Holistic Confidentiality in Open Networks

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker Kyriakos Alexis Pimenidis

aus Münster, Deutschland

Berichter: Universitätsprofessor Dr. rer. nat. Dr. h.c. Otto Spaniol
Universitätsprofessor Dr. rer. nat. Dogan Kesdogan

Tag der mündlichen Prüfung: 10. Februar 2009

This work was created with the help and support of many helpful and kind people.

Representative for all, I’d like to thank my wife for her endless patience.

Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ever since the beginning of mankind people have had confidential messages to send. This still holds true today, be it be brokers exchanging information about possible future mergers, lovers writing letters, or secrets conveyed by spies. Without the presence of prying fellow human beings all of these messages could just be sent as plain letters. However, in most situations, and especially in the presence of modern means of telecommunication, the need for covert communication is omnipresent.

In order to hide the content of a message, cryptographic means have long been known. Early examples are the use of a stick, also known as a scytale, together with a roll of paper by Spartian warriors, or the cipher used by Gaius Iulius Caesar[1]. Of course, both of these methods are outdated and numerous successors have been developed. The Data Encryption Standard ("DES") and its enhancements evolved in the 1970s[2]. Since then it has been possible to encrypt messages in a way that the majority of unwarranted people are incapable of deciphering them.

Still, it can be shown that protecting the content of a message is sometimes not sufficient. The mere fact that a person sends encrypted messages to certain recipients can raise enough suspicion to justify further investigations; possibly leading to the secret being compromised. For example, someone downloading files from a website hosting pornographic material cannot hide his actions by encrypting the traffic. Also, people peering with foreign secret services are likely to be traitors or spies[3].

Additional cases are given for people who would like to send information to somebody without revealing their own identity, e.g., whistle-blowers giving legal authorities tip-offs about on-going or future crimes. This also applies to investigations by law enforcement agencies, like public prosecutors or the police research-

[1]For both see, e.g., [Col04].

[2]See, e.g., [Nat77] and [Nat01].

[3]See, e.g., [Hüt70].

ing information for a criminal case. In these cases, encryption will not solve the problem of hiding the sender's identity.

Thus, we recognize that protecting a message's sender and recipient, or their relationship, can be as important as hiding its content. We can also see that in some cases, encryption is of no help to hide the actual confidential information.

1.1 Anonymous Communication

In computer science, the research area of *anonymous communication* deals with protecting the aforementioned information, i.e. hiding a person's identity and also the relation between two peers in a computer network. To be more precise, it covers explicitly[4]:

- hiding a sender/recipient relationship.
- hiding the identity of a message's sender and/or recipient
- hiding the volume and type of traffic between two peer partners.

Any of the above, if leaked, can give a third party enough information to infer further information or be used in a way to create unfortunate consequences for the sender or the recipient of a message. For example, if a series of encrypted e-mails sent from a hotel abroad to a big industrial company are discovered, it can lead to the conclusion that the person in the hotel is an employee, possibly a manager, from that company. From this point on, committing industrial espionage against the employee is only a matter for unscrupulousness reasons[5].

A number of theories and solutions have been developed in order to protect the privacy for those with the respective need. The seminal paper of David Chaum on "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms" [Cha81] is widely considered to be one of the most influential works and marks the actual start of research in this area. On the practical side, tools like AN.ON [BFK00], Mixmaster [MCPS03] or Tor [DMS04] have found wide deployment and usage across the Internet.

Anonymous communication, and with it this work, lies within a bigger context of different research areas (see also Figure 1.1):

- Hiding the fact that a user is communicating at all has stronger requirements than research into anonymous communication. This area is mostly dealt with in research into steganography, if the message can be embedded in a carrier. Some research in the area of censorship-resistant systems also considers building systems, which hide the existence of communication, e.g., [CSWH00].

[4]The following items are listed in no particular order.

[5]See, e.g., [Int01] or [Fin06].

- It is related to hiding the content of a message. This task is solved by means of cryptography. To a certain extent, this has been well researched and understood[6].

- Host and network security, i.e. research that is targeted towards securing computers, their hardware and software running on all layers. This is an essential part of any other topic in the area of IT security. Since the late 1990s this area has been a vast growing field, due to an uncounted number of computer worms, viruses, break-ins and privacy leakages.

The relation of anonymous communication to all of these areas will be of importance in various chapters of this work.

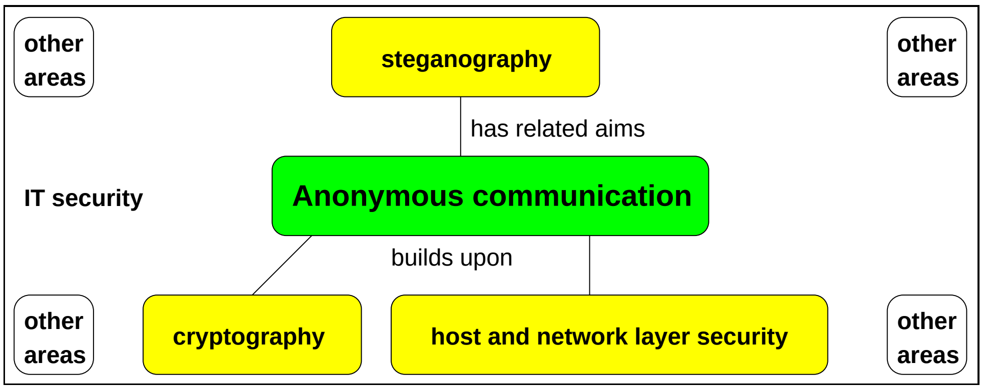

Figure 1.1: Related areas of research, all of them being sub-parts of IT security

1.2 Contribution

One of the most fundamental problems, which anonymous communication has to deal with, is an *open environment*. These are a counterpart to *closed environments*, where most properties can be enforced by either simple technical means or policies. This is possible, as in closed environments there is usually a single entity which controls all hosts and network lines, or possesses the legal power to react to misbehaviour. Also, the identity of all participants is known in closed environments. Conversely, in an open environment there is neither an entity with central control, nor a way to penalise abuse. Hence, *technical means* are the only way to achieve any property, especially security properties.

The aim of this work is to act as a foundational work for providing *data protection in open environments*. To this end, we give an overview of important parts

[6]Compared to the topic of anonymous communication.

and aspects of this research area. This work is targeted at researchers, technicians, engineers and students. It displays the research area in a bigger context and from a practical point of view. This allows us to see the big picture of current advancements and issues in this area. To conclude, we give the reader an outlook on future research questions.

The chapters of this work can be divided into two main parts: the first three chapters follow a constructive scheme. After an introduction into this field's basics, we describe technology which achieves the targeted protection. Then, we consider additional works, which aim at enhancing single aspects of anonymous communication.

In the second part, we will take the position of *devil's advocate* and show weaknesses and vulnerabilities of anonymous communication systems. For this, we discuss the notion of an attacker and classify attacks. In the conclusion we provide the reader with an exploration of today's most dangerous attacks.

1.2.1 Roadmap, Constructive Part

Initially, we introduce the general *terminology* as well as basic *issues* in this field. The goal of this chapter is not only to ease the reading and understanding of this work and other publications of this research area, we also want to raise *awareness* about underlying principles and prepare the ground for more technical details. To this end, we discuss in detail the *targeted aims* of anonymous communication research, as well as notions, definitions and implications of terms like, e.g., *security* or *privacy*.

As we work towards a holistic view, we include a discussion on connections to related research areas. This refers to computer networks, cryptography and IT security. The basics of each of these three areas are discussed according to their relevance to our topic.

In addition, we elaborate on the entities involved into deployments of anonymous communication infrastructure. As we will see, anonymous communication is about *multilateral security*, with different stakeholders having different, even contradicting, requirements. For a full understanding of the severity of the issues in this field, it is inevitable to display the conflicts emerging from the deployment of systems.

On this basis, we introduce *systems* and *technologies* which are designed to protect the privacy of their users. Rather than discussing the potential benefits of theoretical systems, or proposed systems which never left the stage of a theoretical proposal, we will mostly focus on systems which were brought to an actual deployment.

Our main motivation for this is the fact that theoretical proposals abstract from reality and hence do not suffer from a multitude of problems. By working on a conceptual level, many issues which affect the security and scalability of deployed systems can be ignored – examples include fingerprintingfingerprint of traffic patterns [LL06], learning a host's identity from network layer latencies [HVCT07] or even clock skew due to the temperature of loaded CPUs [Mur06].

To conclude the first part, we summarize works which add additional value to the basic building blocks of anonymizing networks. These are, e.g., works on providing *quality of service*: most networks offering privacy-enabled communication are geographically spread and heavily loaded with traffic. This leads to a severe impact on the quality of service. Studies have shown [Köp06] that the number of users in an anonymizing network is linked to its performance. Also, it is generally considered that the degree of protection of these networks is linked to their number of users. Hence, only an adequate speed of transmission is able to attract enough users to a network for a reasonable level of protection for each single user.

Here, one of the core questions is how to achieve this goal without risking a loss of security at the same time, or at least limiting the impact on security while gaining better bandwidth and less latency. More precisely, if implementations of systems try to provide a better quality of service by selecting paths and nodes in a way to achieve good performance, they will more likely fall prey to a specific class of attackers. Similarly, adding arbitrary security measures for an increased level of protection results in a strong decrease of usability. This holds true for, e.g., dummy traffic, induced latency ("mixing"), local route selection by helper nodes, deactivation of active content on webpages and much more. Thus, it is not only a temporarily technical infeasibility which dictates that services for anonymizing systems offer a bad quality of service, but the overall security may depend on it.

Other works discussed add additional features to the networks, like enforced anonymity or extending the scope of data transmitted by the means of these networks.

1.2.2 Roadmap: Devil's Advocate

As virtually every other technique in IT security, anonymizing networks do not achieve perfect protection[7]. There are weak algorithms, implementation errors, mistakes in deployment, and a lot of other ways and opportunities for mischief to happen. However, the biggest danger is given if it is possible for malicious people (*attackers*) to deliberately tamper with the system and learn confidential information.

For the purpose of understanding the threats due to attackers, we set ourselves the goal of shedding light into this area. Thus, we elaborate on who is actually capable of committing attacks. This includes a description of technical capabilities necessary for committing attacks. Further considerations include the motivation and opportunity for doing so.

[7] In this area, the only known exception is the anonymization method "DC-network".

We have elaborated above that adding arbitrary layers of security is not an option in this research area. Hence, it is crucial to choose wisely which methods of protection have to be applied. However, this is usually derived from the set of entities which a user does not want to know his data. If the utilized network, on the other hand, protects against a different set of attackers, this only has a potential negative impact on the user's security, but may also unnecessarily affect his or her quality of service. On the other hand, a network which tries to defend against as many attackers as possible will result in a very low quality of service, thus decreasing the amount of users, which in turn is likely to reduce the degree of protection (a schematic graph of this relationship is shown in Figure 1.2).

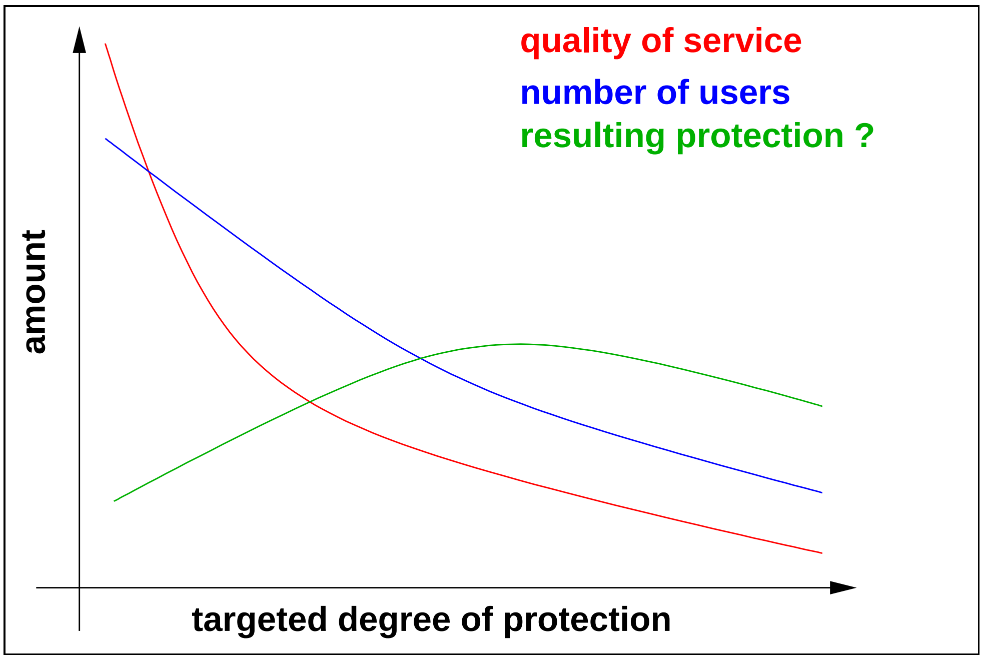

Figure 1.2: Schematic relationship between targeted protection, quality of service, and resulting protection level

For an end user it is very difficult to actually know which attackers are dangerous and interested in his data at the same time. Often people choose the wrong attacker to defend against and are neither aware nor capable of judging the success of their choice because the targets and intentions of attackers are not widely known or publicly discussed[8]. In addition, previous security evaluations for anonymizing networks have been conducted using abstract attacker models with next to no relevance for real-world systems.

[8]This can be compared to security awareness in other areas, e.g., the common perception that utilizing an airplane is dangerous, while smoking is considered harmless.

Following this research, we give a classification of known attacks on anonymizing systems. However, we extend the classical view to a holistic one and include threats originating from related areas. We also show the influence of dangers which were overseen in the traditional research due to a strict abstraction level. To this end, we first discuss the flow of a message through an anonymizing system. The flow is used as a major thread for the discourse through existing vulnerabilities. The chapter is concluded with non-technical attacks and some theoretical results.

Finally, in the conclusion we explore the previous study and identify the most serious attacks on contemporary anonymity systems. We bring the work to an end by pointing out possible ways for their mitigation and identifying research questions for yet unsolved problems.

Chapter 2

Terminology and Scenario

This chapter gives an overview on the terminology as it is used in this document. Its purpose is to support the understanding of this text for readers who are yet not familiar with terms used in IT security, computer networks or anonymous communication. At the same time, we introduce the basic issues on the respective areas. This way we want to raise awareness on important circumstances which influence design patterns and decisions in the area of anonymous communication systems.

As anonymity networks are a rather young field there is no fixed terminology and published works often use homonyms, synonyms or even barbarisms[1] which make understanding difficult and content unclear. Even though some works exist on unification and standardisation of the vocabulary used by researchers in this area [PH06], usage and even capitalization of terms is not constant.

In the course of this chapter we first cover the basics of computer networks (section 2.1) and IT security (section 2.1) – both being indispensable building blocks for anonymity systems. On these grounds we continue with topics on privacy and anonymity (section 2.3). The chapter concludes with some notions on the scenarios and setup (section 2.4). The latter part also introduces and analyses parties and entities related to and affected by privacy-enhancing techniques.

2.1 Computer Networks

Communication is, of course, a central part of anonymous communication research. In this case, it refers to communication done by computers utilizing *computer networks*. Networking technology itself can be considered one of the very basic research fields in computer science. For this reason we do not delve very deeply into this matter. Instead we point out important parts which we will be referring to intensively in the course of this work. For a more rigorous approach to networking basics the reader is kindly asked to consult books like [Com02, Tan03].

[1]In linguistics the term “barbarism” refers to a non-standard, possibly even wrong, use of a word.

While computer networks are ubiquitous today and using them seems trivial, there is a number of non-trivial tasks to be fulfilled in order to exchange data between two remote peers[2]. Most of these tasks meet high demands with regards to efficiency, dependability and security. For a number of contemporary network protocols it was the case that not all of these criteria could be sufficiently taken into account, hence leaving room to trade-offs.

In order to cope with the multitude of problems and their complexity, communication protocols are designed in *layers*. Each layer is designed to cope with only a limited set of problems and offers its services to "upper" layers. The higher layers can then build upon the guarantees made by the "lower" layers. By combining layers into a *protocol stack*, applications can choose from a rich set of network properties to fulfill their goals of communication.

We will briefly describe the seven layers of the *ISO/OSI model* for computer networks in Table 2.1. They depict in a very concrete way how problems of transmitting data to a remote peer can be split into several groups and solved separately. Even though the ISO-model is only loosely related to the protocols used in the Internet, it demonstrates better how to divide a problem into manageable pieces.

#	Layer	Function
7	Application	services to user-defined application processes
6	Presentation	establishes context between single application layer entities
5	Session	manages connections between applications
4	Transport	reliability, flow control and error control
3	Network	routing, fragmenting and quality of service
2	Data Link	detect and correct errors in the physical layer
1	Physical	actual electrical or physical transmission

Table 2.1: A basic view of the seven ISO/OSI layers for network communication

It should be noted that these layers only specify communication processes and do not include human interfaces. As a matter of fact, the human user of a machine is sometimes referred to as *"layer 8"*.

2.1.1 The Internet Protocol

The protocol stack, which is used in the Internet (called the *Internet Protocol* or *IP*), was designed by the Internet Engineering Task Force (IETF). IP was deliberately not designed with accordance to the ISO/OSI layers for several reasons. Nevertheless, it is also divided into layers [IP89], as shown in Table 2.2:

[2] In cases where data has to be exchanged between more than two peers things become even more complicated.

#	Layer	Function
4	Application	services to user-defined application processes
3	Transport	reliability, flow control and error control
2	Internet	routing, fragmenting and quality of service
1	Network	actual electrical or physical transmission

Table 2.2: A basic view of the layers of the Internet Protocol

The core entity of the Internet protocol are *IP addresses*. In version four of the Internet protocol, which is currently prevalent in Europe and North America, these addresses are 32-bit identifiers. In the upcoming version six, these identifiers are of 128-bit size. Loosely speaking, the first bits of an IP address can be regarded as an identifier of the network where a host resides, while the last bits identify the host itself.

IP addresses and networks are assigned in a rather strict manner. The highest level is currently managed by the Number Resource Organization (NRO) which is responsible for coordinating the five Regional Internet Registries (RIR). These delegate ranges of IPs to more regional Local Internet Registries (LIR) or large Internet Service providers. Eventually, single computers are assigned their addresses depending on the network they are connected to.

The success and advantage of IP, as compared to other protocols, is that previously heterogeneous networks can now be interconnected (hence the name *Internet*) with the help of a single network protocol. IP allows any computer to exchange data with arbitrary other computers in the network.

For simplicity reasons, IP does not reserve channels but is a *packet-oriented* protocol and makes no guarantees whatsoever, i.e. it provides services on a *best-effort* basis. Thus, single IP packets are independent of each other and consistency, if required, needs to be added onto higher protocol layers.

Technically speaking, layer one comprises the different physical methods of transmitting data, e.g., Ethernet or modem lines. IP-based routing is done in layer two. Then, in layer three, consistency and streams are added, if the application (layer 4) has respective needs.

For the in-depth understanding of the following chapters it will be required to have a knowledge about the details of the major components of the IP stack. We will use details from the *ARP* protocol [ARP82] on the Network layer, *IPv4* [IPV81] on the Internet layer, *UDP* [UDP80] and *TCP* [TCP81] on the Transport layer.

The Address Resolution Protocol (ARP) is used to map the physical network address of a computer to an IP address and vice-versa. This protocol is often used as an interface for different physical implementations to IP.

Internet Protocol Version 4 (IPv4) is used to route data packets between hosts.

Transmission Control Protocol (TCP) adds stream capabilities on top of IP. It manages multiple separate streams, the rate in which messages are sent, takes care of lost and duplicate packets, as well as a state-full connection establishment and termination. Due to these properties TCP is one of the most often used protocols in the Internet.

User Datagram Protocol (UDP) is a minimal application layer protocol. It is primarily used for simple applications, real-time traffic, and whenever the algorithms offered by TCP do not seem to fit the application demands.

Notably, most anonymizing networks make heavy use of the TCP protocol. However, each of the other three protocols also has a certain impact on the performance and security of anonymizing networks.

It should also be noted that there are a set of application level protocols, which play important roles:

Domain Name System (DNS) [Moc87] is used to provide a more human-understandable way of addressing hosts. As humans are not capable of remembering multiple IP addresses, the DNS service can be used to translate easy-to-remember host names to IP addresses.

This service is widely considered to be the most important service in the Internet.

Secure Socket Layer (SSL) and Transport Layer Security (TLS) [DA99] are designed to add confidentiality and integrity on top of the TCP protocol. These protocols can be used for content encryption and authentication of hosts and services.

Hypertext Transport Protocol (HTTP) [FGM+99] is commonly used for browsing content in the Internet. However, in a broader context it can be used for transmitting any structured information.

In the case of application layer protocols, there are usually only two peers involved with asymmetrical roles: the *client* initiates the protocol, which usually involves a request of some kind. The *server's* task is to answer the request with some kind of response. This *client/server paradigm* has governed the design of computer protocols from the very beginning and is still a fundamental component of today's networks.

A *peer-to-peer* architecture takes a different approach: there is no (central) server, but a set of interconnected peers. Peer-to-peer networks use rather complicated algorithms for achieving tasks in a distributed fashion. Examples of these tasks are distributing calculation tasks or storing large amounts of data.

2.1.2 Overlay Networks

Anonymous communication networks are usually built as extensions to normal networks. This allows the use and re-use of existing networks and their infrastructure, e.g., protocols on top of the *Internet Protocol* (*IP*) can be used to transfer data between (nearly) arbitrary computers in the Internet.

As long as network protocol designers stick to the upper and lower interfaces of a protocol stack they can replace or extend its functionality. This can be used to create *overlay networks*. In an overlay network, the protocol stack is split at a given point – usually between layer 3 and layer 4. In this gap a set of new protocols can be inserted, transparently for the user. See Figure 2.1 for a schematic illustration of this.

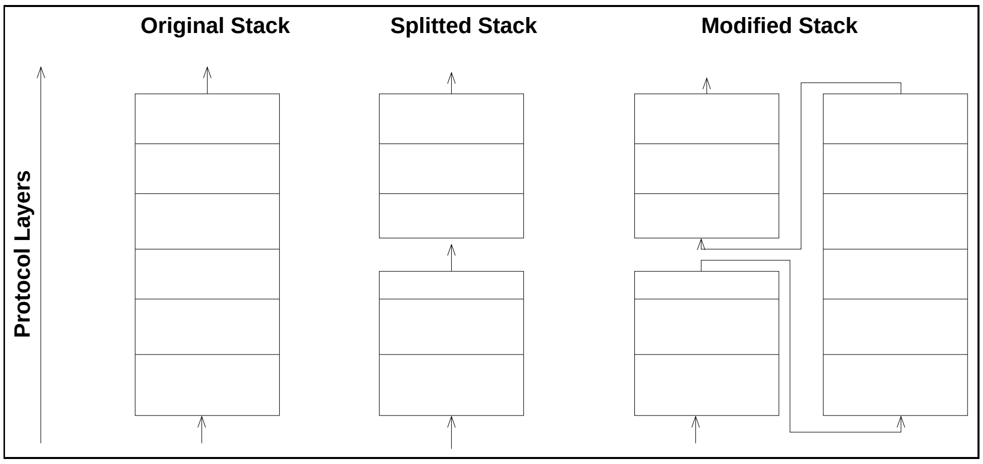

Figure 2.1: An example of extending an existing protocol stack with an overlay network.

Typically, anonymizing networks are designed as overlay networks, as this guarantees a maximum on flexibility and usability for the user. It also ensures that users can continue using their applications without further ado.

For example, anonymization service which offer anonymization of arbitrary IP data packets usually run on top of UDP – the reason is that UDP and both IP-layers (the one which they run on and the one they offer as a service) are stateless and have similar characteristics. Those, which offer anonymization of data streams and HTTP access built upon the TCP/IP layer, possibly even use (“include”) the SSL/TLS stack for encryption. For obvious reasons, this limits the amount of implementation and design overhead. Finally, e-mail-based services run upon SMTP and offers SMTP, therefore restricting the cost of design and deployment to a minimum.

While there are benefits of creating services with the technique of overlay networks, it should be noted that there are also drawbacks.

In our special case, we have one primary issue: security properties usually cannot be combined, i.e. if two protocols, each with its own security properties, are combined it is possible that the composed protocol may lose any of the properties provided by each part. This can effectively cancel the protection provided by a compound system. For example, if a malicious person is able to introduce a certain jitter into lower protocol layers, this might still be detectable on upper layers and even propagate from node to node. If the injected feature is distinctive enough it might be used to re-identify data packets on upper layers at any time later, even if the content of the data packets had been encrypted by the time they got marked.

