peers that want to establish a connection, keep sending packets to each other until the NAT filter identifies it as a single established connection. This method is suitable for both TCP and UDP, however the latter is proven to be more efficient when using hole punching [51]. Since Nessuno needs both TCP's reliability and UDP's efficiency for traversing the NAT, the solution adopted is to use a TCP over UDP approach. This method requires a custom implementation of the TCP's retransmission and congestion techniques in a UDP packet, therefore, is also possible to implement only the mechanisms that are needed for the connection and reduce the overhead of the standard TCP connection.

3.6 Authentication

The nodes in the relay are identified by a public key that is shared with the relay as soon as the node is being accepted by one of the nodes in the relay.
The Nessuno protocol does not endeavour to link the public key of a node to the real-world identity behind it. The authentication process is over when the presence of the private and public keys is verified, hence, it is user's responsibility to apply a Trust-on-first-use (TOFU) approach like in SSH protocol to draw a line between the cryptographic identity and a real-world persona [52].
The key pair is changed every minute to provide forward secrecy, therefore, even if the private key has been compromised, the vulnerable messages will be limited in that timeframe.

3.7 Packet structure

This section contains the description of the packet structure that is available in more details in the Github repository of the project [1]. Details about the cryptographic primitives used are in the chapter 3.10.
In order to forge the packet that contains the message to be sent, Nessuno splits the payload in three blocks: header, HMAC and content.

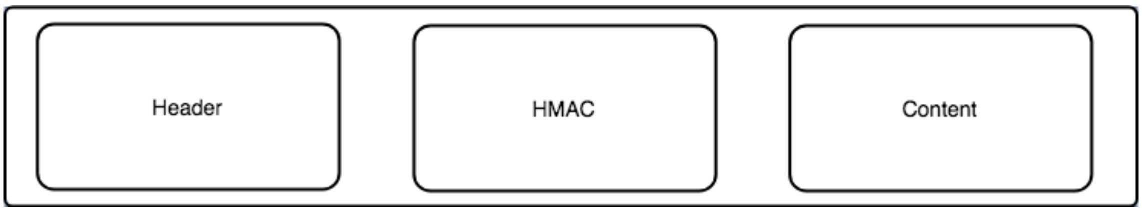

Figure 1. Message packet general structure

3.7.1 Content

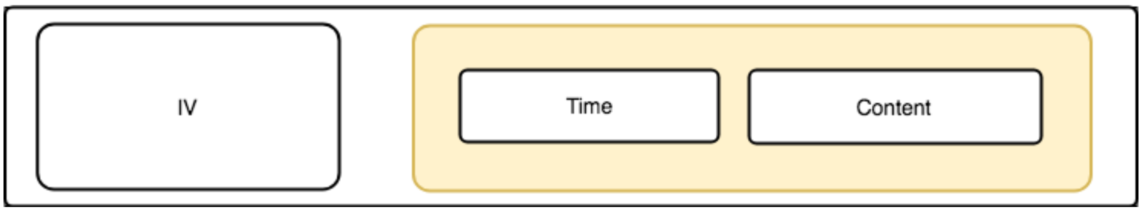

Figure 2. Content block

The content block generation starts from the actual message that the node wants to send, prepended with the UTC time [53] of the moment that it was created. This block is then encrypted using AES-128-CBC (yellow block) and prepended with the random Initialization Vector (IV) used for the encryption.

3.7.2 HMAC

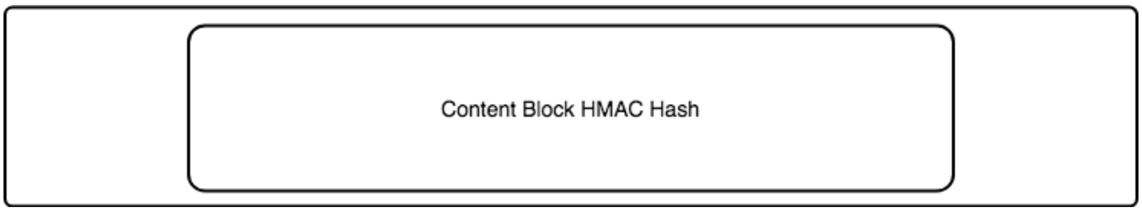

Figure 3. HMAC block

[1] Panarese, Stefano. “Crypto Module”. GitHub. Available https://github.com/Silent93/Nessuno/blob/master/Readme.md [Online] [Accessed 7 5 2018]

The HMAC contains value of the content block hashed with SHA-1-HMAC using a random key. This block is necessary in order to protect the content from tampering or malicious attempts to alter its value.