Another example is that even if users enjoy perfect protection on the network layer, they might still identify themselves – willingly or unintentionally – on the application layer. In this case, the information leaked on the upper layers nullifies any efforts on the lower layers.

This against emphasises the importance in the area of anonymity systems to build, provide, analyse and maintain *holistic anonymity*.

2.2 IT Security

As we briefly discussed in the introduction, the area of anonymous communication is a part of IT security. For this reason, we will introduce a set of IT-security-related terminology, which will be used throughout this work.

Usually security is split into three parts [ISO]:

Confidentiality is assurance of data privacy. Only the intended and authorized recipients (individuals, processes or devices), may read the data. Disclosure to unauthorized entities, for example, unauthorized access is a confidentiality violation.

Integrity is assurance of data non-alteration. Data integrity is having assurance that the information has not been altered in transmission, or in a storage. Source integrity is the assurance that the sender of that information is who it is supposed to be. Data integrity is considered compromised when information has been corrupted or altered, before its intended usage. Source integrity is compromised when a malicious entity spoofs its identity and supplies incorrect information to a recipient.

Availability is assurance in the timely and reliable access to data services for authorized users. It ensures that information or resources are available when required. Most often this means that the resources are available at a rate which is fast enough for the wider system to perform its task as intended.

> It is certainly possible that confidentiality and integrity are protected, but an attacker causes resources to become less available than required, or not available at all.

Anonymous communication can be regarded as being part of *confidentiality*, i.e. it is about protecting information from leaking to untrusted third parties. As previously stated, anonymous communication by itself does not protect the content of a message. This has to be done with cryptographic means, in cases where it is desired and meaningful[3]. Even though confidentiality is the main concern in our topic, properties like data integrity and availability should not be harmed. At the present, however, solutions for anonymous communication are reducing data integrity and availability properties as compared to normal traffic. The negative impact of this will be discussed in the later chapters of this work.

Anonymous communication makes use of other security mechanisms (see also Figure 1.1 on page 3): Cryptography*cryptography* is used to hide information or provide data integrity. Also, *host security* is an indispensable prerequisite; in its absence an attacker might simply take control of involved computers and retrieve the data necessary to find someone's identity or learn his peers. *Network security* is compulsory for similar reasons; even if the actual content of the message is protected by cryptographic means, data packets contain additional information that possibly allows eavesdroppers to guess the content by properties like a transmissions delay, data packet size, volume, or similar properties. Even if these threats can be countered, an attacker might use unauthorized network access to, e.g., disturb the availability of a communication system, consequently rendering it unusable.

Thus, for any anonymous communication system, it is inevitable to build upon network layer security, make use of strong cryptographic mechanisms and assume adequate host security. We will give a brief survey of these three subjects in the following sections.

2.2.1 Host Security

Host security is a part of IT security whose objective it is to protect a single computer from corruption and sustain confidentiality, integrity and availability. Host security is distinctive from most other system requirements as it imposes restrictions and constraints on what the computer is *not* supposed to do. This makes this area particularly challenging because computer programs are very complex[4]. Therefore, enforcing that malicious actions will not take place would require to proof that under all possible conditions the programs and the kernel will not fail

[3]Anonymous publishing is a case where encryption does not make sense.

[4]For example, Alan Turing proved in 1936 that it is impossible to decide if a general computer program finishes in finite time given some input, or if it will run forever (the *Halting Problem*).

from an attack. Taking into account typical sizes of modern operating systems and application software, this is beyond feasibility[5].

The level of difficulty is raised even more by the fact that attackers and authors of malicious software actively obfuscate their actions and tools to mimic well-behaving software. However, it is not possible to simply stop using obscure software: some companies use obfuscation techniques for legitimate software, e.g.,Skype[6], in order to avoid their programs being analysed. Hence we can see that it is today impossible to achieve computer security by analysing the software running on a computer.

Currently, the most viable solution leading to good computer security is to use an operating system (which has to be trusted) and rely on the operating system's kernel to enforce security policies. The kernel also realizes compartmentalization, which is used to prevent misbehaving applications from affecting other processes or the complete system. Policies are implemented by either the use of white-listing or black-listing actions, or asking the user to confirm critical actions if an automated decision cannot be made. White lists define, with varying granularity, what a computer program is allowed to do (e.g., accessing certain files on the hard disk). Black lists, on the other hand, define the actions which are not allowed, for example, waiting for and accepting incoming network connections. There are situation, for example system maintenance, where it must be possible to circumvent these restrictions; if an action takes place that might be initiated by the user, he is asked to confirm that he is really sure that he wishes to complete it (e.g., deleting vital system files or installing new software).

Still, host security is a hot topic, which is mostly based on the fact that the behaviour and interaction of computer programs is too complex to be limited by a set of security policies. In addition, creating these policies is very difficult for the reasons mentioned above. Therefore, it is possible to find flaws in them which can be exploited by malicious software or attackers. Even worse, different users want their computers to do different things and therefore operating systems cannot prohibit all "strange" behavior. More often than not it happens that unwitting users find security restrictions to be inconvenient and loosen them – resulting in a vulnerable system.

Another strategy, besides trying to enforce security policies with the operating system's kernel, is to use only computer programs which were specifically developed to resist malice. However, error-free coding requires the programmer to be always up-to-date with the current state of the art in software vulnerabilities. It is easy to see that this only applies to a minority of developers. In addition, even if a software developer is knowledgeable, he might be pressed, e.g., by firm deadlines, to write code fast and thus cannot take care to write code which is free of errors.

[5]It is commonly counted that there is about one security-related programming error per 100 lines of code [McC04].

[6]`http://www.skype.com/`

To help developers, several software development tools and strategies were developed to mitigate errors in the process of software development, or at least minimize their effect. Examples include security models in high-level programming languages with automated memory management, data tainting or stack protection techniques. Still, it can be seen as a consensus amongst experts that it is best not to create software which is prone to errors in the first place. The reason for this is that defense techniques do not provide perfect protection, but only raise the bar for an intruder. Up to now it has always only been a matter of time until some adversary has found ways to break or circumvent the protection.

The most common threat in host security involves software bugs of various types. This refers to unintentionally introduced errors in software which do not handle unforeseen cases right. The result of this is that the software enters an undefined state. On a case-by-case basis an attacker can then in some cases take over or modify the execution flow of the software. From this point onwards, the software stops doing its originally intended work and is completely controlled by the adversary. He can use the privileges of this software for arbitrary actions: accessing, modifying, deleting or copying all data which the application has access to, possibly installing new software (trojan horses, virii), or anything else at his will.

To a certain extent, software vulnerabilities exist because computers strictly do what they are told to do, while they are programmed by humans who are prone to make mistakes. Basically it can be said that programmers make a number of assumptions during the process of software development, some of them being explicit, others implicit. If one of these assumptions is invalidated later on, the software continues to work with whatever data it is given. Human errors can take place on various levels:

- A researcher or software designer makes errors in the design of an algorithm. If an algorithm or concept is not suited for the task it was designed for, every software implementation of it will fail.
- A programmer creates a program which does not correctly anticipate all cases of input (e.g., passwords with more than 100,000 characters; user names containing HTML code) or which is not generic enough to work within a given context. Also, an implementation might miss covering specific cases of an algorithm, or is prone to failures due to wrong usage of a programming language.
- A system administrator installs software which has more functionality than actually needed – possibly, this additional functionality might even be undocumented[7].

[7]In this case it is impossible for the owner of the computer to disable these functions, as their presence is unknown.

Another way of software vulnerabilities to becoming relevant is if the software is not correctly configured. For example, if the default password is left in place or no encryption is enabled.

- Given that a piece of software is flawless and correctly installed, it is still possible that the actual user fails to utilize the software in the intended way. Prominent examples are users choosing weak passwords, ignoring SSL-certificate warnings, falling for phishing sites, or installing a computer virus when they actually just wanted to watch a video of a nude celebrity.

All of these faults can be triggered accidentally, which will in most cases result in the application, or the operating system crashing[8]. However, an attacker might look for these bugs explicitly and then try to gain advantage from the determinism with which the computer executes whatever instructions are located in its memory.

Some prominent examples of software vulnerabilities affecting host security are (e.g., [Sch00, HM04]):

- Memory corruptions; caused by stale pointers, too-small IO-buffers, false pointer arithmetic, integer overflows, signedness bugs, wrong length calculations, and much more.
- Input validation errors; resulting in format string errors, shell command injection, SQL injection, directory traversal, cross-site scripting, and more.
- User interface failures; giving the inexperienced user too much, i.e. confusing, information and the experienced user too little.
- Concurrency errors can be exploited in multiple forms of race conditions.

One important term in host security is *attack vector*. It describes possible interfaces which can be used by an attacker to subvert software. The more interfaces a program uses, the more likely that it will trip over unexpected input. In this sense, an interface refers to any input which is processed by the program. Besides obvious devices like network and file access, there are also user interfaces, dynamically loaded libraries, database servers, and more. Therefore, minimizing the number of attack vectors does help to secure software.

Unfortunately there is today no *metric* available to quantify the level of security provided by a computer system. Thus, it remains the domain of experts to *create*, *maintain* and *evaluate* secure computer systems. This dominance is likely to continue into the near- and middle-term future as the scene of host security is still evolving very fast and has not lost any momentum.

Finally, it should be mentioned that today all major computer systems are vulnerable via multiple attack vectors. There is even a (black) market for programs,

[8]The Microsoft Windows “blue screen” is an example of this.

so-called *zero day exploits*, which exploit previously unknown software vulnerabilities – thus there is virtually no known method to protect against these attacks. The cost of one of these starts at \$20,000 and can be as high as \$60,000 [Nar06]. To a certain degree it is therefore valid to claim that *if an entity only wants hard enough to compromise a computer system today, it cannot be defended against*.

For more information on this topic, the reader is referred to [Sch00, HM04, Bun06].

2.2.2 Network Security

The main goal of network security is to minimize the impact of network traffic as an *attack vector* on software. Thus, its target is to raise the overall degree of security in a networked and distributed system. Integrity and confidentiality on the network layer is best achieved with means of cryptography (we will come to his in the next section). Still, network security is focused on assisting[9] cryptographic means, wherever possible – and also tries to contribute enhancements in availability.

Historically, computer networks were closed environments, i.e. the number of hosts and the identity of its users were clearly laid out. This led to the development of network layer protocols which did not take into account a hostile environment[10]. Since the (commercial) success of the Internet and its global expansion this has changed: data packets are passing multitudes of intermediate systems with unknown reputations – possibly logging, dropping or modifying data in transit. Some positive properties, namely that every participant can communicate with everybody else, can under certain circumstances become a disadvantage: there is no control over malicious people sending possibly harmful packets.

Network security thus targets to mitigate the negative effect of the Internet's openness. To this end, several methods have been proposed and implemented:

- Packet filters are used to defend against unwanted network connections.
- Intrusion Detection Systems (IDS) are deployed to check network traffic for suspicious behaviour. If an IDS is authorized to actively disable conspicuous connections, is it called an Intrusion Prevention System (IPS).
- Application Layer Firewalls examine requests on plausibility and also check for possible malicious behaviour.

However, there are limitations to network-based security checks. These boil down to the problem that a tool for network security is lacking some context or

[9] Sometimes system operators even replace cryptography with network security means.

[10] It is a common semi-myth that the ARPANET, being the predecessor of the Internet, was designed to resist nuclear attacks. As described in `http://www.isoc.org/internet/history/brief.shtml`, it was designed to survive network losses. However, the main reason was that switches and network links were not reliable enough, even without nuclear attacks.

information, which only the corresponding parties know. Without the missing information, however, the tool will not be in a position to correctly decide whether the data packet or a connection is of legitimate nature and can be allowed to pass. The result is false positive and false negative decisions, whose frequency depends on the implementation of the tool, the type of missing information and the skill of a potential attacker.

The first case of information being kept back in this setup is, if the data in the transmission is encrypted. This is often the case with sensitive data, but more often than not any service can be used with the help of an encrypted connection. In these cases only the context of a connection remains as visible information for network-based security tools, i.e. they can see the source and destination, as well as the time and volume of the transmission. The actual payload is invisible to plausibility checks.

Second, the data transmitted in a single connection is often only a single piece of a complex interaction between two hosts. As the tool for network security cannot in all cases know the complete context[11], it is left to guessing.

And finally, there are ways to tamper with tools for network-based security. These, being software themselves, can be attacked and modified to ignore attacks of a certain type or simply shut down completely.

On the network layer an attacker can try to access data which was not sent to his address by, e.g., manipulating routing information or address information. Examples of attacks that achieve this are ARP spoofing in Local Area Networks, several attacks on the Domain Name System (*DNS*), or even the Border Gateway Protocol (see [KP08]). In these cases an attacker utilises the fact that these network protocols are not secured by modern cryptographic means and replies to requests can be easily spoofed. The result is that traffic gets diverted and can then be read or manipulated by the attacker.

If an attacker is on the path of a user's communication channel, or he has successfully rerouted the user's traffic to his location, he cannot only record all data packets as they pass by, but also modify and spoof information which has not been cryptographically protected. Most network layer protocols do not contain means to ensure integrity, as this would require enrollment of a cryptographic infrastructure.

Finally, an attacker can try to disrupt communication channels or means. These attacks are called *denial of service*-attacks (*DoS*), and can also be carried out by multiple attacking hosts at the same time (*distributed denial of service* attack, or *DDoS*). Their scheme involves saturating a victim's capacity with repeated requests. It often targets the available bandwidth of a host, but can also be directed towards its computing capacity, memory consumption, or hard disk usage. The result is that a system has to be shut down, or becomes unusable. In the case of

[11]If it knew the complete context, it would need to replicate all hosts which is needs to protect – something which is obviously not possible.

anonymizing networks this means that if an attacker disables a system, its users have to choose to either stop communicating or to communicate in plain.

Summarizing, attacks on the networks layer are not as potent as attacks on hosts. Often they are also very noisy, in the sense that their presence and effect is easily noticed, even by non-expert users. On the other side, network layer attacks base on the fact that the Internet is designed to be open for any kind of communication – thus they can (by design) neither by prevented, nor prohibited. And at least denial-of-service attacks are one of the most powerful and dangerous attacks available nowadays.

More information on network security is given in, e.g., [Bun06]

2.2.3 Cryptography

All systems that want to achieve confidentiality and integrity, not only on the network layer, make heavy use of cryptographic primitives. They encrypt information in order to make it inaccessible for (possible) adversaries or untrusted participants in the network. Integrity can be provided by means to detect, if data was modified by an unauthorized party.

Traditionally, cryptography has been used for nearly three thousand years to obtain confidentiality of the content of a message. Early cryptographic techniques were invented by the Spartans, around seven hundred BC [Col04]. More sophisticated techniques were developed starting from around the sixteenth century[12]. However, cryptography which was strong enough to resist most attacks was not available until the middle of the 20th century. Finally, the basic elements of cryptography are today generally considered strong enough not to be broken within reasonable time by most adversaries. This does not only refer to encryption techniques, but also cryptographic primitives which can be used to provide integrity.

Effective vulnerabilities on contemporary encryption systems can usually only be mounted with the help of side channel attacks [Koc96, BB03] or implementation errors[13].

It should be noted that while basic cryptographic primitives can be considered very strong, cryptographic protocols, i.e. protocols composed to work with elemental cryptographic building blocks, can be prone to attacks. This refers to protocols which try to achieve a more complex goal than encryption or integrity protection, e.g., establishing a shared secret between two strangers in a public environment, authenticating a user to a server and vice-versa, or offering a secure directory services which lists the hosts in a network. For this purpose, anonymizing communication is also a case of a cryptographic protocol, where combinations of basic cryptographic elements are used.

[12] The Vigenère Cipher was invented 1553.

[13] See, e.g., the havoc which was caused by a programming error, which caused a weak random number generator in the OpenSSL package of the Debian Linux distribution (`http://www.ubuntu.com/usn/usn-612-1`).

In the following part of this section we will give a short overview of basic primitives, their properties and usage: methods for symmetric encryption, asymmetric encryption, key exchange and digital signatures.

Symmetric cryptographic is the classic form of an encryption technique, where a *shared secret* is needed between any pair of peers. These cryptographic systems can be formally considered a five-tupel, consisting of an input alphabet, an output alphabet[14], an encryption function, a decryption function, and a key space. Usually there is an entity called *Alice* that wants to communicate with *Bob*, while *Eve* tries to intercept or manipulate the communication channel. The interplay of these items and entities is depicted in Figure 2.2.

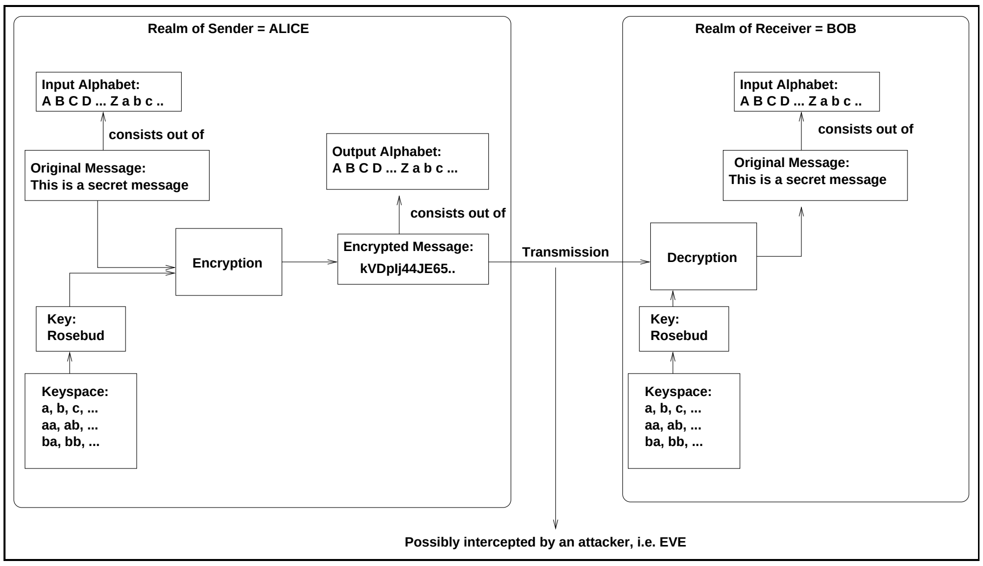
Realm of Sender = ALICE
Input Alphabet: A B C D ... Z a b c ..
consists out of
Original Message: This is a secret message
Output Alphabet: A B C D ... Z a b c ...
consists out of
Encryption
Encrypted Message: kVDplj44JE65..
Key: Rosebud
Keyspace: a, b, c, ... aa, ab, ... ba, bb, ...
Transmission
Realm of Receiver = BOB
Input Alphabet: A B C D ... Z a b c ..
consists out of
Original Message: This is a secret message
Decryption
Key: Rosebud
Keyspace: a, b, c, ... aa, ab, ... ba, bb, ...
Possibly intercepted by an attacker, i.e. EVE

Figure 2.2: The interplay of items and entities in a cryptographic system.

Most symmetric cryptographic algorithms are very strong and comparably fast. Typical key lengths are 128 to 256 bits, and throughput can be around 50Mbit per second on today's hardware, even with software implementations of the algorithm.

However, the security relies solely on the fact that the secret remains secret[15]. If for some reason a third party gets to know the secret, it is not only easy to decipher the ongoing communication, but also any intercepted and stored message can be decrypted. Another drawback is that it is difficult (without further ado) to establish a shared secret between two remote peers. Also, one would need a distinctive key for each peer, which does not scale beyond a very small group of persons.

Known and widely used examples of this kind of cryptographic elements are Rijndael, aka. AES [Nat01] and Twofish [SKW$^+$99].

[14]In our case, i.e. in computer communication, this is often equivalent to the input alphabet
[15]This property was already recommended in [Ker83].

In an *asymmetric cryptographic* system, the need for establishing a shared common secret is avoided by having key pairs, each consisting of a public and a private key. The operations of those are dual, i.e. whatever data is encrypted with the public key can only be decrypted with the private key and vice-versa. This benefit, however, comes with the drawback of a much worse runtime performance, which is 100 to 1000 times slower than symmetric operations.

For this reason, practical systems often use *hybrid cryptographic* systems. In this case, the sender chooses a key for symmetric cryptography, and sends it to her partner using public key cryptography. Afterwards, both can use this shared secret in order to use more efficient algorithms for encryption. A popular algorithm for asymmetric cryptography is RSA. Its keys are each usually 1024 to 4096 bits long, and a modern CPU can make 10 to 100 asymmetric operations per second.

Traditional approaches for asymmetric cryptography have been relying on the hardness of factoring big numbers and discrete logarithms. Recently a new approach emerged, using *elliptic curves,* also known as ECC (elliptic curve cryptography). These allow a similar security with around a factor of ten less runtime and much shorter keys (196 to 256 bits).

While asymmetric cryptography can be used to exchange keys for symmetric encryption, another way of exchanging keys for symmetric cryptography is the algorithm from Diffie and Hellmann [DH76]. It can be used to derive a shared secret between two persons without the need for an already existing covert channel. In contrast to the more deterministic way of sharing a secret with asymmetric cryptography, this algorithm can be used to transmit messages with *perfect forward secrecy*, i.e. for every single stream of communication a new key is derived and used, a so-called *ephemeral key*. After each session, if both partners deleted the secret key, it will be computationally infeasible to reconstruct the message from the intercepted cipher text.

The original algorithm from Diffie and Hellmann, as proposed in 1976, however, falls prone to man-in-the-middle attacks, i.e. if the users have no means to ensure the real identity of their partner, they can be eavesdropped. This attack does not work for a method to derive a shared secret based on elliptic curve cryptography.

In addition to the previously discussed methods for providing confidentiality, cryptographers have achieved ways to ensure *integrity* protection, namely by the way of *digital signature*. As its analogue counterpart, a signature by itself does not prevent tampering with the data, but provides a means to detect it.

To this end, a *fingerprint* is first taken from the data that is meant to be signed, i.e. it a compressed unique bit string of fixed length is created which unambiguously resembles the document[16]. This stage is created with the help of *hash* functions. The result, i.e. the fingerprint, is then encrypted with the private key with

[16]In most cases, that is. While attacks on hash functions exist, we do not go into more depth here.

the signer, resulting in yet another unique bit string with the property that only the owner of the respective secret key could have created it, together with the unmodified version of the document to be signed.

To verify a signature, the recipient applies the same hash function to the document, and decrypts the signature of the received document with the public key of the sender. If both bit strings match, it is very likely that the sender is the actual source of the message.

More information on cryptography is given in, e.g., [Sch96].

2.2.4 Conclusion

As a result from the three previous sections on computer security (2.2.1), network security (2.2.2) and cryptography (2.2.3), we'd like to come to the following conclusion:

> If the cryptographic algorithms are implemented correctly[17], they are usually not the weakest part in an IT-security-related setup.

One should note, however,that this does not hold for cryptographic protocols.

2.2.5 Attack Trees

In general, there is today no way to quantify the security of a complete system. While some algorithms and functions exist which evaluate single parts of computer systems (e.g., for cryptographic systems there is [Sha49]), no framework exists which is able to combine these into a grand total.

The most generic way, albeit its computational complexity makes it difficult to apply it in sophisticated situations, are *attack trees*. These were proposed by Bruce Schneier in [Sch00]. Since then, they have become a popular method to analyse and quantify IT security. One basic advantage of attack trees is that they can, to a certain extent, incorporate findings by other algorithms.

In principle, an attack tree is a directed graph. The sources of the network identify assets which are to be protected by the IT system. On the other hand, sinks are interfaces at which a malicious person is able to start an attack against the system[18]. At every node it is described which actions are necessary in order to continue attacks along this path. Basically, nodes can be distinguished between *and*-condition nodes and *or*-condition nodes. For the former type, if all conditions described in its child nodes have been accomplished by an attacker, then this path

[17]This is mostly true for widely used standard libraries like `openSSL`.

[18]In fact, an attack tree is still called an attack *tree*, even if it is not a tree according to graph theoretical definitions.

is open to him. In contrast, for the *or*-style nodes, it is sufficient if an attacker fulfills *any* of the child nodes' conditions.

An example of an attack tree is shown in Figure 2.3. It illustrates that, in order to get access to this (fictional) confidential database, an attacker has to either get access to the database directly, or he needs access to the operating system which hosts the database. In the latter case the attacker can simply copy the files which are used by the back-end of the database and analyse them on a different machine. An *and*-node is used in the first case, where, in order to log in, an attacker needs a valid user name as well as the accompanying password.

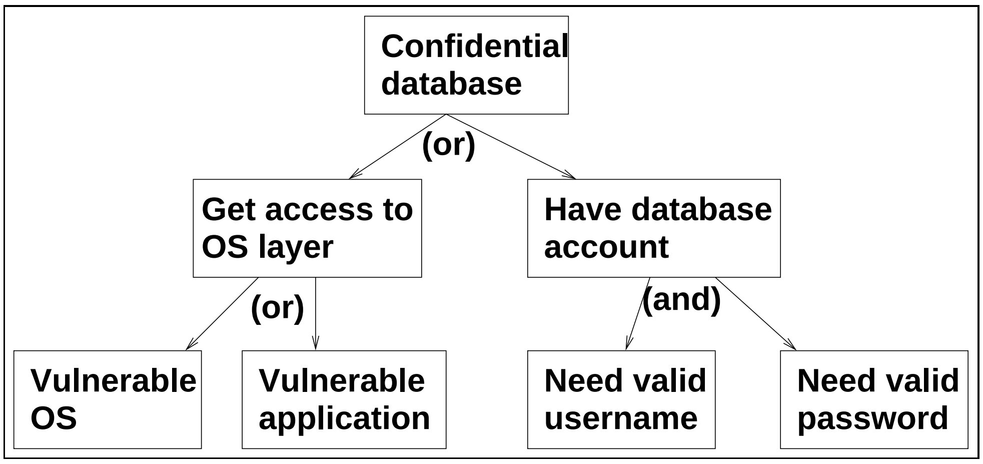

Figure 2.3: An example of an attack tree which depicts methods of illicit access to a confidential database.

Attack trees are used to determine the *weakest* link in the defense. It is assumed that an attacker does not bother to tamper with the strong parts of an IT system, but chooses the way of least resistance. Therefore, graph theoretical algorithms can be used to find the shortest ("easiest") path from the sources to the sinks. In order to save computational power, it is also possible to split the tree into separate parts and evaluate these prior to the complete system.

Improvements of attack trees assign costs to each link, as some attacks are more expensive or difficult to mount than others.

One problem with attack trees is, however, that it is very difficult to rate the difficulty and cost of single attacks, as these might vary for different people: if, e.g., an attack has a certain rare piece of software or hardware as a prerequisite, this attack is difficult to mount in general – if for some reason an attacker is in possession of the item, this attack might be actually very cheap for him.

More problematic is, though, that there is no universal method to create an exhaustive list of attacks. The most obvious fact is that even today there might be new attacks emerging against virtually any algorithm or software, at any given time. But even the existing list of attacks and their applicability is so large that hardly a single expert is able to enumerate all threats. This leads to a situation where even the most general algorithm available for quantification can only be used in the face of a certain probability of including errors.

2.3 Anonymity-Related Terminology

Finally, after we have introduced the terminology for communication networks and IT security, both being building blocks for anonymous communication systems, we start to work on the terminology for the central part of this work.

As opposed to cryptography, which protects the content and integrity of a communication, anonymous communication has the task of protecting the privacy of the context of a communication, i.e. the identity of the sender, the identity of the recipient, the date and time or the communication process, as well as the volume and possibly even the type of communication. Any subset of these can be considered the *items of interest* (*IOI*) for an adversary.

In order to enable anonymity there must be uncertainty about the true value of a certain item of interest, e.g., the identity of the sender. This means that whenever anonymity is a desired property there must be a set of decoy values which must have a non-negligible probability of being the true value. In general, it holds that

> *Anonymity* of a subject means that the subject is not identifiable within a set of subjects, the *anonymity set*.
>
> *Anonymity* of a subject from an attacker's perspective means that the attacker cannot sufficiently identify the subject within a set of subjects, the *anonymity set*. [PH06]

Depending on the setup, the purpose and the intention of the participating entities, different properties and items can be kept anonymous. Regular situations are expressed by the terms *sender anonymity* and *recipient anonymity*. Sender anonymity refers to a situation where the originator of a message would like to stay unknown. Receiver anonymity holds true if a person can be sent a message without the knowledge of his identity or location being disclosed.

It should be noted that there is currently no widely accepted metric to calculate the degree of protection provided by an anonymity system. However, there is a common ground that the size of the anonymity set is a crucial indicator, as well as the probability distribution on the subjects within this set. If the latter is a uniform distribution, then the level of protection is often considered to be the maximum possible within the given scenario. It is also important to consider that the level of protection might differ for single users of the system: it can be the case that certain individuals (subjects, items, . . .) are more likely than others.

In certain cases, when it is not possible to identify a subject, it might still be possible to tell if an action was committed by exactly the same subject as observed in a past event. For example, an observer might not know the owner of a car, but based on the car's license plate it might be possible to tell if it was the same car, which was involved in some event. This state is called *pseudonymity* and is often observed in the world wide web: nicknames in forums, sessions using cookies, and popular private e-mail addresses in the form of, e.g., `some.nickname@yahoo.com` are examples of pseudonymity.

Another important term is *unlinkability*. It describes the state of two items of interest which cannot be related by any means to each other by some third party. Of course, this state depends on the power and knowledge of the entity which tries to *link* the respective items. Thus, it might be possible that the owner of a web forum has the power to link a nickname to an e-mail address, while a normal user of the same forum might not be able to do so. Pseudonymity is one important area of influence for unlinkability considerations: as long as an entity is unable to link information to the true identity of an person, this person's identity is still very well protected. However, as opposed to a situation of true anonymity, an adversary might be able to accumulate knowledge about a pseudonymous person, therewith *profiling* him. This might ultimately lead to identification.

The term *unlinkability* can also be used to described anonymity properties. It even has the potential to allow more fine-grained descriptions such as, e.g., set-based metrics. Instead of describing a set of users being an *anonymity set*, it is possible to characterize the same fact as all of these users being "unrelated" to the message which was sent[19]. This can also be understood as: all users in the anonymity set cannot be linked to the message as a sender (or recipient). In contrast to the definition with the help on an anonymity set, an attacker might gain knowledge about the linkability of a message, e.g., by linking it to a reply message or learning its language, without reducing the size of the anonymity set.

In addition, it should be noted that unlinkability is a sufficient condition for anonymous communication, but not necessary.

As we have seen, the initial level of any protection can only decrease over time. This bases on the fact that it is usually *not* reasonable to assume that an attacker *forgets* information. This assumption bases on the fact that storage place for digital information became cheap enough[20] in recent time, to accumulate and access huge masses of data. This is prominently shown by services like the free e-mail-hosting `gmail.com` from Google, which offers each customer roughly 3

[19]It should be noted that, on the other hand, participants in an anonymizing system are at least somewhat related to messages originating from it – at least more than users who do not participate at all.

[20]In terms of money.

Gigabyte of storage for his email[21]. Another example is the abundance of sites which host videos[22], images[23] or even arbitrary files[24] for free.

As we laid down on the introduction, there are degrees of protection which exceed *anonymity*. Anonymizing networks still leak, e.g., the daytime and volume of a user's communication to a local observer. If the existence of this is also to be hidden, we call the communication *unobservable*. In a more general meaning, one can speak of *undetectability* from a certain attacker's perspective if the adversary is not able to decide whether a certain communication takes place or an item of interest is present. It is noteworthy that the properties of unobservability, respectively undetectability, are not boolean but can have varying degrees. observer

The property of undetectability can be extended from single items to complete networks, in the extreme case. In the latter case, e.g., if there is an undetectable overlay network, we speak of a *dark net*. Besides an unknown number[25] of private networks, some versions of the Freenet network [CSWH00] have claimed dark-net properties.

Profiling is a widely known method of combining the previous paragraphs, i.e. taking into account that an entity might learn information about a message without (at that time) being able to reduce the anonymity set, and storing abundant amounts of information. In the case of profiling, all data which is available about a *pseudonymously* known entity is collected. Together with similar data sets from other persons or companies, profiling allows to *infer additional* information about the main subject by extrapolation. This approach is, e.g., used by marketing companies for targeting advertisements to an audience with a higher precision. But it can also be used to *identify* previously unknown persons by means of their personal profile.

One of the more easy methods to achieve recipient anonymity is to *broadcast* the encrypted information as widespread as possible, while including the intended recipient in the set. This method achieves very strong anonymity even against strong attackers. An extreme example for this technique are *numbers station*[26] which are radio stations that transmit encoded messages on high-frequency radio signals. Given proper conditions, it is possible for these senders to reach any person on earth, thus reaching the maximum available anonymity set.

It should also be mentioned that there are some ways to achieve results similar to broadcasting, even though technically there is no broadcasting involved. A good example for this is the Usenet. There, the usenet group `news://alt.anonymous.messages` exists, which can be used to achieve recipient anonymous messaging[27].

[21]At the time of writing, it is about 2,757 Megabyte, according to the website.

[22]E.g. `http://www.youtube.com/`, `http://video.google.com/`, ...

[23]E.g. `http://www.flickr.com/`, `http://imageshack.us/`, ...

[24]E.g. `http://www.rapidshare.com/`, `http://www.megaupload.com/`, ...

[25]Due to the very nature of dark nets, their number and existence is hard to proof, of course.

[26]See, e.g., [Mas91] or `http://en.wikipedia.org/wiki/Numbers_station`.

[27]In combination with tools for anonymous posting in the web, even sender anonymity can be achieved.

On the other side, the use of intermediary nodes to transmit messages is a possibility for achieving sender-anonymity. By carrying a message over multiple hops it gets more difficult for the recipient of the message to trace it back to its origin. Another property gained with this method is *plausible deniability*: each forwarding party, as well as the original sender, can make it plausible that they only forwarded the message on behalf of another user and are not the actual originator. While plausible deniability might not always be a sufficient level of protection, it can still thwart non-decisive attackers on the system, or attackers with low resources.

One of the more secure variations of relaying messages over several hops was described by David Chaum in [Cha81]. In this proposal, cryptography (*onion routing*) is used to protect the sender of the message also against malicious forwarding entities. In addition, each forwarding entity has to collect multiple messages and forward them in a different order, therewith inducing additional confusion for an external observer. If an *onion routing* implements the latter property, it is also called a *mix*.

DC-nets [Cha88] are one known technique to achieve sender and recipient anonymity at the same time. In this scenario a group of people is formed which is then able to communicate in a perfectly privacy-preserving way: neither an external observer nor any member of the group is able to tell if, when and who communicates with whom. On the downside, this technique is quite expensive in terms of bandwidth and computational effort.

In addition to the techniques described above, users are also able to send *dummy messages*. This refers to a method of hiding the real item of interest within a set of others. For example, in the case of sending messages a user can try to confuse observers by sending a large amount of messages to different people, all of them but one being without semantic meaning to their recipients. Due to its high cost and the difficulty of creating plausible dummy messages in an automated fashion, dummy traffic is usually considered to be a deprecated method by modern standards.

Besides these abstract methods, there are software implementations designed to deliver the properties listed and discussed in this chapter. We will describe them in more detail in Chapter 3.

Finally, currently the most complete introduction into the terminology of anonymizing networks is given in [PH06].

2.4 Setup, Scenario and Setting

In this section we describe the circumstances and the *setup* of a typical deployed anonymizing network. As it might be that there are a number of private networks or dark nets, these setups will only be representative for those open to the public. To this end we shortly introduce entities related to anonymizing network and briefly discuss their role in, as well as their relationship to, these networks.

The main parties involved into the operation of a network, or within the area of its influence are (for a graphical view refer to Figure 2.4):

- Operators of relaying nodes
- Internet Service Providers (ISPs)
- Users of the network
- Providers of Internet services and operators of Internet sites
- Law enforcement agencies
- (possibly any other user of the Internet)

Most obviously, and without the need of going into more detail, the views and goals of the groups of people differ – often within each single set. In the following, we will try to elaborate in them in more detail.

It should be noted that one party is missing in the list above: the attacker. Due to the importance of the attacker to the contribution of this work he will be dealt with in his own chapter (see chapter 5 on page 71).

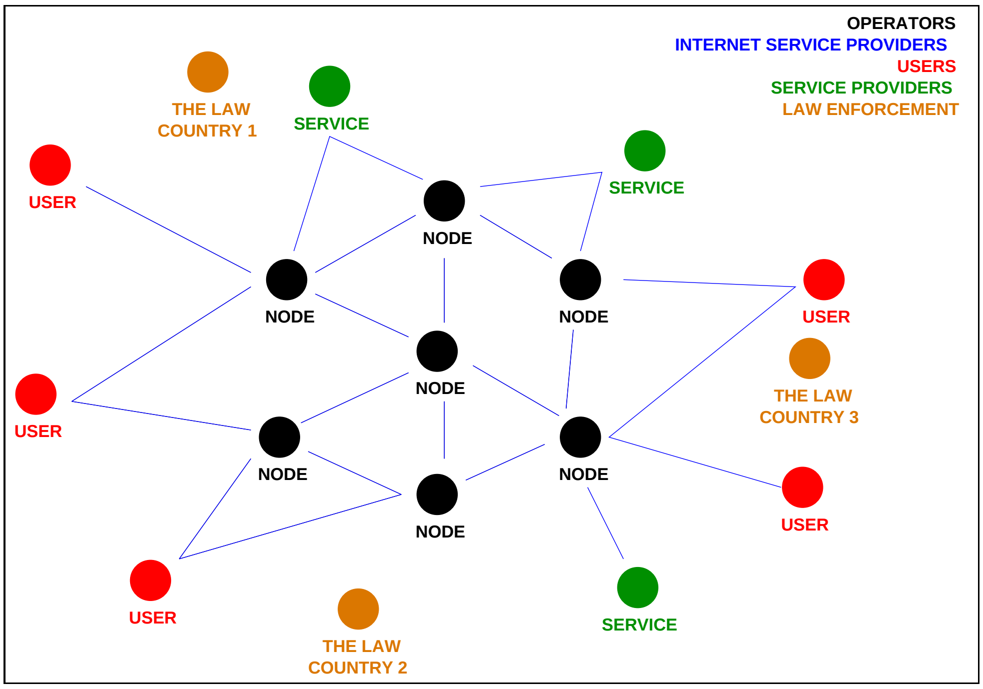

Figure 2.4: Relationship of major participating and affected entities in an anonymizing network.

2.4.1 Node Operators

Today, most deployed networks actually make use of relaying nodes in order to achieve sender anonymity. Under certain conditions, e.g., if the networks is a peer-to-peer network, every participant and user of the network is also a node operator.

Due to the architecture of an anonymizing network nodes have to fulfill high requirements for smooth operation. As public networks are free of charge, node operators do not get refunded for their efforts. In addition, whenever abuse or a crime happens with the help of the network, relaying node operators are approached by law enforcement agencies. This can lead to a house search, confiscation of equipment, lengthy interrogations and being charged of the crime committed by a different person[28].

Some networks allow to mitigate the risk for node operators to have to stand up in court for other people's abuse. By only forwarding messages to other nodes in the network, which will then eventually take over the duty of forwarding the messages to the outside, a node can minimize the impact on itself by abusers. However, a certain number of nodes has to take the risk in order to allow users to access external services.

Thus, the question rises as to what kind of motivation a node operator has to donate bandwidth and computing power to a network. Currently, no survey is known that tries to shed light on this.

The situation in slightly different, if the service is able to de-anonymize its users. This is the case with, e.g., single-hop proxies. As long as the proxy operator keeps log files of the relayed traffic he it able to resolve the sender anonymity in cases of legal disputes. While this is a method of risk-mitigation for the proxy provider, privacy-aware users are known to be reluctant to use these services.

Finally, in commercial anonymizing networks users have to pay for access. In this case, the provider can try to estimate the damage by fraudsters and adjust the prices accordingly. The targeted pricing level needs to include bandwidth and computational power as well as a share of the abuse handling.

2.4.2 Internet Service Providers

*Internet Service Provider*s, *ISP*s for short, are the companies that own and run the majority of data links, especially in the Internet. Their basic incentives for doing so are of commercial nature.

While an ISP usually neither run nodes of anonymizing service nor uses them, they own the data lines which are used extensively by these networks. Primarily,

[28]See especially `http://itnomad.wordpress.com/2007/01/04/tor-the-feds-and-me/` and following. Also in Appendix A. More, e.g., on `http://archives.seul.org/or/talk/May-2008/msg00077.html`

they should profit from the massive use of bandwidth on one side. However, the use of up to several terabytes per node and month usually does not fit into their accounting models which might base on combined costing as well as on users not requiring abundant amounts of bandwidth. The traffic also strains their networks and causes some technical glitches which in turn may affect other customers[29].

Additionally, ISPs are approached as the first point of contact by the police in cases of abuse: as the addressing scheme of networks does not per se allow to link a network address to the real identity of a user, their cooperation is a pre-requisite for any actions of law enforcement agencies which need to prosecute crimes.

As any request from prosecutors for the identity of their users results in unpaid overhead for them, they have been known to restrict operation of anonymizing nodes within their area of influence.

Related to ISPs are *hosting providers*, also known as *web hosting services* or *hosters*. These are companies which offer dedicated servers with direct access to fast Internet lines. To a certain extent, these might overlap with ISPs, as both services can be offered and provided by the same company.

The majority of traffic on anonymizing networks is relayed by high-bandwidth nodes with a good Internet connection. As these conditions are not met at a typical end-user site, the backbone of anonymizing networks is hosted in computing centres[30]. Therefore, besides ISPs, hosting providers are a primary address for the police to contact in cases of prosecuting a crime.

In contrast to ISPs, which usually are unable to control or limit the traffic they relay, hosting providers have been known to limit or restrict the operation of anonymizing nodes. In certain cases they have even been reported to commit barely legal observation techniques and methods to detect and stop the use of anonymizing networks[31].

Consequently we can see that ISPs and hosting providers have the tendency to oppose the operation of anonymizing services. The main reason for this lies in the overhead which is caused by the abusers of this system, as well as the amount of vigilantism which comes with the operation of these systems. Both finally lead to increasing costs for the provider, explaining their attitude.

2.4.3 Users

The end user is, of course, the central entity whose needs of privacy are to be fulfilled with the help of an anonymizing system. The original motivation of most

[29] As some nodes relay traffic out of the anonymizing network, they are perceived as the originators of it. It has been reported that in rare cases of abuse some harmed entities took the law into their own hands and attacked the relaying node and its surrounding. See, e.g., Appendix A. More on, e.g., `http://archives.seul.org/or/talk/Sep-2008/msg00009.html`, or `http://archives.seul.org/or/talk/Jun-2008/msg00003.html`

[30] Some are also located at university sites, but under similar conditions.

[31] See, e.g., Appendix A

designers for privacy-protecting communication systems was to help in circumventing censorship or to thwart user profiling. But, recent research and experiences from deployed systems have shown that the set of users is to a high degree, and on several levels, non-homogeneous. As the very property of anonymizing networks is to protect their users from profiling, it is very difficult to study the motivation and identity of users in anonymity networks[32].

However, as we have seen in the previous sections on node operators and ISPs, one of the major unsolved problems with anonymizing networks is handling abuse and fraudsters. Therefore, there is a certain amount of pressure on systems' designers to approach mitigation of despicable behaviour. Yet, as, e.g., the area of intrusion detection systems has shown, it seems not feasible to design algorithms for this problem. Even worse, there are cultural and regional differences on what is considered to be abuse. Drawing a line is not possible.

Despite these controversial grounds, law enforcement agencies usually differ between legal and illegal actions based on their local point of view. There also is room for some gray scale in between these two extreme positions. In most cases, no legal authority minds the use of anonymity systems as long as users adhere to local laws and only use privacy-preserving techniques to hide the traces of their legit actions.

On the other hand, there is a set of users which abuse the protection granted by anonymizing services in order to commit malice. The observed scale ranges from rather childish behaviour (inserting misleading information into Wikipedia, or insulting others), to criminal actions including blackmail, distribution of child pornography, credit-card fraud, and phishing.

Nevertheless, some empirical research has been carried out to bring light into this area. Unfortunately, early studies like [KPK05] only analyse the type of traffic which is anonymized without drawing conclusions about the users' intention or motivation. In the following we will summarize the findings of the few studies which do so.

In [MBG$^+$08] it has been indicated that there are users of anonymizing countries in most countries. During a 15-day period the authors observed clients from 126 different countries, with the vast majority of clients being present in Europe, China, the United States of America, Russia and Brazil. Therefore, it seems to be a legitimate statement that the user base of these networks is truly global.

With regards to analysing the accessed content, research is getting more difficult. There have been both user surveys addressing the users of anonymization services on the web [Spi03] and observatory approaches, relying on the classification of logged traffic into several categories [Fed05]. However, the results of the two types of studies seem to be somewhat contradictory, concerning both background of usage (with self-reports overstating professional use compared with the

[32]It should be noted that even in areas where confidentiality is of no concern, like plain web browsing, no widely accepted taxonomies on users exist.

measurement/categorization approach) and use cases. While this discrepancy may be explained with the well-documented bias of people to overstate their privacy sensitivity (for an overview see [Syv03, Acq04]), or the generally weak validity of self-report studies [OFB97], to our knowledge the publications based on direct measurements of anonymized traffic do not describe a clear methodology that would allow us to retrace how the results were obtained.

Also, it is notable that users are obviously consuming a large amount of multimedia content over, e.g., the Tor network, in spite of reported sluggishness (e.g., [DM06a, MBG$^+$08]). So, while fast response times are a big factor when browsing websites [GHMP04], for larger multimedia content, this factor seems less deterring, especially in the case of privacy-relevant contents (like adult entertainment).

For the design of the anonymizer, it makes a big difference on which level the attackers the user cares about are. For example, a system only designed to protect against a local administrator (such as the proxy scenarios described in [PP06b]) would hardly be attractive for users who are trying to protect against, e.g., a government.

On the other hand, a strong system aimed at the strongest possible attacker model would probably be more complex and provide less performance (and thus, be less attractive for users) than a system aimed at a lower level of assurance. It is, however, widely understood in economics that adoption heavily depends on perceived ease-of-use [Dav89], causing a feedback effect.

Also note that, as [DM06a] points out, adoption of a decentralized anonymizer is an important factor for both usefulness and usability. Perceived usefulness and usability, in turn, are the main determinants of users' adoption of a technological innovation [Dav89]. However, this raises a question: Why should users use a system, that aims at protecting against very strong attackers, to, e.g., merely browse adult entertainment? While we cannot answer this question directly, due to the anonymity and unlinkability properties offered by the anonymizing network, we would like to point out that:

- In contrast to proxy configurations described, e.g., in [PP06b], anonymity systems are widely deployed, and immediately available at no cost. This may of course skew demand in favor of already deployed systems.
- In many countries, especially countries with heavy usage of anonymity systems (such as Iran [Fed06]), surfing for pornography and/or specific sexual practices in the Internet may be illegal (for example, in Iran, homosexuality and adultery may be punishable by death [Mac01]). In these countries, the strongest attacker model may be absolutely appropriate when browsing adult imagery.

From the last few paragraphs we can see that there is still enough room left open for interpretation of who the users of anonymity systems really are, and what their motivation is.

Empirical Study

Because of these restrictions of the available material, we decided to do a new analysis for this paper, with clearly documented methodology. We are basing our analysis on automation using artificial intelligence, with the aim of minimizing human error, increased data protection and enabling later verification of our measurements.

As a basis for our analysis, we recorded parts of the output of a exit node of the Tor network, as well as data from an outgoing proxy of a German university at the same time. We categorize the visited websites and compared the results.

We actually restricted our analysis to HTTP traffic. [iG07] and [MBG$^+$08] point out that the largest class of traffic in normal, as well as anonymizing networks is peer-to-peer traffic, followed by HTTP. An analysis of peer-to-peer traffic, however, is not feasible within a reasonable time frame, as file transfers are highly decentralized and content analysis may not be possible due to chunking of transmitted files. Therefore, we restricted our analysis to HTTP traffic.

Also, HTTP has a long tradition of scientific analysis and there are well-known methods for analysing HTTP traffic in a privacy-preserving way. The latter was made even simpler in our case, as we were not able to see the originating IP address of a request. In addition, we did not save any information except the requested URL itself; this includes any additional header information like cookies, etc. Requests sent with methods other than GET, e.g., POST requests, or any requests containing GET variables, were completely left out of the analysis. We also filtered the time stamp of a request to store only the year and the month and mixed the request per month so as to hide any information about the order of requests and also thwarts guesses narrowing down the date of a request within a month. The processing was done in an automated fashion and with several people involved so that the researchers for the actual analysis did not have access to the original data, whereas the others only selected the URLs from the original data set[33].