3.7.3 Header

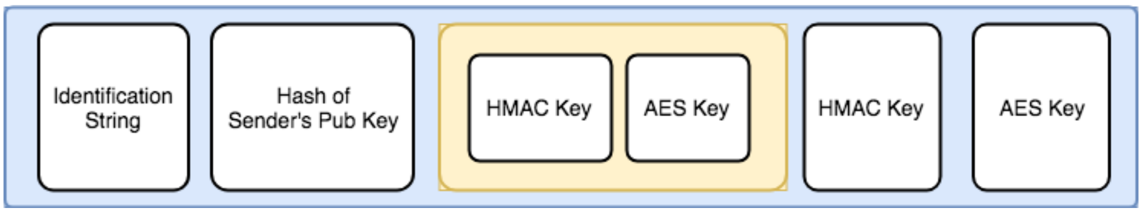

Figure 4. Header block

The header is used to store the keys used to encrypt the previous blocks and it is the block that the receiver will attempt to decrypt. The construction of the header starts with the concatenation of the key used in the HMAC block and the AES key used in the content block. This block is then prepended by its signature (in yellow) generated with the RSA private key of the sender to allow the verification of his identity from the recipient and the integrity of the keys. The resulting block is prepended with the SHA-1 hashed value of the sender’s public key in order to reveal the recipient only the virtual identity of the sender. Finally, the block is prepended with the 4-byte identification string ‘NESS’. This last identification string is useful to verify a successful decryption. The final block (in blue) is lastly encrypted with the RSA public key of the recipient in order to produce the header block.

3.8 Identify received messages

Since Nessuno does not make use of end-to-end addresses to identify the recipient, each node that receives a message should try to decrypt it in order to find out whether it was the designed receiver. If the attempt is successful the node will be able to read the first 4 bytes of the message which are ‘NESS’. This approach is clearly the simplest and least efficient as the computational power needed to the decryption attempt can cause a critical overhead. An acceptable solution would be to label each message with a code that can be recognized by the sender and the receiver of the message but which will seem randomly generated for the other nodes in the network. Although the solution can be effective, it introduces more complexity with the management of the labels for all the nodes and the security and anonymity matters to cover when negotiating the said label, it will not be implemented in this early version of the protocol.

3.9 History of events

It is responsibility of each node to store each event that is happened in the history of the chat. The most important events to be stored are:

- Messages received
- Messages sent
- New node joins the relay
- Node leaves the relay

Due to the heavy traffic generated by the events in the relay, it is not convenient at the moment to design a way to store the events when the node is offline.
This means that if a node is offline, it is not reachable by other nodes until it gets back online again and advertise itself in the network. By using the same key pair, the other nodes will recognize it and trust it again thanks to the TOFU approach.

3.10 Cryptography

All the messages are encrypted end-to-end using RSA public/private key pair.
The content is encrypted with the symmetric encryption algorithm AES-128-CBC with a random key, which will be included in the same packet's header encrypted with the public key of the recipient.
The symmetric key approach has been chosen because of the consistent performance gain in terms of decryption speed while keeping unaltered the level of security as the receiver still needs the private key to get the symmetric key used.
The choice of AES-128 preferred over AES-256 is due to a better stability of the algorithm and an unreliable design of the latter [53].
The hashing algorithm used is SHA-1 despite a collision has been found [54], the HMAC using SHA-1 is still to be considered a secure solution [55].
These countermeasures provide security when an attacker can be in situations like:

- Eavesdropping between nodes
- Altering, dropping or forging messages
- Alter consistency of the messages forwarded to different nodes

- Purposely delay the packets to increase latency and make the system unstable

4 Attacks against Nessuno

4.1 Sybil attack

Nodes in the network rely on the assumption that a single identity is bonded with a single computer. In Sybil attack, the attacker forges multiple identities to gain reputation and perform actions as they were carried by multiple computers. Nessuno does not implement a reputation system as the reputation comes from the trust of the peers, therefore the security level depends on the trust given by the node when choosing who to connect with. Although having a large number of friends (connected nodes) improve the overall security it can affect negatively if the chosen nodes are malicious as they can see the user's IP.

4.2 Eclipse attack

If Sybil attack depends on the trust of the user who invites one to join the network, the eclipse attack relies upon the trust of the invited user. If the user accepts to join a network by connecting with another user, he must acknowledge the fact that all his incoming and outgoing traffic will be controlled by that single node. When an attacker gains that trust, he can filter or forge messages to deliver to the victim. The substantial difference from the Sybil attack is the target. Where the Sybil attack threats the network, eclipse attack's target is represented by a single user.