These methods were applied to log files of the exit node of the Tor network, as well as log files from users which were surfing in plain. The choice of a university proxy as a control group to Tor was based on the fact that we were unable to get a more representative data set from an ISP, because the data we would have needed is either not recorded or not released for research purposes.

As a lot of multimedia content is sent via websites like image-hosting servers where the content is difficult to predict based only on the URL, we had to inspect the actual content. Therefore, we retrieved the content of about 7,000 images from each of the two sets of URLs. To avoid the result being biased by a vast amount of small pictures, used as icons or frame borders in webpages, we limited the search to images with a size larger than 10,000 bytes[34].

[33]This procedure was advised by the ICPP, a German Independent Center for Privacy Protection.

[34]Note also that small images are not perceived by web surfers as such, but rather as part of their ornamental function.

	Class 0	Class 1	Class 2	Class 3	Class 4
Plain	66%	7%	8%	8%	11%
Tor	28%	15%	15%	14%	28%

Table 2.3: Categorizing pornographic content from plain networks and in anonymizing networks

We then used a set of pattern-matching techniques (e.g., Bag-of-Visual-Words Models trained Support Vector Machines [DPN08]), to classify the images into five categories:

Class 0 definitely inoffensive images

Class 1 lightly dressed persons, might be offensive in very strict environments

Class 2 partly nude persons, might be objectionable in school environments

Class 3 nude persons, likely objectionable in many environments

Class 4 pornographic images, i.e. one or more persons engaging in sexual intercourse, likely offensive in most environments

In order to minimize the classification error, we used the automated classification method autonomously only for a preliminary result, as the pictures extracted from the URL streams were too diverse to produce acceptable classification results in a single unsupervised run. The overall correct classification rate was 33% in the case of the pictures coming from Tor and 44% for the pictures from the university's proxy. The classification rate was raised to 70% (Tor) and 75% (Proxy), respectively, if a deviation of one class (e.g., placing images from Class 1 into either Class 2 or Class 0) was deemed an acceptable error.

Because of these shortcomings, the input of the automated classification was then enhanced in a manual process of re-classification. Due to time constraints and data protection concerns we only used a random sample of 1,000 images from each set for a final manual classification. The results of the second step are listed in Table 2.3.

We used the one-sided Wilcoxon rank test to check whether the traffic from Tor contains significantly more pornographic material than the plain traffic and obtained a p-value $< 2.2 \cdot 10^{16}$ which means that the traffic fro Tor does contain a significant higher percentage of adult images.

As a result, we can identify that the Tor network has a much higher percentage of pornographic material than normal traffic: 72% vs. 34% of the pictures. Even if material from "Class 1" is not counted, despite that it can be encountered in, e.g., advertising from European countries, the percentage remains substantially higher

(57% vs. 27%). This correlates with results of related work that activities related to sexual behaviour are very privacy sensitive and therefore subject to privacy-protection techniques [TECA07].

2.4.4 Service Providers

Despite the current hype on user-generated content, which is also known under the term “Web 2.0”, the majority of content in the Internet is run and offered by companies. Even websites which take input from their users are maintained and operated by companies or organizations.

If the service offered does not allow for much interaction with a remote user, service providers typically do not be bothered about, to which extent the service is accessed anonymously. In other cases, they care: As the Web Application Security Consortium (WASC) points out in their latest online publication[35], up to 97% of interactive web applications contain software errors which can be exploited by attackers. As it is difficult to impossible to commit legal actions against malice coming from an anonymizing network, some companies have been known to block access from them.

Second, profiling customers is sometimes part of the company’s business model. By making use of relaying techniques, users reduce the company’s profit. In the case of “Web 2.0” sites, where services rely on input made by users, the company’s profit is directly linked to the value of the collected information.

Thus, unless the purpose of the offered service is widely accepted to be of very a private nature, like health related forums, service providers have a reasonable tendency to dislike anonymizing services.

2.4.5 Law Enforcement, Crime Prosecution

Due to the inherent features of protecting the sender’s identity, anonymizing systems have the reputation of attracting offenders[36]. Therefore, governmental agencies which deal with criminal investigations or prosecution of crime have to tackle problems with anonymizing networks. This includes, depending on the country, police, secret services, judges and prosecutors. Recently, even legislative bodies have become aware of related issues.

In most cases it is the task of the police to do active investigations, backed up by orders from a court or prosecutors. Thus, it is up to them to cope with the higher burden of tracing abusers in anonymous networks, rather than the normal Internet[37]. It can be easily seen that they often have a professional interest in

[35]See `http://www.webappsec.org/projects/statistics/`

[36]Up to now no public study has proven that quantity or quality of abuse in anonymizing networks is greater, equal, or even lesser, than in normal networks.

[37]Which is already difficult enough!

identifying the source or destination of some specific messages. This leads them to ask node operators for the origin or the destination of a given piece of data. If the node is operated within a different jurisdiction, however, this request will most often be turned down. Otherwise, i.e. if the police can get hold of the node operator, they might be able to trace back a single step out of many. Ultimately, there was no publicly known case, in which it was possible to trace back anonymous messages.

In this context, legislative bodies started to act. For example, the European Union decided upon the data retention act [Eur06] which in turn is to be implemented in its member states. This led to an expansion of German law which now also requires forwarding nodes to keep logs of the source addresses from messages they forward. The extent of this measure is disputed – while advocates of the data retention directive claim that this will give the police substantial help, others estimate the raising of the crime detection rate to be less than 0.1%[38].

For completeness, we would like to point out that some countries do not only have no faith in other countries' legal systems, but also actively accuse the legal bodies of other countries of being corrupt. As this is not the place for ethical discussions, the interested reader can get material from international human rights organizations, like Amnesty International[39] or the Human Rights Watch[40], for an overview of legal bodies which are considered to *abuse* their privileges.

Consequently, while it is the very incentive of legal bodies to bring offenders to justice, it might be difficult to decide whether to help them, or not. In the current state, offenders are well protected, which makes the law enforcement agencies clear opponents of anonymizing techniques[41].

2.4.6 Discussion

As can be clearly seen, the desires and goals for the involved parties differ widely. We will summarize the findings from the above sections in this analysis of multilateral security requirements.

Users prefer to obtain a high degree and irrevocable anonymity[42]. The situation for node operators is similar: from the fact that they donate time and other scarce resources to other people we can deduce that they identify themselves with the users' goals. A minor difference is that most users do not care about the set of

[38]See the statement of the German Work Group on Data Retention together with a union of judges and prosecutors at `http://www.vorratsdatenspeicherung.de/images/stellungnahme_vorratsdatenspeicherung.pdf`

[39]`http://www.amnesty.org/`

[40]`http://www.hrw.org/`

[41]Note that it is reported that some crime investigators use anonymizing networks themselves for research purposes.

[42]Bundled with good quality of service – but we are currently only discussing the level of anonymity.

other users, as long as their presence protects them. Node operators, on the other hand, can possibly be held liable for the users' actions and thus do have an interest in reducing the amount of abuse which happens with the help of their systems.

Law enforcement, on the other side, would opt for revocable anonymity, or possibly even networks in which most data streams can be linked to their original sender and recipient – the more transparency, the better. Service providers and ISPs will in most cases not take an extreme position, but rather go with a more "law and order" approach, as this eases their way to make a profit. In cases of abuse, providers will in any case contact the police, rather than seek advise and help from the anonymizing networks' operators.

Certainly, these extreme positions do not reflect the view of all users or persons working for a legal body; especially as these sets are not disjoint. But it can be trivially seen that the diversity of opinions bears some potential for controversial discussion. With no global consensus, or one side to get ahead the other by unexpected technical advancements, no solution is in sight. As social problems can typically not be solved by technical means[43]; we therefore refrain from getting into more depth at this point.

2.5 Conclusion

In this chapter we covered the context and building blocks of anonymizing communication networks. To this end we discussed:

- A rough sketch of related fields whose contribution are essential building blocks for our subject. Namely we covered computer networks and various sub-fields of IT-security.
- An introduction to the terminology used throughout this work. For homonyms we fixed a single meaning and chose a single word out of a set for clarity reasons.
- Protection goals which are targeted by our topic:
  - sender anonymity
  - recipient anonymity
  - unlinkability
  - possibly: undetectability and unobservability
- A detailed view on the entities and parties involved and affected by anonymous communication systems. A brief view of their intentions, goals and motivations was included.

[43]We would also like to recapitulate that in an open environment technical solutions are the only way to enforce properties.

Future research in at least the following areas is necessary:

- To which extent are security properties additive in anonymous networks? While this is known to be not true in general, there are possibly certain criteria in our topic which can be used for general proofs.
- What is the incentive for system operators to contribute bandwidth and computational power to anonymity systems? How can donations be stipulated and rewarded?
- The biggest threat to the deployment of anonymizing networks is abuse. Even though social problems are said to be not solvable by technical means, some questions arise:
  - How does the quantity and quality of abuse in anonymizing networks compare to the amount in normal traffic?
  - To which extent is it feasible to identify (repeating) abusers, while protecting the privacy of the others?
    Are there other means to cope with abuse?
  - Are there ways to cope with different definitions of abuse? The sets of outlawed behaviour are different in different countries.

In the next chapter we will describe the actual algorithms and systems used to achieve network layer anonymity. An overview of additional requirements for deployment is given and technical details of the implementations are discussed.

Chapter 3

Anonymizing Networks

This chapter provides a survey on mechanisms for network layer anonymity. Its purpose is to give a detailed look into the different approaches which seek to achieve protection on the subjects as defined in the previous chapter on terminology, i.e. sender or recipient anonymity.

The first part is a brief introduction into theoretical models. We show which algorithms and protocols have been proposed in order to realize privacy protection. To this end we will first discuss comparatively simple means like broadcasting messages. Despite its simplicity this can still be used to build strong systems. Then we carry on with approaches using relays in several ways. Finally, techniques are shown which utilize strong cryptographic building blocks to obtain their objectives.

In the second part we present predominant implementations of these algorithms. To start with, a short section on trivial means to achieve some anonymization against weak attackers is given. Then, we will describe and compare four networks, which have a significant user base, namely: Tor, AN.ON, I2P and Mixmaster. In addition, a concise sketch of some minor anonymity projects is given together with a list of influential, but abandoned projects.

This chapter is terminated with a short summary in which we will give some classification of the existing systems.

3.1 Theoretical Models

In this chapter we will describe important protocols which were proposed for anonymous communication systems. The basics presented in the previous chapter will be taken as a basis to deepen the knowledge about the algorithms. This section is also used to show common roots for deployed systems which are discussed in the next section.

In principle, all anonymizing networks are composed out of two functions: an embedding function and a grouping, or: cover function [KP06]. If an attacker is able to control or tamper with any of these two, the degree of protection is lowered. We can also notice that anonymity is of *multilateral* concern: a single user is unable to achieve anonymity by himself and is thus reliant on other participants.

Even though we provide the reader with a lot of details, the confined space restricts us from considering very detailed questions. An obliged reader will find more information in the overviews given in works like [KP03, HJW03, DD08] and in the specialized works on the respective sub-topic of interest, of course.

3.1.1 Broadcast

Broadcast channels can be used to achieve an excellent level of protection. However, broadcasting requires extensive use of resources in modern communication media, e.g., the Internet. In real broadcasting media, as with satellites or radio communication, access is strongly regulated and usually out of scope for arbitrary personal communication. Thus, broadcasting is only used in rather uncommon forms of anonymous communication systems.

Sending messages that everybody receives (or can receive) such that only the real recipient is able to read or decrypt them is a classical means to ensure recipient *unobservability*. For example, coded messages were broadcast by radio during World War II to the French resistance. Of course, nobody but the recipients could say which radio listeners were able to decrypt the messages, hence the recipients were unobservable. A similar situation is given with coded messages in newspaper advertisements; while all readers could be possible receivers, there is no way to know who understands what they mean.

On a computer network, broadcast allows a message to be sent to all the addresses of a given network or sub-network. Its usage is, however, constraining as the communication links of all users are encumbered. Even if we can broadcast in local area networks, it is not feasible to do it at a large scale (for example, over the whole Internet) – a prominent example why this is not recommended is spam[1].

When users receive a broadcast message they must be able to distinguish whether they are the intended recipient or not. The easiest way to implement this, as in World War II radio broadcasts, is for each user to try to decrypt the message. Depending on whether he is able to decrypt it into a meaningful clear text or not the user can understand if he is the intended recipient. This approach is called *implicit addressing*, as it works without globally visible addresses on the data packets. To simplify the process of distinguishing meaningful and meaningless clear texts, messages can of course be formatted in a particular way or contain a tag indicating it has been correctly decrypted.

When a message is broadcast with an implicit address to a set of users they form a recipient unobservability set against any attacker; except the creator of the

[1]On the other hand, spam could be used as a near-broadcast medium for transmitting hidden messages.

message who generally knows which user is able to decrypt the message[2]. If an attacker controls a subset of users, the non-controlled users continue to form a recipient unobservability set against him.

Decrypting all messages from all users which are broadcasting with implicit addresses can quickly become computationally unaffordable. In a communication-oriented context this computational cost can be drastically reduced, especially when all the packets of a communication have the same recipient. When a user starts broadcasting a communication with an implicit address, all the users attained by the broadcast will just try to decrypt the first message of the communication. If they do not succeed, they will infer that they are not the recipient of the communication and stop trying to decrypt the corresponding messages. Thus, a user will just have to decrypt one message every time a communication starts and not every time a message is sent.

3.1.2 Layered Encryption, Mixing, Onion Routing

A widely credited academic proposal to achieve additional network layer privacy was described in Chaum's seminal paper [Cha81]. In this paper he proposes a cascade of mixes to hide sender-recipient relationships.

A *mix* is a unit that receives encrypted messages, strips off the encryption, and learns two pieces of information: a new address and a message. If certain security criteria are fulfilled, e.g., a minimum amount of messages have been collected, or a certain time interval has elapsed, or both, all accumulated messages are forwarded in a random order to their respective destinations which possibly is another mix. This way, an external observer cannot trace the relationship between incoming and outgoing messages.

By cascading several mixes it is impossible even for a set of colluded mix operators to learn the sender's peers if at least a single mix is honest. It should be noted that the original message has to be wrapped in one layer of encryption per mix that is traversed.

The same protocol, but without inducing additional delay or reordering messages, is also called *onion routing*. This refers to the layers of encryption around the original message which avoid that any intermediary node learns more information other than the identity of its predecessor and its successor. In the course of forwarding a message, these layers get "peeled" off, until the last hop forwards the original message to its recipient.

This scheme requires messages to be of unified size. Otherwise, an observer can trace messages by their size. However, the overhead of padding messages to

[2]Note that in some situations it may be possible that a user encrypts and broadcasts a message with a key without being able to know to which user this key is associated with. Therefore, the unobservability property can also be held even against the message creator.

this size is feasible only for e-mail messaging or similar use cases; other protocols would require more flexibility in the message sizes and possibly also a return channel. While, in fact, Chaum developed anonymous return addresses for the system which allow addressing of users while maintaining recipient anonymity. The round trip time and jitter in a mix network, however, would be in any case considered too high for any low-layer protocols as, e.g., TCP.

The security properties of mixes are quite high: given that a subset of users is not cooperating to identify a user's peers, and at least one mix in the cascade is honest, an adversary should not learn enough information to break the security. However, if users have a fixed communication pattern, this can still be discovered if they communicate frequently.

3.1.3 DC-networks

Another basic technique to hide communication patterns is the *DC-net*, also proposed by David Chaum in [Cha88]. Instead of routing a message over a series of more or less trustworthy servers, a set of users forms a network and cooperatively uses strong cryptographic primitives in order to hide the sender of a message. Together with implicit addressing, DC-networks also allow recipient anonymity.

DC-networks are widely considered to be the most resilient form of anonymity networks and provide a strong degree of protection. However, they are also slower and more easily prone to denial-of-service attacks than any other network and protocol available. Probably, due to these disadvantages they are only rarely seen in practical implementations.

Another term for DC-networks is *superposed sending*. The basic idea was presented through an allegory of some cryptographers at dinner wishing to know if one of them had paid for the dinner, without revealing his identity (i.e., they wanted to be able to say "I have paid" with sender unobservability). Computer networks that implement the resulting protocol are called dining cryptographers' networks (DC-nets); the protocol itself is named the DC-net protocol.

In a superposed sending protocol, all the participating users send scrambled messages at each round, even if they do not have anything to transmit. This cover traffic is then used to hide the origin of the message to be sent, if there is one.

In any given round, if only one user has attempted to transmit a message, the result of the round is exactly his message. If more than one user has attempted to send a message, there is a *collision* due to specific properties of the network protocol[3]. The result is that none of the messages can be recovered from the garbled output. The easiest way to deal with a collision is to wait for a random number of rounds before trying to transmit again. More complex solutions to this problem have been proposed in [Pfi89, BdB90].

[3]More details on this can be found in [Cha88].

A set of users transmitting through a superposed sending protocol form a sender unobservability set against any attacker, including even the recipients of the message. If an attacker controls a subset of users, the non-controlled users continue to form a sender unobservability set against him, whatever the size of the subset is.

If the users participating in the superposed sending rounds are distributed over the Internet, there are serious performance issues: all of the users' messages are needed to obtain the result of a superposed sending round. Today's Internet connections have good mean throughput and latency; however, the performances are very variable from one connection to another and even at different instants for a given connection. With a superposed sending protocol, the latency of a round is worse than the largest of the users' latencies and the throughput less than the lowest of the users' throughputs. This protocol should therefore be used over the Internet for high-latency and low-throughput communication only.

In a local area network, the users' connections have stable enough throughput and latency and thus this protocol can be used to transmit even a VoIP communication flow. However, the maximum number of users is limited.

3.1.4 Other methods

Some users and designers of anonymous communication systems consider the overhead of making multiple layers of encryption too big. Thus, some researchers proposed to *randomly pass the message around* within a set of users before forwarding it to its final destination [RR98]. This protocol does not provide as much protection as the prior one, and is susceptible to more attacks, e.g., [PP07a]. On the upper side, it offers a certain degree of plausible deniability and is said to offer a better quality of service.

Private Information Retrieval (*PIR*) is a field of research associated with anonymous communication.

Under normal conditions, a user requesting an element from a database sends a request pointing out which element he wants to obtain. The database returns the requested element. This simple method is obviously not suitable, if a user would like to keep secret which item in the database he is interested in. For example, if the database may be:

- an electronic library, and which books we read may provide information about our politic or religious beliefs, or details about our personality we may want to keep confidential;
- stock exchange share prices, and the clients may be investors reluctant to divulge which share they are interested in;
- a pharmaceutical database, and some client laboratories wish that nobody may learn which are the active principles they want to use.

To protect his privacy, a user accessing a database may therefore want to retrieve an element without revealing which element he is interested in. A trivial solution for the user is to download the entire database and retrieve locally the element he wants to obtain. This is usually unacceptable if the database is too large (for example, an electronic library), quickly obsolete (for example, stock exchange share prices), or confidential (for example, a pharmaceutical database).

Private Information Retrieval schemes aim to provide the same confidentiality to the user (with regard to the choice of the retrieved element) as downloading the entire database, with sub-linear communication cost. PIR was introduced by Chor, Goldreich, Kushilevitz, and Sudan in 1995 [CGKS95]. In their paper, they proposed a set of schemes to implement PIR through replicated databases which provide users with information-theoretic security as long as some of the database replicas do not collude against the users.

In theory, PIR could be used to implement limited web-surfing capabilities, but no implementation is known to provide this.

A different approach is the use of *steganographic* network layer protocols. These are designed to hide the real data within dummy traffic, in a way to avoid that third parties do not even recognize that there is a different data stream hidden under the dummy traffic. As opposed to encryption where the volume and the mere presence of a message is clear, these techniques hide even the fact that a user is currently communicating.

A prominent example of dummy traffic is Voice over IP (VoIP). Another widely used medium are pictures: once the message has been embedded, pictures can be posted to a website or the usenet without raising suspicion. The drawback of these network protocols is the small bandwidth available in the covered channel as well as the fact that there are practically no widespread networks or implementations thereof; one proposed system was described in [WWP07].

3.2 Deployed Systems

This chapter deals with a detailed description and comparison of deployed systems.

In contrast to theoretical works which mostly focus on the actual routing of the message, there are more problems to solve in order to deploy a system (besides providing the targeted degree of privacy). Major issues include:

a directory service which tells new clients where to find relays. As the choice of servers can uniquely identify clients and the client has to trust the servers, it is a non-trivial task to design a secure directory service.

quality of service is an absolute requirement in order to gain and keep users in the network.

an update service is necessary in order to provide users with bug-fixes and other security measures. As anonymous communication is a rather new field of research, attacks emerge on a regular basis. Such it is inevitable to release and distribute regular updates for client-side software.

application layer sanitization is sometimes considered to be part of the application's task.

It should be noted that in some places the use of anonymity systems is restricted and possibly fined. Therefore, standard solutions for deploying information cannot be used, if users from these regions are targeted by a given deployment. Possibly, this leads to a "hen and egg" problem where an anonymity system is required to gain access to the system itself. Today, the only feasible solution to this problem is to find trusted third parties which provide bridges to the area which restricts the use of anonymity systems.

3.2.1 Basic Techniques

In this section we will briefly list basic techniques to achieve network layer privacy with simple means. As these techniques do no protect against strong adversaries, they are only rarely considered in academic literature. However, they can significantly contribute to the users' privacy. Depending on the information which is intended to be kept secret and the potential adversaries, they possibly provide suitable protection in combination with an excellent trade-off in performance and overhead.

Single-Hop Proxies, i.e. simple application layer proxies, are one of the currently most popular and probably easiest methods of anonymization to deploy and analyze. Typical proxy protocols are *HTTP proxy protocol* or *Socks* [Lee]. Proxies hide the identity of the sender by forwarding the request and stripping information about the request originator. The peer partner only learns the address of the proxy and is not able to see the original sender. This method can be used either by configuring a proxy server setting in an application or by utilizing web interfaces for anonymous browsing[4]. While this solution sounds trivial, it already protects against one very common adversary: the peer.

In cases the connection to the proxy server is encrypted, this method also protects against local area eavesdroppers (e.g., an local administrator or the user's ISP). Actually, given a trusted operator of the proxy, this method is reportedly used by many people today to circumvent governmental censorship and can be powerful enough to defeat even government-level blocking techniques [Mur08].

On the down side, single-hop proxies have a single point of failure and trust: a user has to trust the operator as the proxy operator on his own has all necessary data

[4]`http://www.anonymizer.com/`

to de-anonymize involved users and their peers. Of course, technically versatile users are able to operate proxies (with encryption) on their own, in which case this point is not valid, but if there are no other users they could be identified by a peer.

Furthermore, the approach is vulnerable to an attack where an eavesdropper can observe and correlate all traffic entering and leaving the proxy. Also, if no padding on the packet layer is applied – which is the case in the use of standard software – this approach becomes vulnerable to fingerprinting attacks [Ray00].

As this approach is simple, and sometimes sufficient, a variety of versions exist, which should be mentioned here. First, there are a couple of commercial offers for single-hop anonymization. These can be readily used by any user willing to pay for some protection. To which extent these services can be trusted not to identify users is unknown and depends on the individual case.

The second variant is the use of open relays, i.e. misconfigured[5] proxy servers that provide their services for anyone who knows where to find them in the Internet. As there are groups of people scanning the Internet for open relays and publishing their results on websites[6], the effort of finding open relays can be actually very small. Besides listing IP addresses and TCP port numbers of open relays, some proxy scanners also display whether the proxy has high bandwidth and if it strips off the original address of the sender.

And third, there are *bot nets*, i.e. computers that have been compromised by viruses or worms and form a network which can deploy arbitrary services for the creator of the malware. One very popular service which is chosen by the virus author is to anonymize his traffic. To this end he sets up proxy servers on the infected computers and can then use the compromised computer's IP address to commit more malice. Of course, running bot nets is illegal – but we take this as one example where a ban of public anonymizing networks would still allow criminals to retain their privacy, whereas the average end user would lose it.

A side note on IP- and MAC-address spoofing: as a computer can only participate in the Internet by using the Internet protocol, it makes virtually no sense to change the IP address in order to stay anonymous[7]. As the user is typically interested in retrieving information, he has to use his own IP address, or otherwise the requested response will be routed to some stranger[8]. Without going into further details: spoofing the IP address is only of use in a very restricted manner, and even then it does not protect against strong adversaries. The same applies to the MAC address.

[5]Most often unintentionally.

[6]`http://www.xroxy.com/proxylist.htm`, `http://www.freshproxy.org/`, ...