4.3 Attack on user's anonymity

Since the adversary could be a global observer, he can spot the node that originated the message, especially with a small size group where nodes are often connected with a single friend. When an attacker monitors the network, he can conduct a timing analysis and correlate the messages coming and sent from a specific node to recognize the difference between a forwarded message and a new message. In order to be more effective, the attack can be carried out in two phases. During the first phase the adversary introduces himself into the network by gaining the trust of one of the node already in the network. In the second phase, the malicious node can forge a message and monitor it while it floods the network getting information about the topology and the distance to the other nodes by examining the time between sending the message and receive its acknowledgement from the recipient. The impact on the user's anonymity is considerable only for the sender as the adversary would notice a new message created from a node after analysing the in and out traffic. Therefore, as volume of traffic generated in the network grows, more difficult the analysis becomes. The recipient's anonymity is not affected as every node would receive and forward the same message but if the recipient replies to the message under a low traffic volume situation, the attacker might correlate the generation of a new message to the message received from another node. A possible solution to mitigate this threat is to use Tor as a proxy to escape the surveillance from the ISP and limit the network visible from the attacker. As a further improvement for the protocol, Nessuno could adopt a mechanism in which every node has to send a dummy message after a certain random time without sending one. This would raise the traffic volume present in the network and an observer even if global cannot draw a line between the sender and the receiver, although this approach is not part of this first early version of the protocol.

5 Implementation

The main design and features for implementing Nessuno are followed in the proof-of-concept representing a concrete example of Nessuno. This implementation's purpose is to show how Nessuno works and analyse how the network congestion can change with different topologies, node number and traffic volumes. The next chapters will break down the structure of an ideal client that implements Nessuno's principles in its core.
The proof-of-concept will not contain some of the following features as they solve issues like the NAT traversing that are not beneficial to the performance analysis.

5.1 TCP over UDP

The NAT traversal matter discussed in the protocol design, forces the use of the UDP protocol to transport the packet through a NAT. The implementation of a reliable UDP can be completely custom or based on a third-party project. A custom solution would be a better fit for Nessuno as the TCP mechanisms can be set for the specific situation. For example, it would be better for the network if the congestion system does not halve the size of the window every time a packet would fail to reach its destination. This is due to the fact that the environment in which Nessuno runs, could often lead to failures and errors during the transmission but they are caused by the latency between the nodes, hence the window size should be calculated accordingly with a wider error margin.
A suggested third-party solution is UDT [56], a reliable implementation of the UDP protocol written in C++ (but with API and wrappers for other languages available). Among the key features of this project there is the high customisability of the congestion control module that would help to fit Nessuno's needs.

5.2 Message queue

The message queue is a module made of a first-in-first-out (FIFO) queue structure containing the messages to be sent and the methods that alter the queue in order to add or remove messages in the queue. The message object found in the queue is supposed to be ready to send, hence already encrypted. The importance of this structure becomes vital when dealing with the socket module to prevent simultaneous access from different points of the software as queue actions such as "append" and "dequeue" must be thread-safe methods. Furthermore, the message queue entity must implement the singleton design pattern as it is supposed to be used by different threads at the same time. The most important methods offered by this module are listed below.

`SendToOne(message, peer)`

This method appends a single message in the queue to be delivered to a single peer. In most cases the message is operational and sometimes required when a peer is requesting an information about the network. For example, when a node just joined the network it can request the list of the public keys for the other nodes from the trusted friend. This method is implicitly used also in the *SendToAll* method. Returns a reference to the message in the queue so it can be manipulated before being sent.

`SendToAll(message)`

This method appends a message in the queue for every peer directly connected with the user. It makes use of *SendToOne* multiple times and it returns an array with the references to the messages enqueued.

`Cancel(message)`

This method removes a certain message from the queue. This method is rarely used as it does not respect the normal flow of the program. However, it represents an emergency action to take in case a message shall not be sent.

5.3 Forwarding

The forwarding mechanism seen in chapter 3.4 is implemented with the help of the before-mentioned message queue and its *SendToAll* method. The aim of module is to provide a policy for deciding whether to forward a message or not. The rule is that when a message is received, its ID is stored in a table together with the timestamp when it was received and is valid for a certain amount of time. Ideally this amount of time must be calculated out of the average latency of the network but in the proof-of-concept implementation, messages are not forwarded again for one minute.

5.4 UI

The User Interface (UI) is text based for the proof-of-concept for scope reasons but the UI is actually an important part of the final implementation. Although a technology can be disruptive and advanced, it cannot be adopted by many users if the usability is poor. PGP was a clear example of an outstanding approach with many issues in the user experience due to its complexity in the daily use. In order to be efficiently usable by the user, the UI must be intuitive and easy to setup.

5.5 Factory

The factory is the core module that is responsible for forging and parsing the message. When the clear-text message is ready, it needs to be encrypted and injected in a packet before passing it to the message queue module for being sent over the connection. This module also implements the different kind of messages that can be generated and takes care of the organization of the information inside the packet so that it respects the Nessuno's design. When parsing an incoming packet, the factory is the main module where the client attempts the decryption of the header in order to recognise if the user is the desired recipient of the message. The most important methods in this module are:

`Parse(packet)`
This is the most important method. It gets a packet that has been received and it tries the decryption with the user's private key. In case of success it returns the message content together with its information and the type of message received.