[7]Some tutorials claim that it even is possible to use the Internet without an IP at all – this, of course, is utter nonsense.

[8]Even if the user is not interested in the response to his request: using IP spoofing for TCP has become very difficult in recent times.

3.2.2 Tor

At the time of writing, the Tor-network [DMS04, Din08] is the network with the most number of users and related research publications. It is capable of anonymizing arbitrary TCP-streams and consists of a client-server architecture.

Clients can build circuits over the Tor network using onion routing. These tunnels can contain several parallel data streams, which relate to arbitrary TCP connections in a normal network. The only asymmetric cryptographic operations in place are used in the build-up phase of the circuits; thus adding a new stream to an existing circuit is cheap in terms of computational effort for all participating nodes.

In between nodes all traffic is relayed in a single TCP connection. This connection is protected with TLS and all data is padded to a unified size ("a cell") such that an external eavesdropper is not capable of distinguishing between traffic which belongs to an existing connection and packets that initiate new circuits.

However, before a client can use the network, he would first have to get the network information about the available servers from a cascaded cache group of dedicated directory servers. This hierarchy is ultimately controlled by a very small group (three to five) of master directory servers, i.e. their operators. The directory listing not only contains information about the servers' IP addresses and TCP port numbers, but also their available bandwidth, contact information, and their policy for relaying traffic out of the Tor network into the Internet. The latter is a means of restricting abuse over the Tor network, like spam or bandwidth consuming file sharing.

The Tor protocol also provides perfect forward secrecy, i.e. logged traffic cannot be decrypted later on, even if the secret keys of the participating parties are leaked. The same mechanism, which in fact is the use of the Diffie-Hellmann algorithm, is also used to thwart replay attacks.

As the Tor network is comprised of nodes which are operated by a number of privacy enthusiasts, the bandwidth, uptime, and computational power of the nodes vary widely. Thus, clients try to route more traffic over those nodes which announce a bigger amount of available resources. Recent statistics show a heavy tailed distribution with only a handful of nodes carrying the majority of traffic (see Figure 3.1 on the next page; note that the Y-scale is logarithmic).

Due to the distributed setup of Tor, it offers a unique range of opportunities to its users. One is that clients can choose a country where their streams shall emerge from the anonymity network – thus effectively circumventing geolocation-based filtering. The big set of Tor users (it is estimated that at any given time several-hundred-thousand users are connected [Tora, Torb]) also provides a good level of protection.

Tor also offers location-hidden services. Thus, it makes it possible to offer web services, e-mail, file storage, or any other TCP-based service under a pseudony-

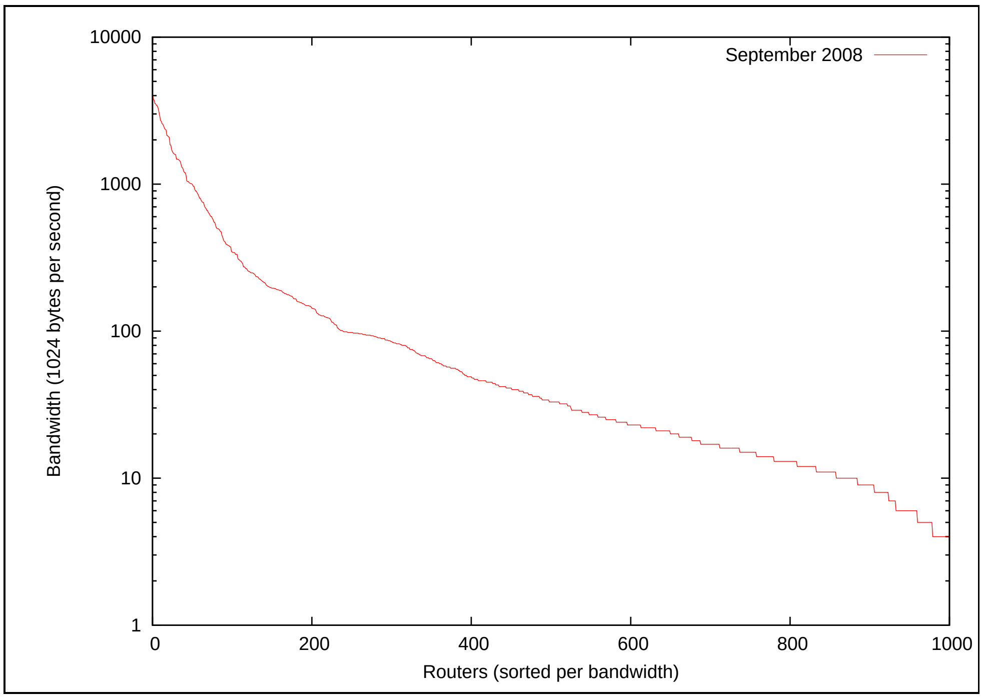

Figure 3.1: A graph depicting the available bandwidth for all running Tor routers, September 2008

mous address. Some services offer VPN-like services to tunnel arbitrary IP packets[9].

Besides there are a couple of inherent issues with Tor, too[10]. First, due to its low-latency properties an attacker can link two peers if he controls the link to the user or the first node, and the link to the user's peer or the last node. This vulnerability has been shown to be applicable in a couple of attacks and first in [ØS06]. In addition to this, the load balancing algorithm is widely considered to be a weak point: if an attacker is able to deploy even only a handful of high-bandwidth nodes, he is able to eavesdrop upon the majority of traffic running through Tor.

Another obvious problem with Tor is that it is stream-oriented and reserves static resources for circuits; if a node in the path breaks down or gets disconnected, the circuit, together with the attached streams is gone and cannot be reconstructed.

Summarizing, Tor is currently the most interesting anonymizing network from a researcher's perspective as it has a lot of open questions due to its complexity[11].

[9] `http://www.abenteuerland.at/onioncat/`

[10] There are a plethora of known caveats in Tor, e.g., the DNS leakage problem. However, given a proper implementation and integration of Tor, these can usually be circumvented.

[11] The downside is that this prevents a thorough security analysis.

In addition, based on an aggressive advertisement of its developers, it is also the most widespread used network.

3.2.3 AN.ON, also known as JAP or Jondos

A second well-known system is AN.ON which has been known under a set of different names: formerly belonging to the German project AN.ON (for: *AN*onymity *ON*line) [BFK00], the project was sometimes also referred to as JAP[12], which is actually the name of the client side-software. Recently, the project founded a commercial spin-off company, called Jondos[13]. For the sake of consistency, we will refer to this protocol as *AN.ON*.

In AN.ON, similar to Tor, traffic is routed over three nodes ("hops") and uses onion routing to avoid the hops learning too much information. It is also capable of relaying arbitrary TCP-streams.

In contrast to the Tor network, where any user can deploy servers and clients are free to choose arbitrary routes through the interconnected network, the AN.ON network offers a limited set of fixed cascades, thus the traffic is more channeled than in Tor. The cascades also need to register with a central authority. The rather small amount of them makes a caching hierarchy or other more sophisticated technologies for distributing information unnecessary. To which extent this influences the network's security is disputed: for example, the small number of cascades as well as their static IPs and restricted geographic diversity makes wiretapping and end-to-end traffic confirmation easy. Deployment of rogue servers or attacks on the directory service, on the other hand, are quite difficult.

As a cascade is accepted by the central authority only if the nodes are providing a fair amount of bandwidth, the quality of service is better than in Tor. Especially, the jitter is much lower, which results in a more consistent user experience.

On the other hand, AN.ON does not have forward secrecy, i.e. if at some point an attacker gains access to a server's secret key, he can decrypt recorded encrypted traffic. Mixing as well as a protection against replay attacks is implemented but not enabled with the default settings. Support for pseudonymous services is not given.

Despite the network having the capabilities in principle to relay arbitrary TCP streams, the usual configuration does only allow HTTP and HTTPS, as an abuse-reducing policy.

3.2.4 Invisible Internet Project, I2P

The Invisible Internet Project [I2P07], also called I^{2}P or simply I2P, is another big network for anonymous communication.

[12]The abbreviation of Java Anonymity Proxy.

[13]This refers to the term "Jon Doe".

Whereas the developers of other projects are known and can be linked to real-life identities, I2P's developers are only known under pseudonyms, if at all. To a certain extent this culminates in the fact that there is next to no academic coverage of this network. Despite this, it was designed with a higher generic abstraction than Tor and AN.ON, and targets a higher security level.

As opposed to the prior introduced applications, I2P is packet based, with its own implementation for streams to handle window sizes, retransmissions, etc. It is also possible for I2P-routers to put several smaller packets which are directed to the next same hop into a single large packet – this is called *garlic routing* by the developers. Even though it is mostly used for low-latency communication, its protocol has built-in support for variable latency, similar to the functionality proposed in, e.g., Stop-and-Go mixes [KEB98]. Instead of a central directory, I2P uses a *distributed hash table* (*DHT*) to locate other nodes.

Additional security mechanisms include different routes for inbound and outbound messages to thwart fingerprinting attacks. This also serves as a load-balancing mechanism and allegedly enhances the quality of service. Similar to Tor it is possible to offer pseudonymous services and in fact every participant can be reached under the pseudonym of his public key[14].

However, as I2P has near to no academic coverage and analysis, it is rendered efficiently insignificant despite its merits. Drawbacks are also the lack of easily available documentation, i.e. there is no byte-level specification of its network layer protocol online.

3.2.5 Mixmaster and Mixminion

The primary anonymizing network for sender anonymity in e-mail messaging and usenet postings is currently *Mixmaster*. It is comprised of a loose network of *actual mixes*, i.e. incoming messages are kept for a while in the mix and the output of messages is done in a reordered way.

A predecessor of this e-mail anonymity system was a single proxy node known as `penet.fi`. It allowed anonymous relay of e-mail messages until it got shut down by law enforcement which acted upon an abuse request. The network protocol evolved from `penet.fi` to the cypherpunk network which finally became Mixmaster.

Differences to Tor, I2P and AN.ON include not only the obviously different scope (TCP streams and IP packets on one side, e-mail messages on the other), but also the lack of a central directory authority. For the Mixmaster network, there is a set of pinger nodes that constantly measure the availability, bandwidth and reliability of the network's nodes. It is then up to the single user to choose a set of pinger node operators he trusts, or operate a ping service of his own.

[14] If the key is known, that is.

Due to this specialized design Mixmaster *seems* to be the network with the highest degree of anonymity[15].

Even though messages can take several hours or even days to pass through the network, it can be used for limited web "surfing". This can be achieved by the help of *mail2web* gateways[16]: these are services which can be asked by e-mail to download a URL. The result is subsequently sent to either an arbitrary e-mail address which is specified in the request, or posted to a usenet group. Depending on the choice of the person's need and capabilities to receive pseudonymous e-mails, the latter is the preferred way of getting the requested content. While this procedure is not suited for interactive web applications, like web mail, it can still be used to poll web forums, news or encyclopedic webpages.

A parallel development in the area of e-mail security is the Mixminion protocol [DDM03]. However, the development seems to have stalled from September 2007 on.

3.2.6 Minor Projects

In addition to the larger and well-known projects listed above, there is a plethora of smaller projects on the topic of anonymous communication.

Freenet [CSWH00] is a content-oriented network providing a distributed censorship-resistant platform for file sharing. It does so by distributing the data in an encrypted form over the various participating computers. Hence, once a file is in the network, the original source is able to disconnect without the content getting lost. Access to the files is provided by means of a de-centralized *distributed hash table*.

The level of protection which is currently provided to users accessing the content can be described as "plausible deniability" rather than true anonymity. Onion routing and multiple layers of encryption are scheduled to be included yet within the next major release. In the current version anonymity is achieved by forwarding requests over multiple hops without encryption.

Besides providing content, Freenet supports pseudonymous e-mails and usenet functionality provided by add-ons to the basic software.

In 2007 Landsiedel, Pimenidis et al. proposed *MORE*, a low-latency onion-routing network that could avoid holding any states in the forwarding nodes at all [LPW$^+$07]. In addition to saving memory for the forwarding nodes, this gives communicating nodes the opportunity to change their paths transparently at will. The downside is extensive use of asymmetric cryptographic operations.

As asymmetric cryptography is known to require extensive computational efforts, MORE uses elliptic curve cryptography, which is somewhat faster than traditional algorithms based on discrete logarithms. Even though this relieves some

[15]There is no general means to quantify anonymity.

[16]A popular gateway is located at `agora@dna.affrc.go.jp`

burden of the forwarding nodes, MORE supports a mode called *key-reuse* for an even bigger performance boost: under normal circumstances clients generate one public/private key pair for each hop and data packet which is sent. With the help of this and the server's key, a shared secret is computed which is used to decrypt the payload of the packet at the forwarding server. To increase performance, clients can choose to reuse an asymmetric key pair with a hop to avoid repeated computation of the shared secret. On the other hand, the price to pay in exchange is a (small) loss of anonymity, as the server is able to link the two data packets to the same sender.

As with I2P, MORE is a pure IP-overlay network and allows sending of IP packets; it also has support for location-hidden services. However, it does not have its own algorithms for data streams; therefore, the very high jitter brings down the performance of TCP and in consequence all application layer protocols on top of HTTP.

One central problem of anonymity networks is deployment. Without widespread use there is only limited protection, as the number of participants is generally considered to be an important factor for security. A new approach to solve this issue is by *Shallon*[Wes08]. In its requirement analysis the author states that the complexity of the protocols which are used by the major implementations is a huge hindrance factor to their deployment, but also to a thorough security analysis. Shallon was therefore designed to consist of a set of well-known network protocols (SSL/TLS and HTTP), which were assembled to form a simple and fast anonymity network – early tests confirmed a good throughput and low round-trip times.

Actually, the routing protocol of Shallon is basically a layer of encryption around a stream-based proxy protocol. In contrast to most other anonymizing networks, the users are free to choose any cipher suite from SSL. Thus, they can choose high-security ciphers with ephemeral keys or cipher suites which barely provide protection but are blazing fast[17]. SSL is also used to authenticate the servers to the client.

Shallon claims to use a protocol which is most easily adaptable, as there are ready-to-use libraries for its building blocks in virtually any programming language. In contrast to this, other protocols utilize heavily tweaked versions of common algorithms for various reasons.

A completely different approach is taken by *ant routing* [GS03]. It is used by the projects *mute net* [Mut08] and *Ants* [Ant08], both providing anonymous file sharing. Although the ant routing algorithms are well researched for ad-hoc networks, their suitability for anonymity is at least disputed, often doubted. However, no attacks against these networks or algorithms have been published yet. This can be attributed to the fact that there is only a very limited number of users present in these networks and therefore developing an attack does not "pay off". Academic coverage is also virtually non-existing.

[17]SSL even supports the NULL-cipher, which does not encrypt at all. However, the use of this cipher is not possible here.

Both networks, i.e. Ants and mute, try to keep an attacker from harvesting information about the number and identity of the networks' participants. The motivation for doing so is that it is hypothetically possible to de-anonymize traffic, if an attacker knows the pseudonym and the real network address of a user. As these networks are mere peer-to-peer networks, all the participants are also possible intermediary nodes, and any new user needs to learn some of these identities in order to connect to the network. Therefore, these networks have the dilemma that they do not want to disclose information about the IP addresses of their users, on one hand, but they need to, on the other.

To solve this problem both protocols try to make this information as difficult as possible to get. Participants only get to know small parts of it upon arrival of a new node and there is no central point where all information is gathered together at any given time. Furthermore, the information is transmitted in-band, to make it more difficult to harvest it with conventional tools. Anyway, it is legit to doubt that this method ("security by obscurity") is fail-safe, but as pointed out in the previous section, there have been no public attacks against this system, yet.

Stop-and-go-mixes (SG mixes), first presented by Kesdogan in [KEB98], are widely considered to be the most robust and efficient type of *mixes* for message-based communication, e.g., e-mail. *Reliable* [Rel04] is an implementation loosely based on this technique. As it fails to implement some crucial parts of the algorithm, it cannot provide a sufficient amount of protection [DSD04].

GNUnet [BG03] is yet another anonymous file-sharing system. Similar as in Freenet, requests are forwarded a random number of times in order to achieve plausible deniability for its users. GNUnet also makes use of a distributed hash table as directory and for searching contents.

3.2.7 Abandoned Implementations

This chapter briefly lists implementations of anonymity networks which are no longer developed and deployed.

Freehaven [DFM00] was meant to be an anonymously distributed data storage for privacy-friendly publishing – thus, a predecessor of Freenet. The project was put down due to problems with the reputation system. Also, at that time there was no effective method of achieving strong anonymity on the network layer.

An often cited system for low-latency communication is Crowds [RR98], developed by Reiter and Rubin. Unfortunately, Crowds never left research state. Still, there is a good academic coverage on security analyses of Crowds, e.g.,[ALFH04, MAFH06, VSV05].

The Six/Four-System [Hac07] makes use of the GNutella network[Gnu01]. Technically, it is designed to build circuits through a network, using some kind of onion routing with hop-to-hop encryption as well as end-to-end encryption. However, active development stopped in October 2004, and their homepage now advertises Tor as a method for secure communication.

DC-Net-chat [DCN03] was a short-lived attempt to create an instant messenger based on DC-nets. Last activity in the project was recorded end of 2003.

Cebolla [Bro02] was an approach in 2002 to achieve network layer anonymity using UDP-packets on the transport layer to build virtual tunnels.

While Tarzan [FM02] was an actual implementation, it did not receive any kind of relevant deployment in order to make statements or measurements on its performance under load or with real traffic.

Morphmix [RP02] has been proposed as a solution specifically designed against Sybil attacks[18] and collusion in general. However, due to the complexity of establishing a circuit, there was never significant deployment.

Herbivore [GRPS03], a network using DC-network technology instead of onion routing, has never left alpha-status. However, there was a prototype implementation that was used for performance studies on PlanetLab[19].

A simple proof of concept using satellite broadcasting was shown in [AGL05]. Due to the limited deployment of satellite Internet lines, project development has been stalled.

3.2.8 Overview

A short overview of the described techniques is given in the Table 3.1 on the next page. In the column *Type* we denote if the network is designed for data streams, message based protocols, or has an emphasis on transmitting contents/files. In the latter case, the targeted latency is not of high importance as even plain content-oriented networks take a lot of time to transmit files. For stream- and message-oriented networks, we listed a targeted latency in the table – this value is widely expected to correspond to the amount of anonymity which is provided to the end-user; induced latency is a countermeasure against a third party observing the messages going in and out of a mix unit. To which degree the desired protection level is reached using induced latency is uncertain as there are yet no measures to quantify the degree of anonymity provided by a system. It should also be noted that high latency makes no sense for stream-oriented networks.

In the column *Directory* we listed the type of directory which is used to tell the clients where to find the servers. Finally, the last column displays which basic technique is used to provide anonymity. From this table, and the descriptions above, we see that there is a wide variety of systems – however, there are not enough of them to fill the full cardinality of the feature space given in Table 3.1. However, we illustrated the degree of relationship between these systems in Figure 3.2.

We used the colour red to mark the four major systems. Box-shaped nodes present stream-oriented networks, while round boxes are used for message-based

[18]See Section 6.4.2 on page 102 for an explanation.

[19]An open platform for developing “planetary-scale” services, [CCR$^+$03]

	Name	Type	Targeted Latency	Directory	Based on
major	Tor	Streams	low	Hierarchy	Onion Routing
	AN.ON	Streams	low	Central	Mixing
	I2P	Messages, Streams on top	low	DHT	Onion Routing
	Mixmaster	Messages	high	External	Mixing
minor	Reliable	Messages	high	(unknown)	(SG-)Mixing
	Shallon	Streams	low	DHT	Onion Routing
	MORE	Messages	low	Central	Onion Routing
	Mute	Content	—	Internal	Ants Routing
	Ants	Content	—	Internal	Ants Routing
	Freenet	Content	—	DHT	Multiple Forwarding
	GNUnet	Content	low	DHT	Multiple Forwarding

Table 3.1: Overview on deployed anonymity networks

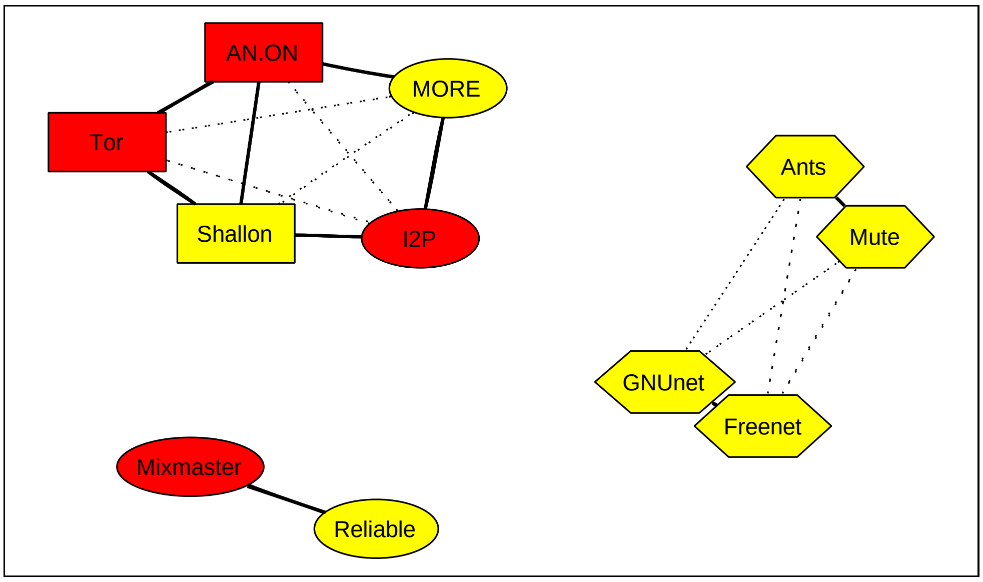

Figure 3.2: Degree of relationship between deployed anonymity systems

systems. Networks used for anonymous content distribution are displayed with hexagonal nodes. Bold lines connect similar networks whereas dotted lines represent a somewhat medium relation between two nodes.

In this figure we can see three groups: the networks targeted for e-mail messaging and usenet: Mixmaster and Reliable, the group of content-oriented networks, and the group of low-latency systems for interactive network protocols. Differences between these three groups are so high that there is currently no widely accepted way of comparing implementations from different groups.

But there is also virtually no research which compares implementations in a single group. One exception is the work of Diaz et al. [DSD04]. In this paper the two implementations of Mixmaster and Reliable are compared with regard to the amount of security they provide. The bottom line is that Mixmaster provides a higher degree of security than Reliable; however the security analysis did only take into account the routing/forwarding protocol. Vulnerabilities originating from the respective directory functionality, deployment, or implementation details were beyond the scope of their work.

3.3 Summary

In this chapter we gave an overview on theoretical approaches to support network layer anonymity. In addition we discussed the technical details of four important and some minor deployed systems.

We can conclude that there are basically three protocols which can be used to achieve anonymity: broadcast, forwarding with layered encryption and DC-networks. As both, broadcasting and DC-networks, have strong technical constraints they are today no deployed systems of significance which make use of these protocols.

In fact, most deployed systems make use of multiple forwarding, possibly enhanced with layered encryption and induced delays. We gave a categorization into three different groups, depending on their focus. Unfortunately, it is not possible to give a quantitative comparative analysis of their security properties.

Hence, future research in at least the following areas is necessary:

- Which security implications are due to each choice of directory? Is the centralized approach superior to a distributed hash table? If so, is this true for security or performance? Are there possibly other forms of directories (hybrids?) with even better properties?

- To which extent can the properties (security and performance) of each routing protocol be quantified? Is it possible to compare the routing protocols with regards to their security?
  In addition to this: all existing implementations are far too complex for security proofs. It might be beneficial to have a very simple implementation deployed in order to study it in real networks.

- Today, low-latency networks are mostly stream based. The major reason for this is to introduce states for avoiding asymmetric encryption. However, the success of, e.g., IP has shown that stateless routers have advantages. To which extent is it possible to create an anonymous network which performs well and does not need to keep state in the intermediary nodes?

In the following chapter we will switch the focus from the view on protocols and networks to a special entity related to these: the attacker that strives to break the security provided by these systems.

Chapter 4

Value-Added Works

This chapter lists works which are independent of specific anonymous communication networks and their implementation, but still try to enhance their properties. This does not only refer to security properties, but in general also to the provided quality of service, i.e. latency, bandwidth and jitter.