`ForgeMessage(text)`
This method generates an encrypted packet with the given clear-text ready to send to the message queue for the delivery.

`ForgeKeysInfo()`
This method generates the information about the public keys and pseudo identities in the network to send to the added friend in order to inform him about the other users that he can contact via their public key. This process is will not be mandatory when Nessuno implements a query system that will allow a new user to ask the network for the public keys along with the pseudonyms of other users participating.

5.6 Multithreading

In order to keep the client's performance high, some tasks must run on a separated thread. The main thread is dedicated to user's inputs while the majority of the workload is left in the background. The first thread is immediately run when the application is started and it waits for incoming messages listening on a specified port or 787 by default. When a user wants to connect with another friend, a different threat is needed in order to keep sending and forwarding messages while the hole punching technique is in process. Multithreading is vital in this kind of applications as the protocol needs multiple tasks to be carried out without interrupting the logical flow.

5.7 Storing the key pair

When the user generates his private and public key during the first run, the problem of how to store them safely arises. A custom implementation in this case is definitely not recommended as the key pair is the most sensitive information stored locally and if an attacker can access their location, he will be able to intercept and even impersonate the user in the network. A valid third-party solution is a wrapper for the operative system’s keyring which supposedly is the safest part of the system where the client is run. In the specific context of the proof-of-concept made for Nessuno, the python module loaded is ‘keyring’ [57] which offers some API to read, add and remove the secrets in the keyring found in different operative systems.

5.8 Hole punching

The hole punching technique explained before in chapter 3 is implemented when two users are trying to make a connection for the first time, becoming friends and directly connected nodes. Assumed that the users do not need a rendezvous server to know each other address and port as they are supposed to have this information. When this process is started the application starts a separated threat that sends UDP ACK packets to the other end waiting for a response. The client will try to send 99 datagrams with a 500ms delay and in case it is not successful it throws an exception. Sometimes a restricted corporate NAT might not be compatible with this technique. At the moment, the alternative solution is to enable port forwarding and explicitly dedicate a port to Nessuno. Once the ‘hole’ is opened, the keep alive module makes sure that the connection does not interrupt.

5.9 Keep alive

This module is basically consisted of a task that sends an ACK packet to the nodes which hole punching was successful in order to keep the connection active. The timeout is not fixed and can vary depending on the operative systems and the NAT type. Therefore, an ideal solution would calculate this timeout at the first connection and then use the value found as a timeout for the keep alive packet. For the purpose of testing the timeout is set to 10 seconds which is enough to make sure the connection is not lost.

5.10 Console commands

The interaction with the user happens via console commands that the user can type as a message. The application parses the message as a command when the first character is a '\' (backslash). The commands that are expected from the user are:

```
\connect [ip] [port]
```
This command starts the UDP hole punching with the specified IP and port

```
\changeID
```
This command generates a new key pair to use when encrypting/decrypting a message

```
\list
```
This command lists the nodes participating in the network

5.11 Source code

The project is open source and available on GitHub. [1] The goal is to involve more experts from different fields in the improvement of Nessuno as inputs from cryptanalysts and other cybersecurity experts are vital to bring this project to a level where it can be used in real world scenarios where surveillance limits people's freedom of speech.

[1] Panarese, Stefano. "Nessuno". GitHub. Available: https://github.com/Silent93/Nessuno [Online] [Accessed 7 5 2018]

6 Performance

A performance test aims to show the volume of traffic generated by a variable number of nodes using Nessuno to communicate. Since a real-world test was not possible at the time of writing, collecting the data needed is a demanding task. In this paper, the method used to solve this challenge is to combine the size of the message composed by Nessuno and the number of nodes to predict the volume of traffic that a single node will receive.

6.1 Packet size

The size of the packet is calculated from the output of the cryptographic functions applied in order to build the final encrypted packet containing the message.
According to the packet structure described in the 3.7 of this paper, the total size is an aggregation of the three units composing the full packet.
Header. Considering that the header is eventually encrypted with the receiver's public key, the size estimation of the encrypted output depends on the size of the input. This is due to the fact that although the output is predictable, the maximum size allowed to be encrypted is 214 bytes as the algorithm used is RSAES-OAEP with 2048-bit key using SHA-1 digest. The generated block to be encrypted is calculated as a sum of:

- 32-bit from the 'NESS' flag prepended for identifying a successful decryption
- 160-bit from the SHA-1 hashed public key of the sender (fingerprint)
- 2048-bit from the signature of the HMAC and AES keys
- 160-bit from the HMAC key
- 128-bit from the AES key

After aggregating the values from the list above, the header of the packet considered is 316 bytes. Since the maximum size accepted by the RSA algorithm used is 214 bytes, the solution is to split the header before applying OAEP padding and encrypt each block separately. Finally, the encrypted blocks can be joined together to produce the header block with a size of ***428 bytes***.

HMAC. The HMAC block is composed by the SHA-1 hashed value of the content block, hence its size is fixed to ***20 bytes***.
Content. The content block generation starts with the plain-text message prepended with a UTC time in format *YYMMDDhhmmssZ* compliant with ASN.1 UTCTime type [58]. This value is encrypted via AES-128 according to the PKCS#7 standard [59], therefore the output size depends on the length of the message. The encrypted block is then prepended with the IV used for the encryption, adding more 128 bits (16 bytes) to the sum. Assumed a message length of 255 characters (255 bytes), the size of the AES generated block will be 272 bytes. Considered the prepended IV the entire content block size is ***288 bytes.***

The performance test will be executed considering the size of the packet sent by Nessuno to be the sum of the sizes listed above, hence **736 bytes**.

6.2 Bandwidth estimation

A performance estimation test needs to be combined with the factors that are part of the environment in which Nessuno is run. One of the general factors analysed is the global average connection speed, retrieved from the traffic to the Akamai CDN platform that counts 149 countries across the world and trillions of requests. From the latest report available at the time of writing, the global average connection speed is 7.2 Mbps [60]. A data visualization provided by M-Lab, gives more information about a single country like Iran which average speed of 1.5Mbps [61] is considerably lower than the global average. In conclusion, since a global value is not meaningful for the purpose of this performance estimation, the connection speed is arbitrary chosen to be 1.5 Mbps to reflect a real-world scenario like Iran.

6.3 Results

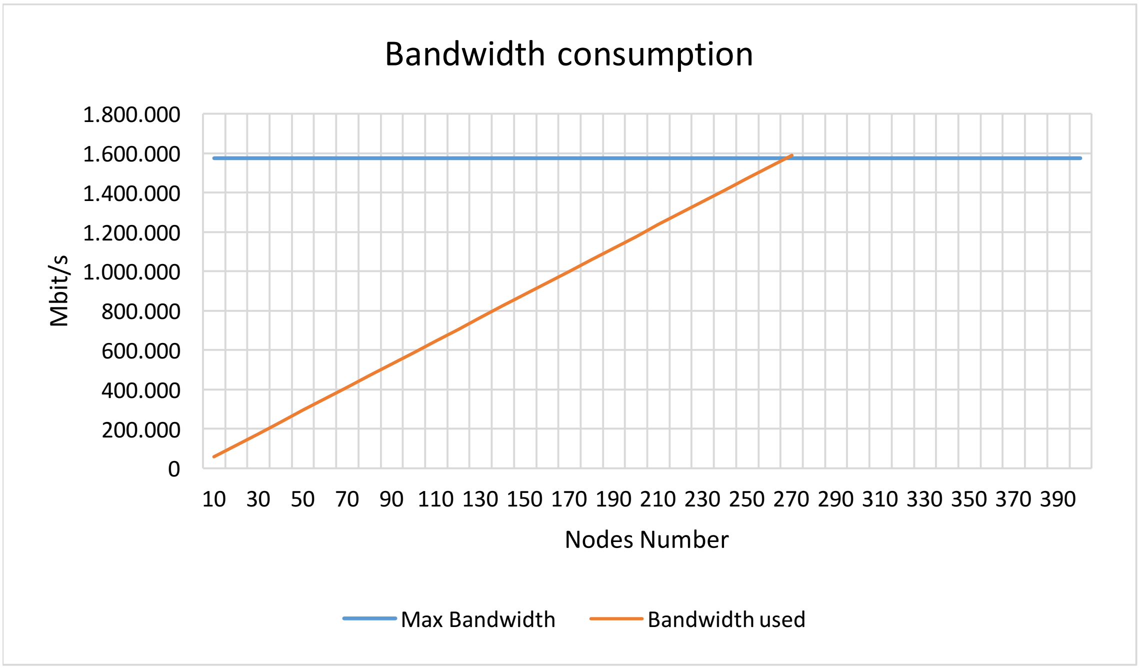

Figure 5. Chart representing the estimated bandwidth consumption

The results are obtained combining the bandwidth available with the traffic that can be generated by a certain number of nodes in Nessuno's network and received by a single node. The estimation is purely theoretical and in a real-world scenario it would be affected by various elements such as NAT, real latency, ISP and implementation of the protocol. However, the representation in Figure 1 gives an understanding that the theoretical size that a node running Nessuno can handle in this early version is around the number of 270. Consistent improvements can be implemented in the routing system including workarounds like a probabilistic algorithm that drops packets with a high chance to be addressed to other nodes (explained in "Future Work").