As examples we discuss

- a method to deploy server-side enforced anonymity. This refers to services which should only be reachable via anonymizing networks. The main rationale for this is that these services want to enforce privacy-enabled behaviour by their clients.
- broadening the scope of protocols which are transported on anonymizing networks. To this end we will discuss some issues which arise, if we add a layer of IP on top of anonymized streams.
- research which is targeted into enhancing the user experience of anonymizing networks. Especially the quality of service.

4.1 Server-Side Enforced Anonymity

One example application for anonymizing systems are webpages or online communities which offer help for issues of health, psychological problems or similar. However, installing and using an anonymizing system is known to be error prone and possibly too difficult for the average end user[1].

The main problem in the named scenarios are that the average end user is either incapable of deploying anonymity solutions by himself, does not know about their

[1]We do not know a publicly available study to show this. However, mere encryption systems have been shown to be too difficult to use in [WT99, GM05]. As anonymity systems are by far more complex than encryption systems, we can safely assume the hypothesis above.

existence or cannot judge which system to use. Another problem is that if some users access the service without anonymity service, their network address might get recorded; as any recorded data can unwillingly be abused, e.g., the server can be compromised by a hacker, it would be better to not have any data in the system at all.

The desired property the service should have is *anonymity enforced from the server-side*, i.e. there is no possibility to get access to the content provided by the service provider other than using an anonymizing network. However, there are currently no out-of-the-box solutions available for this.

Though the use of proprietary software which includes a client for an anonymizing network is one solution ("client to anonymously access the webpage for people which have prostate cancer"), the deployment of such a solution is not easy: there are a variety of different platforms which clients may use, including smart phones or PDAs. Also, the existence of such a specialised application on a device can be seen as a problem in itself: other users might not only learn that somebody is in need of some special service, they also learn which service it is.

A second way to achieve this goal would be to use location-hidden services in anonymizing networks and the use of a gateway which allows access to location-hidden services outside the overlay network. Examples of these gateways are, e.g., `http://tin0.de/` for I2P, or `https://www.awxcnx.de/` for I2P and Tor. In this case, however, all the critical information which should not be collected in the first place is present at these gateways.

While a good solution is yet unknown, we developed a hybrid solution which at least exceeds the capabilities of today's alternatives. To this end, we built an example website which enforces the use of an anonymizing network with the help of a trusted *Java Applet*. The existence of an implementation of Tor in Java [PP08] made it possible to include the protocol into an applet which in turn can be loaded into a modern web browser. It took about two person days to include the applet into a webpage and create an example webpage that loads and reloads its content anonymized over the Tor network in return.

Figure 4.1 on the facing page provides an overview of the message flow of the deployed setup. The user first downloads the applet and some initial Javascript code from a trusted third party. The Javascript functions are used to boot strap the content loading and prompts the user to choose the webpage which he would like to visit. Any further accesses to the privacy-sensible content is then anonymized, i.e. the user can then access the material in the described privacy-friendly method.

To test the implementation, we created a test webpage which was highly modular, i.e. HTTP requests were done asynchronously and multiples of them could be done in parallel. With the help of the given infrastructure it is easy to develop a framework for any kind of web application. In addition to our work it would

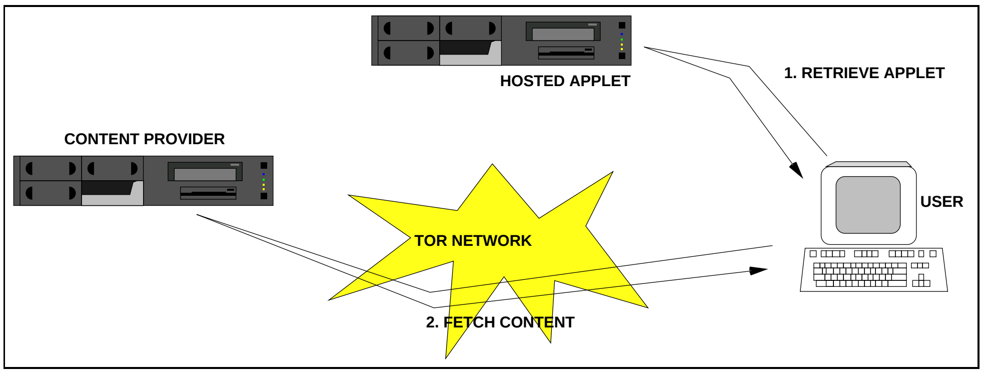

Figure 4.1: Flow of message in content provider's scenario.

be easy to extend any given framework which bases on Javascript, i.e. Rico[2] or QooxDoo [3].

The downside of this deployment is that the applet needs to have full access to networking resources, i.e. in the download process, the user needs to allow the applet to make arbitrary network connections. As these privileges are not in the standard set of web applets' privileges, modifications to the Java policy file need to be made.

4.2 Virtual Anonymous Networks

Anonymizing networks are usually realised as overlay networks, and as such they need some interface for applications to send and receive data. In most cases this interface is comprised like a traditional proxy protocol, like Socks or HTTP Proxy.

This means, however, that users can only use applications which support the use of proxies. One implication of this is that users have to configure the application they would like to use with the anonymizing network. This is known to be a non-trivial task. One way to circumvent this, and a number of other minor traps, is to use a wrapper around an application which intercepts the networking commands of the application and transparently routes them through the proxy server[4]. While this is technically considered as a very good solution, again, it requires a fair amount of technical understanding and Microsoft Windows is not capable of supporting this.

Another problem with proxy interfaces is that it does not offer support for server functionality, e.g., if a user would like to offer a service anonymously or

[2] `http://openrico.org/`
[3] `http://qooxdoo.org/`
[4] This technique is usually called *library preloading*.

under some pseudonym. While some networks actually allow such a setup, it has to be done by manual modification of configuration files. Again, this is not feasible, or at least prone to least errors, if it is done by average users.

One solution to this problem is described in [PK05]. The interface, i.e. the Socks or HTTP-Proxy, is used to connect to a VPN and thus support IP on a newly created virtual network device. With the help of such a network device it is easy to re-route all outbound traffic through the anonymizing network and also offer arbitrary services on the pseudonymous IP.

This technique is called *Virtual Anonymous Networks* (*VAN*). They introduce an anonymization layer that is hidden behind the operating systems and allow transparent access to the global network through an anonymizing overlay network. This approach assigns temporary pseudonymous IP devices and addresses to the user, that can be used to send and receive IP packets. As a result all of the user's traffic can be hidden using this temporary address. The user side configuration is consequently reduced to setup just one system and does not require any modification or configuration of programs.

The overlay network consists of two additional network layers. An anonymizer on the lower layer hides the host's real IP address against the VAN servers. We chose Tor for the demonstration purpose. The second layer is provided by a virtual private network (VPN) that provides the pseudonymous IP addresses. In our implementation, we chose OpenVPN [ope]. Both tools are installed on participating machines and can be pre-configured for clients.

Using the anonymization layer, the VPN clients are then able to connect to one of the VPN servers and receive an IP address from the server's private IP range. Consequently the client can communicate anonymously with other clients in the same virtual network. If different virtual networks are interconnected, all clients can communicate with each other, without knowing the other user's identity.

After the software is set up, every client computer will have at least two IP addresses. The first one is that of the real network device. All traffic that uses this address is not automatically protected and will be routed directly to its destination. The second IP is the one provided by the VPN server. Messages that are sent or received by the host on this address cannot be linked to its real IP address. It now depends upon the routing of the operating system as to which traffic will be anonymized and which will be sent directly into the Internet. One possible configuration could send local traffic directly, while sending traffic to remote hosts through the VAN.

An example setups of VANs is depicted in Figure 4.2 on the next page. It shows five interconnected VAN servers and four clients. The clients' real IPs are anonymized by the Tor network and their VAN IPs are used for internal communication. The picture does not show that the VAN servers also act as Tor servers and some connections between the VAN servers have been left out for clarity.

Another system which implements similar features is OnionCat[5].

[5]`http://www.abenteuerland.at/onioncat/`

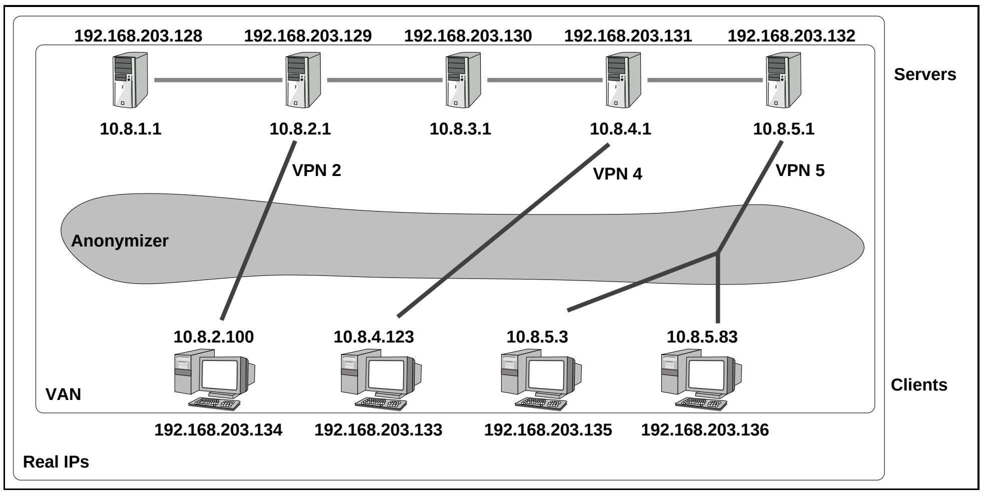

Figure 4.2: An example setup for Virtual Anonymous Networks

While the advantages of this approach are clearly visible, it also contains a number of drawbacks. The most blatant problem is in fact that users can be reached by means of their pseudonymous IP address even if they did not initiate the request in the first place. This means that attackers can make use of potent tools like nmap [Lyo] or Nessus [Sec] which are capable of identifying a host's operating system, its uptime and running services, just by sending a few probe packets and evaluating the replies.

Even worse, there are a number of administration utilities[6] which provide versatile passive fingerprinting of incoming connections. This is made possible by the slightly different methods by which operating systems generate IP sequence numbers, TCP timestamps, IP ids, source ports, etc.

Hence, together with the raised number of possibilities offered and discussed in this section, also a certain higher threat to the degree of anonymity arises.

4.3 Quality of Service

Available practical implementations suffer from poor performance [PPR08, WHF07]. This results in a stalling participant number of anonymizing networks, as users are known to be impatient and only willing to wait a short time in order to get, e.g., a requested webpage [Köp06]. Even more, due to the increase of multimedia content transmitted over the Internet in the recent years, the requirements on bandwidth are drastically increased. While the backbones and access networks of the plain Internet were adjusted to current needs, anonymization networks like Tor are not able to meet the demands.

[6]e.g., `http://lcamtuf.coredump.cx/p0f.shtml`

On an abstract level there are two possible remedies, to enhance the situation:

- More bandwidth can be added to the network by increasing the number of nodes. This requires not so much technical effort, but rather advertisement and incentives for more people to run nodes. To which extent this kind of campaign might be successful was covered in Section 2.4.1 on page 31.
- Making better use of the bandwidth available to the network. While this means will not be able to gain arbitrary more bandwidth, this issues can be tackled with better routing algorithms and thus is subject to intense technical research.
  On the other hand, by using more sophisticated routing techniques it is also possible to enhance a network's latency and reduce jitter; both of these can only be marginally improved by new nodes.

In the following part of this section we will give an overview on works which follow the second path to enhance the quality of service for anonymizing networks.

First, an accepted tolerated latency of about 4 seconds for a website request has been demonstrated in [WHF07]. Another study [RES03] shows differences between polychronic cultures (e.g., Saudi Arabia) and monochronic ones (e.g., Germany) in terms of delay-acceptance during web browsing. Users from polychronic cultures were eager to accept longer delays than those from monochronic ones. For the use of anonymizing systems, another reason for a higher delay tolerance of the users from Saudi Arabia might be the following: due to strong censorship they have higher incentives to wait longer in order to browse anonymously. Subsequent research [Köp06], however, shows only marginal differences between the tolerated waiting times in different cultures and a linear relationship between increased delays and the drop-out-rate of users. Since the strength of the anonymity provided by such a system is usually considered to be linked to its number of users, the protection for the remainders is weakened with each user leaving the network.

Ideally, all clients participating in the network would select the nodes to be used in virtual circuits uniformly from the set of all currently active routers. Since the probability to be included in a path is the same for all routers, this method offers the maximum achievable anonymity[7], but at the cost of performance. The latter is due to the fact that routers with a weak performance are chosen with the same probability as very powerful nodes having abundant resources.

Works which have been published in this area include: Rollyson [Rol06], who proposes a method to improve the client performance in Tor by a modified method of path selection that is based on latencies between routers. The proposal requires the Tor directory servers to provide a list of router-to-router latencies that can be consulted by the clients when choosing a path. The proposed algorithm is limited

[7]When no metrics based upon QoS criteria are involved when choosing paths, attackers cannot influence the path selection of clients – except for operating more routers.

to picking only the middle node of circuits in an efficient way, because of issues concerning trustworthiness of entry and requirements for exit nodes. Since Tor directory servers do not provide lists of link-wise latencies for clients, the author proposes to use an approximation technique that is based on measuring latencies between responsible DNS servers [GSG02].

Snader and Borisov [SB08] propose an opportunistic bandwidth measurement mechanism for Tor nodes. It is based on the idea of assigning a capacity value to nodes equal to the median of the peak bandwidth all other nodes recently experienced to the given one. For path selection itself, the authors propose a cumulative distribution function based on quantiles in order to reduce the influence of single routers advertising very-high-capacity links. According to their own measurements, though, the opportunistic bandwidth estimation is less accurate than self-advertised values from the descriptors. The former, however, cannot so easily be manipulated by malicious nodes. Bauer et al. [BMG$^+$07] demonstrated that by artificially increasing bandwidth reports, an attacker can compromise 46% of all circuits while controlling only 6 out of 66 routers in a Tor network. The authors evaluate the performance and anonymity of their proposal regarding different system parameters and compare them to the vanilla Tor. For quantification of anonymity, the Gini coefficient is used. In this case it is a measure for the equality of the selection probabilities. Further, according to their evaluation, it is possible to improve the performance while providing the same degree of anonymity as it is currently the case in vanilla Tor, or to improve the anonymity without significantly affecting the performance.

Sherr et al. [SLB07] aim on the design of anonymity networks, while they focus on meeting application-specific performance and security constraints, rather than optimizing only one of these. The proposed system, Application-Aware Anonymity (A^3), provides three rather obvious approaches for path selection: completely random, random with constraints, and one with a so-called tolerance parameter regarding constraints. Due to the abstract description of the approaches it is not directly comparable to the other proposals.

Panchenko, Pimenidis and Renner [PPR08] studied the performance of Tor under various circumstances in order to detect bottlenecks, as well as to learn about the overall situational behavior of the network and limits of nodes regarding performance metrics like latency and throughput. Furthermore, they show the influence of geographical diversity of routers in a path on the performance of circuits. The paper justifies that client performance can be improved by choosing routers that are located geographically close to clients, respectively destinations. The diversity of the nodes in a path, though, is an important substance to the security of anonymity systems [MZ07, FD04]. Choosing nodes located in different countries involves different jurisdictions, and thus, increases the protection of users. Additionally, if the chosen routers are located close to each other, it is more likely that they belong to the same operator. Therefore, the authors propose to achieve performance improvements by other means, i.e. by performing routing based on geo-independent performance metrics.

Murdoch et al. [MW08] explore the effectiveness of path compromise with regard to the Tor's default path selection algorithm as well as to the methods proposed in [SB08]. Their metric of security is the probability of compromising a path by controlling the first and last hop. The cost of an attack is considered in terms of the number of nodes available to the attacker, as well as the bandwidth available for each node. The main result is that in the presence of a node-rich but bandwidth-limited attacker the Tor's default path selection algorithm offers improved protection compared to the uniform path selection algorithm. Thus, the vulnerability of the path selection is not affected that much because of proliferation of bot nets. These usually have a large number of nodes with a high geographical diversity, but poor upstream bandwidth.

TorFlow [Per] is a multi-purpose framework with the general aim to improve the performance and the security in the Tor network. It contains an extended implementation of the Tor Control Protocol [DM], a text-based protocol that allows to implement *controllers* with the ability to control a running Tor process by listening to events and sending commands. TorFlow-specific extensions include additional features to support path building, while preserving arbitrary restrictions on the properties of the generated paths. These were also used to implement the methods proposed in this work.

4.4 Summary

In this chapter we have displayed some works on enhancing the performance of anonymizing networks. Despite some excellent advancements in this area, a few words of caution seem to be appropriate: even though it might seem like a good idea to use elaborate algorithms for a higher utilization rate of the available bandwidth, preliminary studies have shown that *any* additionally provided amount might not be enough[8].

Future research in these areas is manifold:

- deployment at the user side is still a major problem. This goes hand in hand with the ubiquitous weakness of IT security products used by end users: the interface (especially the graphical user interface) is often not comprehensible and thus cultivates harming behavior.

- the experience of users with anonymizing networks is closely connected to the quality of service. However, especially in this area, it is very difficult to provide a good level of quality; aggressive attempts to do so will even cost a lot of protection.

[8]See, e.g.,
`http://www.lightbluetouchpaper.org/2007/07/18/economics-of-tor-performance/`

- again, a means to quantify security is missing, especially to measure the degree of anonymity. Thus, it is currently not possible to size the different levels of security provided by QoS-enhanced routing mechanisms. Hence, it is impossible to recommend or prefer one over the other, even if the exact requirements of the user are known.

In the next chapter, we will work out details about attackers, i.e. entities whose target it is to compromise the security of an anonymity system.

Chapter 5

Attackers

Life in general would be much easier if people did not have opposing interests. However, this is not the case and thus there are entities whose interests include to learn information on the whereabouts of specific persons. If this happens without the consent of the observed person, these entities are called *attackers*[1]. We have seen in Section 2.4 on page 29 that there are a number of different participants related to anonymizing networks. Interestingly enough, next to all of them, plus external entities, could be attackers to the network.

The main reason for the highly detailed discussion in this chapter is that, despite some merits of the theoretical foundation laid by the early academic papers in this area, existing models are not fine-grained enough when it comes to the properties of real-world attackers. To a certain degree they also miss to differ between widely different concepts which exists in deployed networks. In addition, some of the theoretical models of attackers more no distinction between two classes of attackers, where no distinction is necessary or possible in a real network, or at the very least: does not make sense.

Therefore, it follows that it is not easy to design secure anonymization networks for real deployment using theoretical attacker models. For the given reasons any implementation will likely fail to provide adequate security in some places, while providing it unnecessarily in others.

To estimate whether an attacker will be successful in breaking a real system or not is part of a security evaluation or risk analysis. One critical part of this is to properly define a realistic attacker model. If the chosen attacker model is too powerful – most of the protection techniques will be unnecessary. If the attacker model is too weak – the system will inevitably provide false and undesired means about protection level of its users.

A common example for an often used theoretical attacker model with no real-world equivalent is the *passive global observer*. This refers to an entity which is

[1]Compare this also to e.g. [Bis02].

able to monitor traffic on all network lines on earth, but does not have the capability to inject or modify data. While we agree that this model is interesting for mathematical analysis, end-users should be aware that theoretical results based on this analysis are not representative of real scenarios: an attacker having the capabilities in the real world to intercept traffic at the global scale can typically also easily alter and manipulate the traffic and, therewith invalidate the results of the analysis and protection vision of the end-user.

Seen from another perspective: it is also not realistic for an average end-user to defend himself against an adversary which is capable of observing the whole worldwide network because of two reasons: first, such a powerful adversary can make use of more efficient means in order to obtain the same information, and secondly most end-users are not an actual target of these kinds of adversaries.

In addition to that, different users are concerned about different attackers each. While a typical European user might, for example, be concerned about profiling websites or a prying ISP, users from, e.g., the Middle East or China, have to fear severe punishment for surfing on webpages with political, sexual, or arbitrary other content.

To make the situation more complex, users sometimes confuse potential attackers, i.e. those which have the capabilities to harm them, and real attackers, i.e. entities with the opportunity, the capabilities and the intent to harm them[2]. Common misunderstandings of threat scenarios include a fear of European citizen to be spied upon by foreign secret services. Even if this is true [Tem01], there are by far more dangerous entities for the average end-users, as we will see.

Besides giving a taxonomy on attackers and a detailed description of them it is therefore inevitable that users are able to correctly identify their personal threat model. This goes along with good practice in security evaluation, where choosing an appropriate attacker model is a necessary precondition before any security evaluation can start. This was known and practised even more than two thousand years ago: “If you know your enemy and you know yourself, you need not fear the result of a hundred battles.” [TzuBC].

This chapter gives a detailed view on attackers to anonymization networks. First, we will list theoretical models as proposed in research papers. Second, we develop a more practical-oriented attacker model which can be used for analyses of deployed implementations. To this end, we will cover the properties and possible motivations of an attacker to create a new taxonomy of attackers.

At the end of this chapter we will have a good understanding about what and who an attacker is, what his typical capabilities are, and which goals he wants to achieve.

[2] It is noteworthy that this also applies to other scenarios like, e.g., physical security. One famous example is sexual harassment: while children are taught not to trust strangers, the majority of sexual assaults are committed by people close to the victim ([LNW07], page 64).

5.1 Theoretical Attacker Models

The purpose of theoretical attacker models is to make statements about the security of abstract models of anonymization networks. The main problem with applying these models to real systems is that their level of abstraction is too high in order to make relevant statements about fine-grained real systems. The origin of assuming very strong attackers originates from the area of cryptography. There, it is has become reasonable to defend against nearly arbitrarily strong adversaries. As anonymous communication is a rather young field, the primitives developed and deployed so far are not as strong. Thus an attacker originating from a “theoretical attacker model” would usually either break an anonymizing system with no effort, or would have no chance against it.

While theoretical attacker models can be used, however, to find basic statements on security properties even for real systems, these results again only hold – in theory. The reason for this is that these analyses work with models of networks, rather than with real networks. Hence they assume, for example, that the implementation of the protocols is perfect or that users behave uniformly. However, both assumptions are not true in the real world.

In general, one should be cautious to assume that security proofs based on models and theoretical analyses can hold up in deployed systems. This is due to the fact that by modelling a network there will be *necessarily* some loss of accuracy, which in the end will possibly invalidate the result if ported back to the original system.

5.1.1 Simple Attacker Models

Some attacker models in literature are quite simple. While this can be correct from a theoretical point of view, it raises difficulties in cases of the risk estimation in the real-world settings. In [WALS02] the adversary is described as a participant that collects data from its interactions with other participants in the protocol and may share its data with other adversaries.

[SS03] describes an attacker as some entity that does passive traffic analysis and receives the data by any means that is available, hence it is some form of *maximum attacker*. These kinds of attacker models might be interesting in certain special cases but are difficult to generalize and identify in a real system: depending on the influences these attackers might have they can be completely different entities. So, for example, they can be a secret service or a standalone hacker, each being a different threat to the end-user. Also, the means that should be taken in order to provide the protection depend on the concrete threat entity.

5.1.2 Simple Taxonomies

A more general attacker categorization is given, e.g., in [KP03]. The authors introduce three classes of attackers with increasing amount of power and capabilities, namely the *global external passive attacker*, the *passive attacker with sending capabilities* and the *active internal attacker*. While this distinction makes sense within the context of the paper [KP03] because it helps to show a difference between Mixmaster and Stop-and-Go-Mixes, the difference is marginal in real systems. We agree that a purely passive attacker is different from an attacker that also participates in the network and is possibly detectable. On the other hand, it's quite unlikely that an attacker that has global access to network lines does not also have the possibility to inject messages. So, the first two attacker types wouldn't differ in their capabilities in real systems but rather in the decision whether to make use of all their features. Also, access to anonymizing networks is not only not restricted but even actively endorsed (*"Anonymity loves company"* [DM06b]); thus, there is also no real difference between the second and third class in a real network.

Similar arguments apply at [SDS02], where the authors propose to split a *global active attacker* into the one that can only insert messages, and the one who can delay messages. However, if an attacker is able to deterministically delay messages in a real system, he will also be able to insert messages – the reason for this being that there is no way to actually delay a message in a real system other than removing it and reply ("inject") it at a later time. On the other hand, if an attacker is able to insert messages in a system and observe their effect, he is most probably in control of some part of the system and thus also able to delay messages.

A more detailed list of adversaries can be found in [HJW03], where four attacker types are listed: the *eavesdropper*, the *global eavesdropper*, a *passive adversary* and an *active adversary*. Again there will be little difference between, e.g., the global eavesdropper and an global adversary in practice. We leave the proof as an exercise to the reader.