7 Conclusion

This paper focused on the privacy and anonymity issues emerging from the contexts of those countries with a heavy employment of surveillance. In a preliminary phase, it was important to define what privacy and anonymity are and what factors in the communication affect it. In presence of a global observer whose motivation is to intercept and identify the authors and recipients of messages in secret communication, users need tools to preserve their identity and make sure to be anonymous. As shown in chapter 2, the existing solutions to protect the privacy of the user are numerous but considering decentralised solutions only, they are not optimal for being adopted by a restricted group of users. Solutions such as Bitmessage, Retroshare and Mixnets contain inspiring features that led to the development of Nessuno, the protocol presented in this work and the contribution to the research in the anonymous way of communications. Nessuno is a friend-to-friend network where a considerable level of security and anonymity depends on the trust in the *friends*, the nodes chosen to be directly connected with the user. In order to provide confidentiality and integrity, messages sent are encrypted using the public key of the receiver who is the only node able to decrypt the message. The flooding mechanism applied prevents time-correlation attacks that a global observer can execute as every node attempt to decrypt the message before forwarding it via its *friends* to the rest of the network. The implementation of Nessuno and particularly of the flooding mechanism generates a trade-off in terms of performance showed in the "Performance" section that contains a theoretical estimation of the bandwidth consumption. Guidelines about the client implementation of the protocol are given in the "Implementation" section and a basic proof-of-concept has been developed and shared with the community of GitHub

8 Future work

Many improvements can still be researched for Nessuno from various point of interests. Due to the complexity of the topic, more work has to be done with the cryptographic choices in the packet considering that despite the current solution does not contain critical cryptographic vulnerabilities, a better solution would favour less computational power needed without affecting the security. As shown by existent solutions [62] the performance trade-off that results from the flooding mechanism can be reduced with the implementation of a prediction algorithm that based on factors like the chance of being addressed by a message or the capability of relaying that message to the *friends* can decide to drop an incoming packet to reduce the overhead. This work does not include a real-world test due to its time-consuming nature although this test is needed in order to collect more realistic data about the performance and security impact of Nessuno. This early version of Nessuno represents a solution that aims to involve more experts from the open-source community that can help by developing, giving feedback and raise awareness among the subjects that want to evade the surveillance of their country.

Bibliography

[1] P. Winter and J. R. Crandall, “The Great Firewall of China: How it Blocks Tor and Why it is Hard to Pinpoint,” *Login: The Usenix Magazine,* vol. 37, no. 6, pp. 42-50, 2012.

[2] R. Deibert, J. G. Palfrey, R. Rohozinski and J. Zittrain, Access Controlled: The Shaping of Power, Rights, and Rule in Cyberspace, London: The MIT Press, 2010.

[3] Reporters without borders, “Enemies of the internet 2013 Report Special Edition: surveillance,” 2013. [Online]. Available: https://www.reporter-ohne-grenzen.de/fileadmin/docs/enemies_of_the_internet_2013_01.pdf. [Accessed 07 05 2018].

[4] O. Tkacheva, L. H. Schwartz, M. C. Libicki, J. E. Taylor, J. Martini and C. Baxter, “Internet Freedom and Political Space,” RAND Corporation, Santa Monica, CA, 2013.

[5] G. Serge, L. Ronald and d. H. Paul, “Reforming European Data Protection Law,” Springer, Tilburg, 2015.

[6] W. Jisuk, “The right not to be identified: privacy and anonymity in the interactive media environment,” *New Media & Society - NEW MEDIA SOC,* vol. 8, no. 6, pp. 949-967, 2006.

[7] S. Jeff and T. Jordon, Cyber Disobedience: Re://Presenting Online Anarchy, John Hunt Publishing, 2014.

[8] Network Security, “In brief,” *Network Security,* vol. 2013, no. 10, p. 3, 2013.

[9] Committee to Protect Journalists, “Court upholds 10-year sentence for journalist Shi Tao,” 1 7 2005. [Online]. Available: https://cpj.org/2005/07/court-upholds-10year-sentence-for-journalist-shi-t.php. [Accessed 7 5 2018].

[10] J. King, “Chinese journalist Shi Tao released after 8 years in prison,” 8 9 2013. [Online]. Available: https://edition.cnn.com/2013/09/08/world/asia/shi-tao-journalist-free/index.html. [Accessed 7 5 2018].

[11] R. Mackinnon, “Shi Tao, Yahoo!, and the lessons for corporate social responsibility,” 30 12 2007. [Online]. Available: http://www.rconversation.blogs.com/YahooShiTaoLessons.pdf. [Accessed 7 5 2018].

[12] A. White, “Protecting the People Behind the Stories That Keep Journalism Alive,” 10 6 2015. [Online]. Available: https://ethicaljournalismnetwork.org/ethics-at-source-protecting-the-people-behind-the-stories-that-keep-journalism-alive. [Accessed 7 5 2018].

[13] C. Stöcker, “Snowden Reveals How GCHQ in Britain Soaks Up Mass Internet Data,” 7 7 2013. [Online]. Available: http://www.spiegel.de/international/world/snowden-reveals-how-gchq-in-britain-soaks-up-mass-internet-data-a-909852.html. [Accessed 7 5 2018].

[14] Spiegel, “Edward Snowden Accuses Germany of Aiding NSA in Spying Efforts,” 7 7 2013. [Online]. Available: http://www.spiegel.de/international/world/edward-snowden-

accuses-germany-of-aiding-nsa-in-spying-efforts-a-909847.html. [Accessed 7 5 2018].

[15] M. Marquis-Boire, B. Marczak, C. Guarnieri and J. Scott-Railton, “You only click twice: FinFisher’s Global Proliferation,” 13 3 2013. [Online]. Available: https://citizenlab.ca/2013/03/you-only-click-twice-finfishers-global-proliferation-2/. [Accessed 7 5 2018].

[16] S. Winkler and S. Zeadally, “An analysis of tools for online anonymity,” *International Journal of Pervasive Computing and Communications,* vol. 11, no. 4, pp. 436-453, 2015.

[17] Common Criteria, “Common Criteria for Information Technology Security Evaluation,” 4 2017. [Online]. Available: https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf. [Accessed 7 5 2018].

[18] A. F. Westin, Privacy and freedom, London: Bodley head, 1970.

[19] C. J. Bourn and J. Benyon, “Data protection : perspectives on information privacy,” Leicester, 1983.

[20] A. Pfitzmann and M. Hansen, “A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management,” 18 12 2009. [Online]. Available: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf. [Accessed 7 5 2018].

[21] D. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonym,” *Communications of the ACM,* vol. 24, no. 2, p. 84–88, 02 1981.

[22] P. J. Honigsberg, “The Evolution and Revolution of Napster,” *University of San Francisco Law Review,* vol. 36, p. 473, 2002.

[23] “U.S. Copyright Office: Brief in A&M Records v. Napster,” 2000. [Online]. Available: https://www.copyright.gov/docs/napsteramicus.html#summary. [Accessed 7 5 2018].

[24] United States District Court, “Order against Napster Inc.,” 5 03 2001. [Online]. Available: http://news.findlaw.com/cnn/docs/napster/napster030601ord.pdf. [Accessed 7 5 2018].

[25] M. Portman, P. Sookavatana, S. Ardon and A. Seneviratne, “The Cost of Peer Discovery and Searching in the Gnutella Peer-to-peer File Sharing Protocol,” in *IEEE International Conference on Networks*, 2001.

[26] X. Ren-Yi, “Survey on anonymity in unstructured peer-to-peer systems,” *Journal of Computer Science and Technology,* vol. 23, no. 4, pp. 660-671, 2008.

[27] I. Clarke, S. O., W. B. and H. T., “Freenet: A distributed anonymous information storage and retrieval system. In , volume 2009, pages 46–66. Springer Berlin Heidelberg, 2001.,” *Lecture Notes in Computer Science ,* vol. 2009, 2001.

[28] Suma-Ev, “Ausbruch aus der Monokultur, 12-ter SUMA-EV Kongress,” 2015. [Online]. Available: https://suma-ev.de/presse/Ausbruch-aus-der-Monokultur.html. [Accessed 7 5 2018].

[29] T. G., D. Z., B. T. and D. Y., “A Traceback Attack on Freenet,” *IEEE Transactions on Dependable and Secure Computing,* vol. 14, no. 3, pp. 294-307, 2017.

[30] “Elttam,” [Online]. Available: https://www.elttam.com.au/. [Accessed 7 5 2018].

[31] D. Hodson, “RetroShare Advisory,” 10 01 2017. [Online]. Available: https://github.com/elttam/advisories/blob/master/Retroshare/RetroShare%20Advisory%20-%20elttam.pdf. [Accessed 7 5 2018].

[32] G. Glenn, No Place to Hide: Edward Snowden, the NSA, and the U.S. Surveillance State, Metropolitan Books, 2015, p. 148.

[33] S. Alfonsi, “60 Minutes: Hacking your phone,” 17 04 2016. [Online]. Available: https://www.cbsnews.com/news/60-minutes-hacking-your-phone/. [Accessed 7 5 2018].

[34] A. Biryuko, I. Pustogarov and R.-P. Weinmann, “Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization,” in *2013 IEEE Symposium on Security and Privacy*, Berkley, 2013.

[35] J. Warren, “Bitmessage: A Peer-to-Peer Message Authentication and Delivery System,” 27 11 2012. [Online]. Available: https://bitmessage.org/bitmessage.pdf. [Accessed 7 5 2018].

[36] M. Jakobsson and A. Juels, “Proofs of Work and Bread Pudding Protocols,” in *Communications and Multimedia Security*, 1999.

[37] C. Dwork and M. Naor, “Pricing via Processing, Or, Combatting Junk Mail,” in *Advances in Cryptology — CRYPTO’ 92: 12th Annual International Cryptology Conference Santa Barbara, California, USA August 16–20, 1992 Proceedings*, vol. 740, Santa Barbara, Springer, 1992, p. 139–147.