5.1.3 Taxonomies

The most systematic listing of attacker types for theoretic modelling is found in [Ray00], where Raymond introduces three dimensions of attackers:

internal-external Attackers can be distinguished on whether they are participants in the network or not.

passive-active Attackers can actively change the status of the network or remain passive.

static-adaptive Attackers can't change their resources once the attack has started or they can continue to build up their capabilities.

An additional dimension is given by Pfitzmann in [Pfi04]: active attackers can either limit their actions, follow the protocol and thus reduce the chance of being detected, or trade-off their stealth in favor of more powerful attacks by committing actions that are not part of a network's protocol.

The most realistic attacker model can be found in [STRL00] where not only the method of attack is provided (ranging from an observer to a hostile user or a compromised network node) but also the extent of the attacker's influence on the network (i.e. whether it's a single node or some large parts of the network).

5.1.4 Discussion

As we have seen there is a wide and scattered range of attacker models in theoretical works. For obvious reasons it is not feasible to unify them in a reasonable framework. One more issue which we would like to draw attention to, are the notions of an *active attacker* versus a *passive attacker*.

In theoretical attacker models, there might be a difference between a passive and an active attacker. In this very basic notation, we denote an active attacker as someone who tries to modify an anonymizing system in order to gain advantages, whereas a passive attacker reduces his set of actions to those that do not alter the attacked system. While being active seems to be the superior mode, passiveness usually includes a smaller probability of being detected as an attacker by the other participants in the system.

Usually, the following actions are considered passive, and are unlikely to be detected by the other entities running the system:

Recording traffic , e.g., eavesdropping.

Resolving identities of users, in order to map network addresses to real people.

Reading log files in case they would be accessible to him by some means, e.g., if the attacker is a system's operator.

Breaking cryptographic primitives is considered a passive action, as this usually done off-line and without the users noticing.

On the other hand, these actions are usually considered active:

Injecting, dropping, altering, or delaying messages

Denial of Service attacks

Please note that neither of these lists is meant to be exhaustive.

If we take, however, a real-life situation, the sets of actions converge: both imply that an attacker has physical access to the communication lines, in which case he can of course commit passive record as well as active manipulations. We can therefore conclude, that if an attacker is passive in a real-world scenario, this is rather based on the decision to stay invisible, rather than due to missing capabilities.

Strangely enough, there are actions carried out on a regular basis by real-world attackers which are usually not covered in any theoretical model. This means that these actions are simply ignored by traditional research:

Compromising servers or clients in the network.

Profiling application layer data in order to identify users by their behaviour, writing style, specific applications or versions of applications, operating system, etc.

Providing bait information in order to lure users out of their anonymity by tricking them into distinctive actions.

Side channel attacks on hardware layer, or the operating system layer.

Summarizing this section, it seems inevitable to develop a novel system for attacker classification. To this end we take a look at the properties of real-world attackers and introduce a comprehensive classification.

5.2 Practical Attacker Model

In this section we develop a novel attacker model for anonymous communication systems. The motivation for this is to overcome the limitations and ambiguities in the existing theoretical works. After we have described the properties which should be fulfilled by a classification, we continue with a list of properties inherent to attackers. Furthermore, we describe their motivations and finally provide a taxonomy which fills the gap between theoretical attacker models and real networks.

As given in [Amo94] and [How97], a taxonomy should have classification categories with the following characteristics:

mutually exclusive – classifying in one category excludes all others because categories do not overlap,

exhaustive – taken together, the categories include all possibilities,

unambiguous – clear and precise so that classification is not uncertain, regardless of who is classifying,

repeatable – repeated applications result in the same classification, regardless of who is classifying,

accepted – logical and intuitive so that they could become generally approved,

useful – can be used to gain insight into the field of inquiry.

We will use these attributes in order to verify the validity of our newly developed attacker model.

5.2.1 Properties of an Attacker

As much as humans differ, so do attackers[3]. However, all of them have at least one thing in common which is the fact that they are attacking an IT security system. It is also safe to assume that in the majority of cases the attacker likes to learn information which is hidden by the means of anonymizing network. This *intention* together with an attacker's *action* distinguishes the attacking entity from normal users, operators, unrelated third parties, etc.

Consequently, the attacker is then interested in some or all items of interest which are related to one or more peers of the system. For example, the adversary might be interested in either a user's peers, the identities of all parties regularly polling a webpage, or who sent a specific message.

As we are currently not interested in estimating the total effort of an attack, we will assume in the remainder of this section that the attacker is only interested in a single item of interest related to a single user. We can do so without loss of generality, if we decompose all other scenarios to be multiple or repeated instances of this.

Other than this, we can identify several properties and capabilities in which attackers differ:

Physical Influence This capability can be used to gather information from nodes and lines in the network. In fact, we consider the owner of IT equipment to have physical influence over his infrastructure.

In addition to known eavesdropping techniques physical presence and influence may be used for physical capture and extraction of information out of computers, network equipment, printed papers, and persons. Depending on the attacker's choice and possibly other properties these actions take place stealthy or noisy.

A practical unit for measuring this would be the *area of physical influence*. For example, a single user does not have any significant authority beyond his dial-up line and beyond maybe some kilometers around his physical location. Governments have quite a huge physical influence which contains the home state and possibly even limited influence on allied territories[4].

Computational Power is a capability which is easy to quantify, e.g., in FLOPS. It can be used in traditional ways to break cryptographic primitives, e.g., by brute force. In our context it can also be used to set up a lot of different (rogue) nodes in the network, or simulate participants to lure a victim by pretending these nodes are honest users. Other uses for computational power include *data mining* on captured data and doing traffic analysis on network flows.

[3]Unless we take Hollywood movies for real.

[4]See, e.g., the extradition of Gary McKinnon as happened in 2008 (`http://www.freegary.org.uk/`, `http://en.wikipedia.org/wiki/Gary_McKinnon`).

Knowledge and Skills are unconditional prerequisites for an attacker. In order to compromise a system in a targeted way, i.e. such that it fails to provide protection but rather allows the adversary to learn whatever he is curious about, it is inevitable to know the inner workings of the system.

In our context this means that an adversary can be proficient in any areas as discussed in Chapter 2: computer networks, IT security, cryptography and anonymous communication systems, or any subset thereof. These also include the knowledge about typical vulnerabilities of anonymizing networks and common mistakes that users make. The potential *attack vector* grows with the amount of the attacker's knowledge and increases chances of success.

The skill of exploiting known vulnerabilities is of substantial significance. It influences the probability of success and the amount of noise generated during an attack.

To a certain extent, e.g., in cryptography, analysts assume that an adversary knows all public information as well as algorithms in use. This has been even understood to be of importance since 1883 [Ker83]. As anonymous communication itself is still a rather new area of research, it is safe to make similar assumptions.

Hence, we will assumed that the attacker knows the *infrastructure* of the network, the *algorithms* that the network bases on, and *strategies* that are deployed[5].

Potentially, this property can also be used to develop *new* attacks.

Man Power refers to amount of *human resources* at the attacker's disposal. This can be used for large-scale social engineering, or broad penetration of large anonymizing networks. It is also useful for all kinds of impersonation attacks.

Legal Influence refers to the ability of an entity to either commit actions which are of privileged character in a jurisdiction (for example, physical capture of other people's computers), or the property to commit these actions without the threat of being prosecuted (for whatever reason).

Most actions which are related to attacking computer systems are either illegal, or at least falls into a legal grey area. Exceptions are only on a base-to-base case, whenever an attacker is legally permissible to an action: For example, a father may legally access the computer of his son, a company may access all of their workstations, and law enforcement may access all computers in a country, given they are in possession of a warrant.

[5]This is a commonly used assumption, thus we intentionally omit a long list of references. See, for example, `http://www.freehaven.net/anonbib/`

Therefore, the amount of actions which an attacker can commit within a short time interval is somewhat related to his legal influence. However, it might also happen that adversaries ignore these restrictions: malevolent persons might be willing break laws in order to obtain information.

Additionally, for practical security evaluation purposes there is the possibility that an adversary originates his actions from a different jurisdiction, thus thwarting all of his risks being legally prosecuted by the victim. As in fact any person can use resources within different jurisdictions in order to commit attacks, which makes legal prosecution at least very difficult. This circumstance should also be taken as an incentive not to rely on laws in order to protect assets in the Internet.

Money can be used to a certain degree to substitute a lack of any of the above properties and capabilities. For example, it can be used to hire employers, private investigators, or researchers. Also, hardware and bandwidth are legal goods to buy.

However, money can also be used to hire hackers, thugs, henchmen, or to bribe people.

Risk averse vs. Risk ignorant makes a distinction between the attitude of attackers, to either ignore the possible consequences of illegal and risky behavior, or not. This property makes a difference, even though this work does not consider the aftermath of an attempt to attack a system, regardless if it was successful or failed.

A number of attacks either take a lot of time or consist of multiple steps. Hence, if the first steps are somewhat of legal risk, an attacker might be scared off practising these as preconditions to the following steps of an attack. Another case is given if a preparative action is noisy, i.e. easily detectable by the victim: a risk-averse attacker might be too cautious not to call attention to his presence and possibly not be able to continue the attack with the necessary second step.

In fact, not all of the properties listed above are equally important: for example, vast computing power does not help attackers, unless they have a huge physical presence and are able to collect enough data which can be processed in a second step.

As we have already discussed, it is a reasonable assumption that a dedicated attacker will be up to date with academic literature and additional knowledge, having read documents describing typical vulnerabilities. Basically, this skill allows an attacker to choose attacks from either the complete set of attacks, or a subset thereof.

Since attackers might be willing to break laws in order to accomplish their goals, and possibly are able to avoid prosecution, it seems reasonable to only marginally consider the amount of legal influence of an attacker as a major point of distinguishing classes of attackers. As we have seen, this is of central importance especially in the case where attackers come from different jurisdictions. However, legal influence can reduce the cost and risk of certain attacks. As we do not consider cost in this chapter, this variable has no influence on our attacker classification.

Finally, it can trivially be seen that money is a wildcard, having mostly no effect on its own, but being a substitute for any other property except *Risk Affinity*. The later is special in a certain sense: while the other attributes are properties which can be acquired or traded to a certain extent, risk affinity is likely to remain unchanged.

This leaves us with the most influential variables *Physical Influence* and *Man Power* being the two most important variables to distinguish different attacker types. *Computational Power* and *Risk Affinity* are ranked at second level. Finally *Legal Influence* and *Knowledge and Skills* only play minor roles.

These interdependencies, plus typical classes of attacks that are possible with the given properties are listed in Figure 5.1.

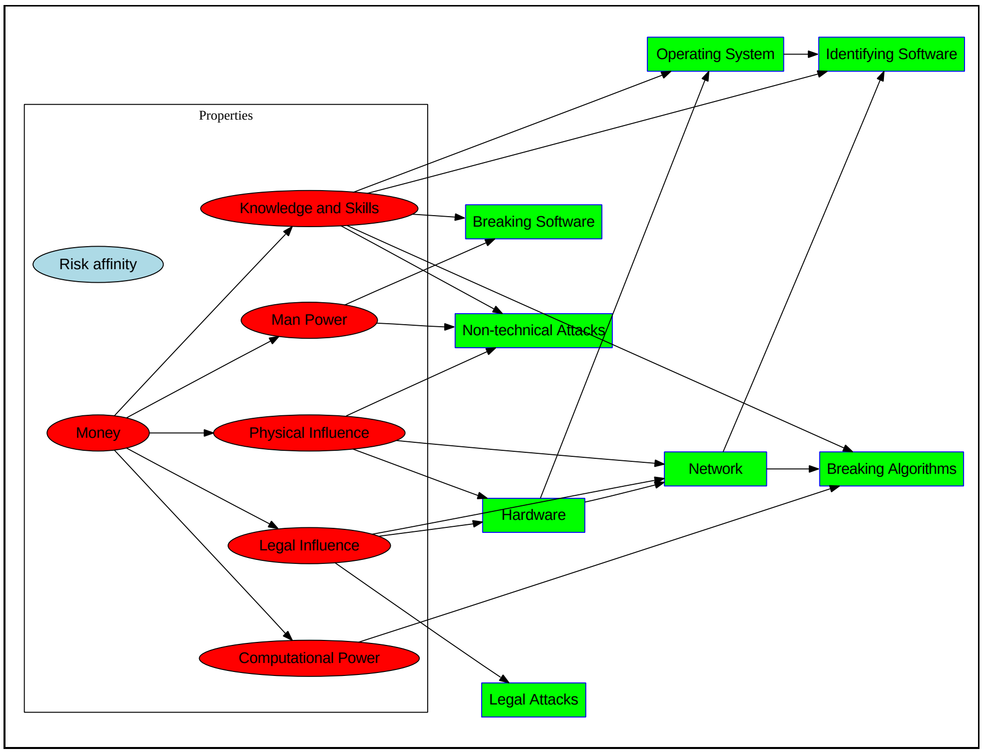

Figure 5.1: Overview on Properties of Attackers and their interdependencies

As can be trivially seen, the variables are, with the given exception of *money*, mutually exclusive, unambiguous, repeatable and useful. Since a preliminary version of the model has been published in [PP06b], we claim it to be accepted, too. However, we are not able to claim a (pre-)final degree of exhaustiveness. The reason for this is that our model consists of more variables than other existing attacker classifications. Our model even has a finer granularity of variables than attacker models in the general field of IT security [How97, Rog05, Hec05]. Hence, we at least claim to fulfill a sufficient coverage of attacker's properties.

5.2.2 Attackers' Intentions and Motivations

If, however, we would not only try to evaluate the probability of an attacker to be successful, but also assess the damage afflicted, than we would have to take into account which motivation an attacker has, and what he will do with the information gained from his attacks.

Thus, in addition to the possibilities of an attacker, it is important to know which incentive and intentions an attacker might have. For example, even if a secret service can easily spy on arbitrary citizens, the probability of an actual (targeted) attack is rather low. In general this holds true if the perceived gain of an attack is too low in relation to its costs. Conversely, the owners of bot net (*herders*) do attack people because of negligible cost and risk, and the possible gain of yet another additional computer in their bot net. In this section we list attacker's *motivations* and *intentions*. These play a central role in an attacker's subjective perceived measurement on gains and costs of an attack. Hence, we see that it might be possible to defend against *some* attackers by raising their perceived cost of attacking.

Works describing the motivations for attackers on computer systems originally copied jargon from the world of counterespionage, using the acronym *MICE*, which translates into *money*, *ideology*, *compromise* and *ego*. This was extended in [KAS04] by Kilger et al. to *MEECES*, and refers to

Money as the most self-evident cause. An attacker can abuse an anonymizing network infrastructure in order to make money.

One of today's most easy approaches to this would be to operate an exit node of the network and wait for users to make network connections to the webpages of a bank or similar critical webpages. In these cases the attacker can either eavesdrop or modify the users' requests, or impersonate the bank's online site, in order to place money transactions in his own favour.

A more passive attacker could plainly eavesdrop credit-card numbers from the network traffic exiting from his node, or profile users in order to mount large-scale impersonation attacks or commit identity theft.

Entertainment is one of the less dangerous motivations for an attacker to have. Usually this attitude implies no direct will to harmful behaviour but rather pranks that are to be played on the users. In our scenario the damage that can be inflicted by this motivation is quite small, and can be easily circumvented.

Practical instances include redirecting users' requests for websites to shock websites[6], replacing images in business websites with pornographic pictures, or modifying stock-exchange information which is forwarded through their area of influence.

Ego is one of the most driving factors for individuals to attack computer systems. As also discussed in [KAS04], it is highly unlikely that high prison terms for crimes have an impact on individuals to reduce their activities, but rather gives them new incentive to do so.

Cause (Ideology) is a motivation often shaped by political influences. Therefore, the impact of this motivation is roughly determined by the power of a political group in accordance with their technical savviness.

Law enforcement agencies and maybe small groups of political activists form the main representatives with this motivation.

On the other hand, companies are usually not to be found within this category.

Entrance to a social group, especially a group of active computer hackers, is a motivation to commit damage to other people's computer systems. However, this motivation is most likely for single attackers with a limited skill set, therefore reducing their impact on, e.g., world wide distributed systems like anonymity networks.

Status is a similar driving force like *entrance*, de-facto being the follow-up which takes places after joining social groups.

However, the above reference to an attacker motivation is not complete. In "Scene of the Cybercrime" [Shi02] (pp 113 ff.) a generic attacker on computer systems and his intentions are described:

Just for fun is described as a motivation for people with no deliberate intention to deliver harm to others.

Monetary profit has been discussed above.

Anger, revenge, and other emotional needs is a motivation that leads to timely actions, usually committed by individuals.

Due to the international deployment of anonymizing networks and their size, it is quite unlikely that unplanned attacks will not result in severe damage.

Political motives have also been discussed above.

[6]E.g., (no, sorry, not in this work)

Sexual impulses are among the most severe drivers for people to commit crimes. However, since these are only significant for individuals rather than groups, corporations, or agencies, we regarded them as a special case of emotional needs in our case.

Serious psychiatric illness is a similar case to sexual impulses.

It is obvious that both lists focus mostly on individuals rather than organizations.

In addition to the motivations above, there are two special cases of motivations which were experienced in real networks:

Research is a motivation to break anonymizing networks and find out their vulnerabilities. The effect of this kind of attackers is mostly similar to those which act due to *Just for fun* in [Shi02] or *Entertainment* in [KAS04]. This means that attackers with this motivation do not have the intention to deal harm to others; most often they even use their attacks to improve the security for the attacked networks later on.

However, as opposed to the types of attackers that do it *Just for fun*, researchers usually have a deeper insight and broader skill set.

Law enforcement purposes, i.e. handling of a network's abuse, are motives for agencies and social entities that strive to punish sanctioned behaviour. Forbidden behaviour may include distributing pornographic material, political opposition or terroristic activities.

While in total, the basic motivation for these attackers is based on political grounds, one can say that the actions of these attackers are not illegal, since they act upon (in democracies) a legal legitimation of the public.

In Figure 5.2 on the following page we depict a short summary of the motivations. It also gives an a priori estimation on the degree of danger related to the motivation, as well as the man power of an adversary.

5.2.3 Classification

Based on the previous sections on attackers' properties (5.2.1 on page 77) and motivations (5.2.2 on page 81), we propose a novel scheme for attacker *classification* in anonymizing networks.

As we have seen above, the attributes that distinct real-life attackers best are the *amount of man power* and the *amount of physical influence the attacker possesses*. The latter also correlates with the number of nodes and links that the attacker controls or which are within his reach. To a certain extent, with increasing man power, there will also be an increase in physical influence.

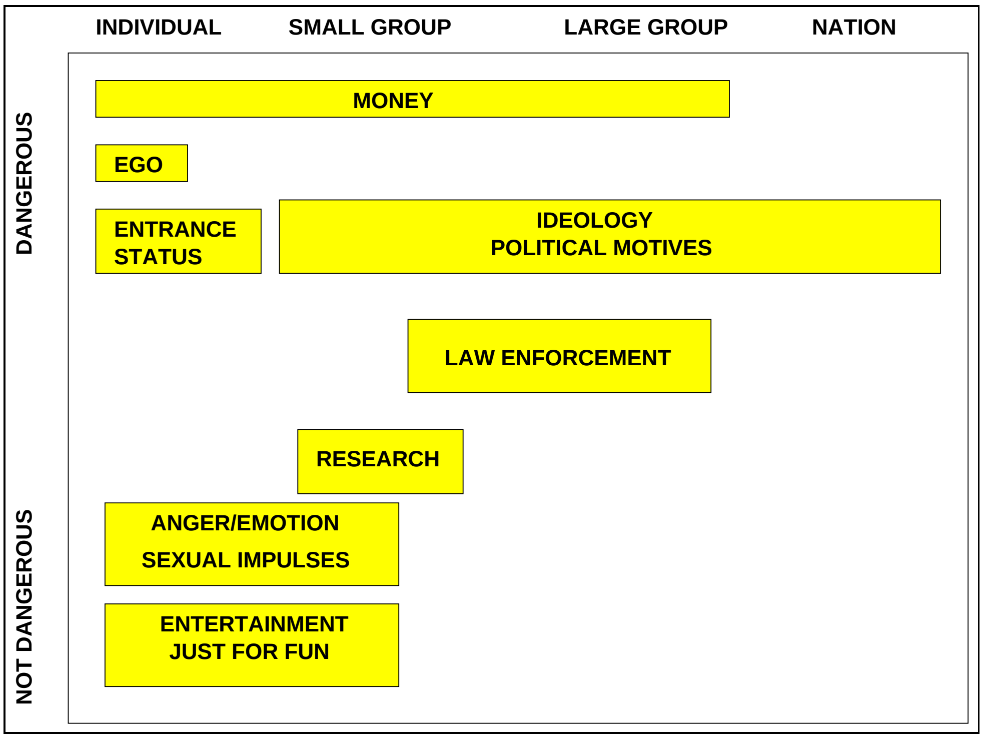

Figure 5.2: Differences between attackers' motivations.

Furthermore, computational capabilities are not as relevant in today's scenarios because cryptography is usually too strong to be broken. Hence, breaking of cryptographic primitives is only seldom a preliminary to successful attacks on anonymizing systems.

On the groups as prepared previously we thus created the following classification of attacker types [PP06b]. This specification is independent of a specific network's infrastructure and topology, as well as coherent with the findings we had in the previous sections.

The classification can also be seen as classes of entities and social stereotypes participating in, affected by, or being interested in a transaction between two parties using an (anonymizing) network.

0. External Party The least powerful class of attackers has no control of any computer between the two communicating parties.

While this kind of "attackers" is hardly worth consideration, this class forms an excellent base check: if even these attackers can successfully compromise an anonymizing network, surely problems exist. Thus, countermeasures should be taken to prevent them from gaining information. Since failing to do so will result in a system which is neither secure nor confidential.

Note that external parties can be very powerful, e.g., competitors in international trade. While the initial position of an external party is limited, it might be possible for this attacker to levitate its position, depending on the network used and parties involved: e.g., he can lure a victim into communicating with him, hence gaining "peer" status. Another possibly is to participate in the network as a node operator and wait for the victim to choose these nodes as forwarding nodes.

However, unless this class of attackers undertakes actions to increase their influence on anonymizing networks, their power is limited to, e.g., collect public information which is available on the world wide web, the usenet or any other public source.

It should be noted that systems sometimes leak information to outsiders even if they're not supposed to do so, e.g., some implementations of e-mail servers allow to check if a given e-mail addresses exist without actually sending them an e-mail. [MD05] is an excellent example of an attack which allows external parties to infer information about the inner workings of an anonymizing network.

1. Service Provider/Peer This class of attacker represents the victim's communication partner in scenarios where the victim does not only communicate with a closed group of entities. This attacker is technically bound to the receiving end of the communication and its close neighborhood.

In addition to all sources of information that the external party has access to, the peer learns (by definition) the content of some of the victim's communication. This includes application layer information. This class of attacker is also able to commit quite elaborate attacks against the victim: first, he can try to manipulate the victim on a social level and persuade him to, e.g., use different, i.e. non-anonymous, means of communication. This attacker can also try to inject malicious content into his messages which target to exploit the anonymous victim's software and then take control over his computer.

More subtle attacks include profiling the victim's software and learn his operating system, language and time zones, as well as individual writing style [ZLCH06].

2. Local administration In contrast to the victim's peer, this attacker can eavesdrop and manipulate data in the network close to the victim. This includes, but is not limited to sniffing data, manipulating DNS-responses, man-in-the-middle attacks, denial of access to anonymizing networks in order to force plain communication, and much more.

These capabilities of this attacker class are very powerful if the user trusts all received and transmitted data or is clueless about the abilities of these attackers.

On the other hand, this attacker can be easily evaded once the user manages to establish a secure connection to an outside trusted relay with strong encryption. Yet, there is still a small information leak remaining as, e.g., the number and size of transmitted data packets can be used to identify certain webpages [Hin02].

3. Internet Service Provider (ISP) The next most powerful class of attackers mostly resembles the capabilities of the previous one. However, this class has access to a significant larger number of computers and network lines in the vicinity of the user. The amount maybe so large that it can even be a non-negligible part of the whole global network.

Thus, while a local administration can be circumvented by means of a relay outside the attacker's scope, the relay would need to be in another country in order to thwart this attacker. Given the interdependencies of big ISPs (particularly multi-national companies like T-Mobile, AOL, Tele2, and similar), however, there are chances that nodes in other countries are operated by affiliates of them.

4. Government This class of "attackers" does not only have the power to access a significant portion of all networks but also has large resources to setup *honey traps*[7], break simpler encryption schemes[8] or prohibit access to specific services.