[38] “Bitmessage Wiki,” [Online]. Available: https://bitmessage.org/wiki/Main_Page. [Accessed 7 5 2018].

[39] P. Zimmermann, “Why I wrote PGP,” 1999. [Online]. Available: https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html. [Accessed 7 5 2018].

[40] B. Schneier, Applied Cryptography, New York: Wiley, 1995, p. 587.

[41] R. Thomas, Attacks on PGP: A User’s Perspective, Sans Institute, 2003.

[42] N. Weaver, *USENIX Enigma 2016 - The Golden Age of Bulk Surveillance,* 2016.

[43] D. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonym,” *Communications of the ACM,* p. 84–88, 1981.

[44] D. Goldschlag, M. Reed and P. Syverson, “Onion Routing for anonymous and private internet connections,” *Communications of the ACM,* vol. 42, no. 2, p. 39–41, 02 1999.

[45] R. Dingledine, N. Mathewson and P. Syverson, “Tor: The Second-Generation Onion Router,” in *Proceedings of the 13th USENIX Security Symposium*, 2004.

[46] N. Borisov, G. Danezis, P. Mittal and P. Tabriz, “Low-Resource Routing Attacks Against Tor,” in *Workshop on Privacy in the Electronic Society*, 2007.

[47] A. Johnson, R. Jansen, A. D. Jaggard, J. Feigenbaum and P. Syverson, “Avoiding The Man on the Wire: Improving Tor's Security with Trust-Aware Path Selection,” in *Network and Distributed Security Symposium*, 2017.

[48] Tor Project, “Hidden Services need some love,” 22 04 2013. [Online]. Available: https://blog.torproject.org/hidden-services-need-some-love. [Accessed 7 5 2018].

[49] C. Egger, J. Schlumberger, C. Kruegel and G. Vigna, “Practical Attacks against the I2P

Network," in *16th International Symposium on Research in Attacks, Intrusions and Defenses*, St. Lucia, 2013.

[50] H. Vhora and G. Khilari, "Defending Eclipse Attack in I2P using Structured Overlay Network," *International Journal of Science, Engineering and Technology Research,* vol. 4, no. 5, pp. 1515--1518, 5 2015.

[51] A. Milanović, S. Srbljić and V. Sruk, "Performance of UDP and TCP Communication on Personal Computers," in *Electrotechnical Conference MELECON*, 2000.

[52] R. P. P., L. R. Nair and T. Ijyas, "Incorporating Trust in Public Key Infrastructure Certificates," *Advances in Computational Sciences and Technology,* vol. 10, no. 5, pp. 671-686, 2017.

[53] B. Schneier, "Another New AES Attack," 30 07 2009. [Online]. Available: https://www.schneier.com/blog/archives/2009/07/another_new_aes.html. [Accessed 7 5 2018].

[54] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, Y. Markov, A. P. Bianco and C. Baisse, "Announcing the first SHA1 collision," 23 02 2017. [Online]. Available: https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html. [Accessed 7 5 2018].

[55] H. Krawczyk, M. Bellare and R. Canetti, " HMAC: Keyed-Hashing for Message Authentication," 1997. [Online]. Available: https://www.ietf.org/rfc/rfc2104.txt. [Accessed 7 5 2018].

[56] Y. Gu and R. L. Grossman, "UDT: Breaking the Data Transfer Bottleneck," 2011. [Online]. Available: http://udt.sourceforge.net/. [Accessed 7 5 2018].

[57] K. Zhang, "Keyring Library," [Online]. Available: https://pypi.org/project/keyring. [Accessed 7 5 2018].

[58] "ASN.1 - UTC Time," [Online]. Available: https://www.obj-sys.com/asn1tutorial/node15.html. [Accessed 7 5 2018].

[59] R. Housley, "Cryptographic Message Syntax," 09 2009. [Online]. Available: https://tools.ietf.org/html/rfc5652#section-6.3. [Accessed 7 5 2018].

[60] Akamai, "State of the Internet report," 2017. [Online]. Available: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q1-2017-state-of-the-internet-connectivity-report.pdf. [Accessed 7 5 2018].

[61] Measurement Lab, "Islamic Republic of Iran," 2018. [Online]. Available: https://viz.measurementlab.net/location/asir?aggr=year&isps=AS43754_AS12880_AS16322&metric=download&regional=0. [Accessed 7 5 2018].

[62] R. Gaeta and M. Sereno, "Generalized Probabilistic Flooding in Unstructured Peer-to-Peer Networks. Parallel and Distributed Systems," *Unstructured Peer-to-Peer Networks. Parallel and Distributed Systems,* vol. 22, no. 12, pp. 2055-2062, 2012.

[63] [Online]. Available: https://tools.ietf.org/html/rfc5652#section-6.3.