This adversary is also able to take measures that are illegal or impossible for others: confiscating hardware and other material, or introducing laws on data retention are only two of them.

5. Secret Services are forming the highest class of adversaries. They can be assumed to either have access to most parts of the global networks or they can get the access to it if they think it is necessary for their operation.

To a certain extent, it can also be assumed that this class of attacker is also not bound by any kind of laws[9].

It should be mentioned that the latter two types of attackers will probably have the highest advantage by using non-technical methods to get information – this includes but is not limited to the physical capture of nodes.

In addition, it should be noted that some countries deploy their secret services also for industrial espionage [Tem01].

[7] A computer service which fulfills the task of an *Agent Provocateur*.

[8] The German Federal Office for Information Security (BSI) factored the RSA-640 number in September 2005: `http://www.rsasecurity.com/rsalabs/node.asp?id=2092`

[9] Consider, for example, the kidnappings of alleged Al-Quaida terrorists by the CIA: `http://dip.bundestag.de/btd/16/003/1600325.pdf` or `http://news.bbc.co.uk/2/hi/europe/6368269.stm`

We deliberately specified the categories in a way which is intuitively understood by researchers as well as by end-users. Note also that these classes must not be seen with strict boundaries, hence real-life attackers can be found at a certain point somewhere on this scale.

At this point, we would like to put a side note on *collusion* of attackers. Collusion of different attackers is a commonly seen act in traditional analysis of anonymization networks which we deliberately did not covered in this model. The reason for this is simple: if attackers cooperate in real life, they are not independent of each other. Either they can be found to be in a "natural alliance", like two peering ISPs, or in a clear command chain, like, e.g., a government forcing ISPs to cooperate by means of laws. A second reason for not handling collusion is that (in our model) the resulting power will rather be the power of the bigger partner.

5.2.4 Evaluation of the Attacker Model

We now evaluate the classification given in the previous section according to the characteristics defined in [Amo94]:

mutually exclusive – with the exception of the victim's communication partner, we have created the categorization in an increasing order of power. The attacker which comprises the peer partner does not intersect with any other class.

exhaustive – the classification gives a list of entities having an increasing influence over the user's communication. The scale starts with entities having little impact and next to no power over the user, and ends with entities with a global view, a huge budget, and free of any legal or ethical obligations. Therefore, we have covered the complete range of (human) adversaries.

unambiguous – While it might not be possible to uniquely determine a real-life entity classification within this scale, it should be possible to give at least two adjacent points on our classification list in order to identify the power of a real-life adversary.

repeatable – Due to the natural description of the entities, this characteristic is naturally given.

accepted – The proposed scheme shows basic similarities to those as given in, e.g., [Rog05], [How97], and [Hec05].

We also published this scheme at [PP06b]. Therefore, we can assume a necessary minimum degree of acceptance.

useful – as we demonstrate in the upcoming chapters, this model can be used for a detailed in-depth security analysis for anonymizing networks.

From these points we can see that our classification fulfills all but the first characteristic very well. The first characteristic is still fulfilled to a very good degree. However, as it was pointed out in [Amo94], it is very hard, if not impossible, to find a metric that complies with all of the given characteristics.

Thus, we can see that this model forms a valid classification for threat analysis and exceeds existing attacker models.

5.3 Summary

As we have seen in this chapter, there is essentially no significant related work on attacker classification in anonymizing networks. As a consequence, researchers have to either work with over-simplified attacker models, or adapt them individually to their needs – the latter will, however, result in work which cannot be compared to other researcher's results.

The most far-reaching outcome of research results based on wrongly chosen attacker model is that they possibly defend against the wrong adversaries. Thus, the resulting algorithms are often impractical to deploy or rely on unrealistic assumptions.

We tried to overcome this situation by proposing a novel classification for attackers that is realistic, i.e. adaptable to real deployed networks, as well as suited for security evaluations. As we have shown, our taxonomy is currently the best solution to access the problem of attacker classification.

Table 5.1 shows the relation between the entities as we discussed in Section 2.4 on page 29 and the classes of attackers, as developed in this chapter. Positions marked with a filled cycle (●) mark likely relations. I.e., if an entity which is involved in anonymizing networks would attack the system, it is likely to be an attacker of this class. This also holds true vice-versa: if an attacker from a certain class considers attacking a system, he would use the denoted positions as a starting point. Relations marked with with empty circles (○) denote a low, but not negligible probability.

However, future research in at least the following areas is necessary:

- Building a formal model based on the classification as given in Section 5.2.
- Empirical evaluation of occurrence of the single classes of attackers.
- Investigate relation to attacker models from other fields and subareas in IT security. To which end can these different models be merged or mutually benefit from each other?

In the next section we will again switch the focus and comprehensively discuss attacks on anonymizing systems.

	3rd Party	Peer	Local Admin.	ISP	Govern-ment	Secret Service
Node Operator		○	●	●	●	●
ISP			●	●		
Users	●	●	○		●	●
Service Provider		●		●	●	
Law Enforce-ment	●	●			●	●

Table 5.1: Relation between entities as discussed in setup (Section 2.4) and classes of attackers (Section 5.2).

Chapter 6

Attacks on Anonymizing Networks

After we introduced basic principles of anonymizing networks, discussed their deployment, the involved entities, and their attackers, we discuss attacks on anonymizing networks. To this end, we give a thorough overview of manifold ways to compromise the security of privacy protecting mechanisms. The task of this chapter is to show limitations and constraints of anonymizing networks. We target an evaluation which points out the most vulnerable points of these networks.

Even though a plethora of security issues are known in the area of IT security we will only be as verbose as necessary in order to highlight important relations between well-known weaknesses and information leaks in our area of research. However, in the core area of anonymous communication we strive to be as complete as possible, covering all major relevant attacks, known vulnerabilities and, if viable, inherent limitations.

As large-scale networks have only been deployed since recently, their properties could not have been studied before. This resulted in a number of surprisingly easy attacks, which undermined their security. This showed that it is inevitable to enlarge the horizon from the traditional point of view of anonymity researchers, which is focused on the network layer, especially routing functionality. The current situation is best described with the need for *holistic anonymity*, i.e. future solutions to provide anonymity will have to target *all* layers of communication protocols in order to achieve any kind of reasonable protection.

In the next section we will briefly recapitulate the flow of information through an anonymizing network. Then, we discuss vulnerabilities and attacks on messages in the different parts of this flow. Special attention will be given to attacks on the networks layer, as this area has undergone the most intensive research. We conclude this chapter by building an *attack tree* and summarizing the findings.

6.1 Flow of an anonymized message

To prepare the view for holistic anonymity problems it is inevitable to have a detailed understanding of the flow of a message through an anonymizing network. We use the details provided in this chapter to explain recent findings on anonymizing networks' failures.

As we laid down in Section 2.1 on page 9 about computer networks a message has to pass through multiple layers in an anonymizing network in order to reach its destiny. It is also a peculiarity of our topic that messages traverse protocol stacks multiple times on different hosts. Making this flow explicit is a crucial precondition in order to understand at which points attacks influence or tamper with this flow. An illustration of this flow is found in Figure 6.1.

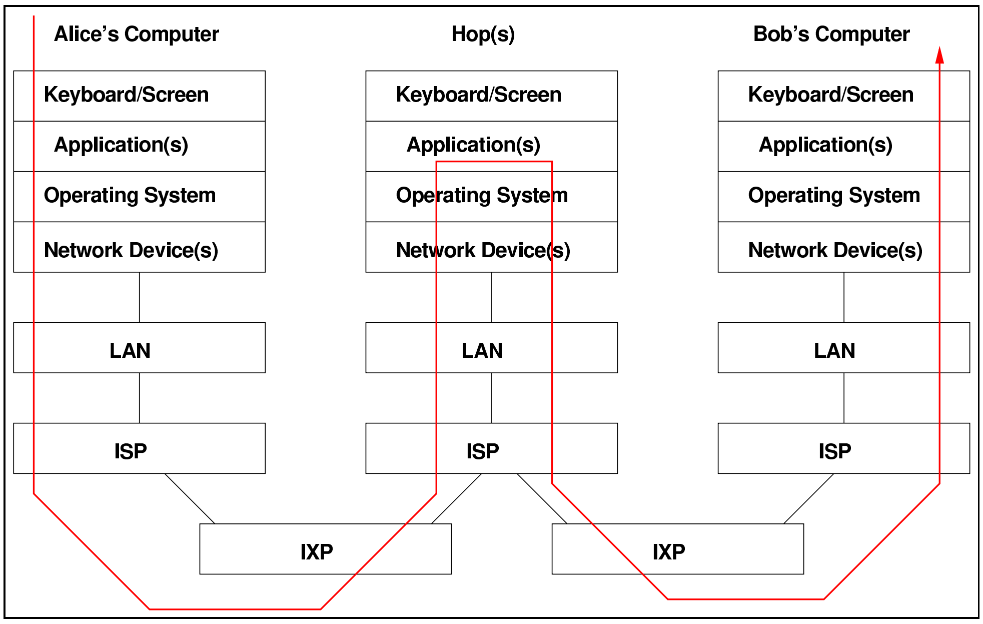

Figure 6.1: Schematic flow of an anonymized message

Even though the major deployed systems for anonymity use circuit switching, in the end there will be IP packets transmitted from one host to another. Because of that we will lift some assumptions and only consider the course of a single message whenever it makes no reasonable difference to do so. However, the reader should keep in mind that depending on transmitted content there might be number of transmitted data packets, possibly within a short time frame and along the same path.

User's Computer A message is generated on a computer or similar device[1]. This usually happens by a user typing in a message on his keyboard; other possibilities include clicking an URL. While there are messages also sent by background processes without user interaction, those processes also received input from an user at a prior point in time, e.g., a configuration file.

The majority of messages are HTTP-requests, file sharing and e-mail [iG07, MBG$^+$08].

If the user participates in a multi-hop anonymizing network (see Section 3.1.2 on page 43) a series of intermediary hops to relay the message needs to be chosen. This can either be done by the user in a manual fashion or is carried out by his software on his behalf.

Depending on the chosen anonymizing network, the original message is then encoded, transformed, encrypted and otherwise processed by the machine's software and hardware. The algorithms are possibly also determined by the chosen message type.

Finally, the resulting message leaves the machine over a network interface for further processing by other machines.

Network Depending on the kind of network the originating machine is connected to and the next chosen hop, the encoded message will traverse multiple physical networks. This might be a local WiFi network, or an ethernet LAN to the next gateway connected to the Internet. Other possibilities include GSM, DSL, modems, fiber cables and satellite connections. Whatever technology is used, the message is very likely to be passed into an ISP's backbone network at some point in time.

If the next recipient is located at a different ISP, the message will consequently also pass some IXP[2], and/or NAPs[3]. These form central switches where ISPs interconnect their networks.

From these places onwards, it is routed along a similar topology in order to reach the specified host for which it is addressed.

Hops and Relays Most anonymizing networks make it mandatory to forward a message over one or more other nodes before passing it to its final destination – or at least the purpose of an anonymizing network suggests to do so. At each of these nodes the message is transformed, decrypted, re-encoded and possibly re-encrypted, much like the processing done at the originator. Also, depending on the network, the message may be stored and withheld before being forwarded for some amount of time. Only after this handling took place, the message is send to the next node or the final destination.

[1]In the remainder of this work we will not distinguish between servers, personal computers, mobile phones, PDA, etc.

[2]Internet Exchange Points.

[3]Network Access Points.

Some deprecated network protocols, like Crowds, require the intermediate nodes to assist the original sender by choosing the next hop themselves[4]. While this has a number of security implications, none of the currently deployed systems requires this. One reason for this is that this feature required users to place more trust in the forwarding nodes – however, this should be reduced to the most possible minimum extent.

One of the most important differences of the forwarding nodes in contrast to the original sender is that these nodes do their work without any human intervention, i.e. all work is done by fully automated processes. Depending on the message and protocol in use this might apply to the receiving end, too; e.g., HTTP messages are usually handled by a server process, whereas e-mails are typically targeted to be read by another human.

Network Depending on the anonymizing network and the sender's preferences, the last two steps might be iterated a number of times. Depending on the network in question, the number might be deterministically chosen by the user or the result of a process involving random variables.

Depending on the type of message, i.e. if the targeted entity is inside the anonymizing network, or not, the last intermediary node has to pass the message to the final recipient outside of the network.

Recipient's Computer Arriving on the recipient's device, the message is processed a final time by hardware and software before it is presented to either the user, stored on the disk, printed out or thrown away.

In case the message was received by a server process, the message will be reacted upon in an automated fashion.

If the anonymity network and the type of message allow to return a reply, a similar chain of events will take place in order to transport the reply back to the originator.

On each of the stations above which a message is traversing can involve multiple sub-steps.

In the following sections we describe in detail vulnerabilities and weaknesses of anonymizing networks, i.e. in which ways an attacker can compromise an anonymizing system and gain information about its users.

The only widely known academic work presenting a similar point of view, but on a much smaller scale and less structured, was presented in [CDK01]. In contrast to this work, we take a structured approach on listing and discussing publicly known attacks.

[4]Other networks allow nodes to add detours for messages. However, these still put the messages back on their original track after a certain amount of time.

To this end, we will start on the lowest layer, seen from a communication theoretic point of view: the physical hardware (Section 6.2). From there, we discuss issues of operating systems (Section 6.3), which is followed by an examination of problems of application layer software in Section 6.4. We continue with attacks on the network layer (Section 6.5) and conclude with non-technical attacks (Section 6.6) and theoretical results (Section 6.7). As an outcome of the discourse, we will be able to build an *attack tree* in Section 7.1 on page 117.

6.2 Attacks on Hardware

Anonymizing networks are built as overlay networks. Accordingly, they require some kind of software, i.e. an application, for their operation. This application runs on top of an operating system which in turn is run on *hardware*. In order to ensure a secure and trustworthy application the operating system as well as the hardware have to be secure and trustworthy, too. This is called the *chain of trust*.

To state it as a general problem: it is in most circumstances very hard, if not impossible, for an application to determine if one of the lower layers, like the operating system, are compromised by an attacker. This is especially true for applications with restricted privileges which do not possess the capabilities which would be necessary to delve into the lower layer's depths. The extent of this problems difficulty can be illustrated by this example: it is even hard for a human administrator with full system privileges to determine if a system is compromised by an attacker, hence it is much harder for a piece of software. Another point underlining this is that there is its own discipline of computer science, namely *digital forensics*, dealing with these kind of problems.

Therefore we can conclude that if an attacker gains access to the lowest layer, i.e. the hardware, he has access to a very powerful attack vector on a victim's computer.

Fortunately, this kind of attack requires *physical access* to the hardware in most cases and is thus rather difficult to conduct, if the hardware is either safely stored or located at a remote place. Another problem for attackers mounting attacks on hardware is that it is usually impossible to automate the attack. That is, unless the attacker produces the hardware and builds in deliberate back doors or sells computer systems on a large scale[5].

6.2.1 Attacks

Attacks on hardware traditionally include reading the victim's screen either by *shoulder surfing* or utilizing a hidden camera. These techniques essentially provide

[5]Compare e.g. `http://online.wsj.com/article/SB122366999999723871.html`

an adversary with a good overview of a victim's actions. It is also possible to install microphones to listen into (otherwise encrypted) Voice over IP conversations.

The victim's input can be grabbed with hardware key loggers; these are very small devices which are placed between the keyboard and the computer case and can record up to two gigabytes of keystroke data[6]. There are versions which can be soldered into existing keyboards such that there is no physical or electronic presence detectable[7]. Hardware key loggers give an attacker the opportunity to gain access to a user's passwords and arbitrary complex key phrases.

Other methods include extracting information directly from the memory of running or suspended computers [PP07b, HSH$^+$08], or reading fragments of data from a hard disk [FV04].

Of course, full access to network equipment gives a wide range of opportunities. These will, however, be discussed in Section 6.5 on page 106.

6.2.2 Discussion

These attacks imply high requirements on the attacker's resources, especially on his physical presence, but also on his budget, as they include the available of specialised equipment. On the other hand, they are very powerful. There is virtually no defense against a well-placed hidden camera reading the screen of a user or a keyboard recording all keystrokes.

Also, analysing hard drives has become one of the major information sources for law enforcement agencies to prove that a suspected individual has committed a certain action. Hence, if an *anonymity set* is getting small enough and its users can be enumerated, it might be feasible for some adversaries to capture the devices of all remaining participants. These seizures makes it possible to analyse hard disks and search for digital evidence.

It should be noted that access to hardware cannot only be gained by means of official search warrants, but also with *burglary* or *social engineering*. One possibility for an attacker is to use some malicious software to tamper with an unwitting end-user's computer and then offer to repair it. As he inflicted the damage himself it is trivial to removing the problem's cause and the access can be used to plant bugs into the respective PC.

Attacking laptops is even easier, as they might be accessible outside a victim's house. Depending on the hardware modification to be made it is enough to have less than a minute of unattended time with the target's hardware [Fin06].

In recent time, attacks on hardware have become even easier. This has two main reasons: one is that hardware is increasingly becoming more and more "multiple purpose hardware" with software loadable firmware. With this mechanism

[6]These are devices which are available for commercial purpose (about US$80).

[7]About US$20, soldering equipment is extra.

software is able to control the behaviour of hardware. One example of creating a malicious back door in the firmware of a network card is demonstrated in [HM04]. The same authors also discuss ways to inject malware into a computer's BIOS. In the latter case the malware would be executed before any operating system is started.

The second reason bears even more potential: hardware is becoming more complex, i.e. is capable of more functionality than before. With this additional complexity comes also danger, as it introduces new security risks[8]. This danger comes from faulty hardware, i.e. one in which the security mechanisms are not implemented correctly; for example, some versions of recent Intel processors included bugs which allowed access to arbitrary memory regions for any software [dR07, Cor07]. These kind of faults allow malicious software to circumvent any protection provided by operating systems which thrives to prevent that application software tampers with other processes or the operating system itself.

Another danger arises from advanced hardware features, like hardware layer virtualization. This can be used for transparent malware to operate on a computer below the operating system, separated by a layer of hardware virtualization [KCmW$^+$06, Rut07]. Thus, it is not possible to detect this malware by conventional means.

Finally, it is possible to use special hardware characteristics in order to learn information over side channel attacks. One example is the *Hot or Not*-attack by Steven Murdoch [Mur06]. In this attack Murdoch uses the fact that a heavily loaded computer system produces more heat than an idle system. The temperature rise then influences the internal clock of the computer in a way that can be detected from a remote computer. In [Mur06] it is shown how this effect can be used to identify users of anonymizing systems.

We can conclude that once an adversary has gained access to a user's hardware, the attacker has a plethora of powerful and stealthy methods to get all information he is interested in.

6.3 Attacks on Operating Systems

As discussed in the previous chapter on attacks on hardware it is necessary for a secure system to rest on solid pillars. In addition to hardware this specifically refers also to the operating system of the computer that runs the software.

Modern operating systems make use of *compartmentalization* in order to protect themselves and other processes from malicious applications. Today's hardware provides operational modes which support different levels of privileges[9] in order to enforce these protection schemes. The typical goal of attacking an operating

[8]"Complexity kills security".

[9]This protection provided by the hardware may fail, see Section 6.2, page 97.

system is to break through this protection and manipulate the functionality of the operating system.

There are basically two methods for attacking the operating system: the most common way is to first gain unprivileged access to a computer. This can be achieved by means of, e.g., a faulty user-land application. Even if the adversary targets a specific application on a computer, e.g., the software running the protocol for the anonymizing network, he might take advantage of any other software running on the same computer. Typical examples are mail user agents, browsers, instant messengers or web servers with interactive content. Once the attacker has succeeded in compromising one of those he can attack the operating system as a local user of the system. From this point on, an attacker can use *local* faults in the system to breach kernel security. This act is called a *local privilege escalation*. Once succeeded, the attacker is in control of a layer “below” the targeted application and hence has arbitrary control over it.

Attacks utilizing *trojan horses* to gain (unprivileged) access to a victim’s computer are technically related. Once the software is installed on the target computer it provides the attacker with some access to the victim’s computer. Typical ways of installing trojan horses include social-engineering techniques or peer pressure[10].

The second way of compromising an operating system is by using flaws in its implementation that interface directly to the network. Prominent examples are errors in implementations of network stacks, like the TCP-stack. However the security of network interfaces has been vastly improved in the last years and hence these attacks became rather unlikely.

With access to the operating system, an attacker has essentially a similar set of opportunities available as if he would have access to the hardware: some possibilities include reading the screen and the keyboard, getting a copy of the network traffic or copying the hard drive and the computer’s memory. The main difference is that an operating system provides the attacker with unified, and thus more convenient, methods to access the data, as well as a ready network stack for remote control and data transfer.

On the downside[11], it is easier to detect a modified or misbehaving operating system, than malicious hardware.

6.3.1 Configuration Errors

Although exploiting configuration errors requires as a unconditional precondition an absent-minded administrator, attackers can still take advantage of these lapses. Despite this, there are still rare conditions, at which an attacker can influence the configuration of some software. For example, the default Tor implementation had

[10]See, e.g. `http://www.skype.com/`

[11]Seen from the attacker’s perspective.

a vulnerability published in late 2007 in which an attacker could overwrite, and hence control, the user's configuration file.

Configuration errors can happen at operating system level, as well as for each individual application. Both have potential impact to an attacker gaining access to a computer.

Typical scenarios include software which is delivered "insecure by default" and relies on the user to configure it correctly before usage. Prominent example are most low-latency networks for anonymity: these require the user to configure his browser to make use of, e.g., a proxy interface *and* turn off all active web content like JavaScript, ActiveX, Java applets, etc. It is frequently reported that unexperienced and inpatient users miss this step for various reasons, or do not want to use the world wide web without active web content.

Another common mischief is the so-called *DNS-leak*: even if a user's software is correctly configured for proxy usage, there are chances that the application bypasses this setting in order to make DNS lookups. These requests can in turn be eavesdropped by a set of people and allow to trace the user's connections despite the actual data being anonymized. To a certain extent, this bug can even be triggered by, e.g., malicious websites, by embedding parts of the websites with the `ftp`-protocol; while all major browsers are capable of loading content over `ftp`, there is a good probability that they will not use a HTTP-Proxy to fetch the content. Consequently data loaded by `ftp` will be fetched with the user's original IP.

As we have seen, these issues have severe impact on the degree of protection. However, it is most often beyond feasibility for an adversary to tamper with the users' configuration, even more, if a *specific* user is targeted. On the other hand, an attacker can be sure that in a bigger network there will always be a certain amount of mis-configured clients.

6.4 Attacks on Software

This section deals with one of the major attack vectors on anonymizing networks: issues regarding implementation details of the actual *software*. Implementing any software introduces a plethora of problems, most of which are known, but there might also be a number of yet unknown problems. Some of these errors lead to unpredictable behaviour, likely to crash the process. Others can be exploited by an attacker to trick the software into doing arbitrary actions, other than those planned by the developer.

For a basic overview on the possibilities which software vulnerabilities offer an adversary, the reader is suggested to consult e.g. [HM04].

This chapter also deals with some attacks on algorithms that are used in anonymizing networks. Algorithms are theoretical constructs which have to be implemented in software to be used in, e.g., real networks. Therefore, there are at least two different ways to attack them. The first is to find an inherent deficiency in them which consequently can be abused in all of their implementations[12]. Or second, it might be possible to find an implementation specific flaw[13].

This section will discuss issues in the following order: we first list generic ways of breaking software and their effects on a system. The ubiquity of these mistakes is displayed with an examination of their relation to anonymity systems.

Given that an attacker cannot find and use these ways of compromising a system, does not have the required capabilities or simply chooses to try different means first, we then focus our view on methods to learn desired information even in absence of generic implementation errors, i.e. by attacking the algorithms itself.

Finally, we describe even more possibilities for an attacker: this includes identification of specific types of software and form a last resort to identify users, even if the algorithm for anonymization or its implementation do not leak sufficient information to succeed with an attack.

The section will be concluded with a small summary.

6.4.1 Breaking Generic Software

Even if algorithms used for anonymizing users' network traffic would be perfectly secure and all users would behave uniform, software errors could still compromise their protection. This holds true since programming errors can be abused by attackers to learn information or even get arbitrary control over other people's computers.

The technical background of software errors was introduced in the terminology section, 2.2.1 on page 15. Their presence rests mainly on the imperfection of human application developers, i.e. missing foresight of specific input and behaviour patterns. This applies to any software in general, hence also to implementations of anonymizing networks.

The ubiquity of software errors makes it today impossible to build a "secure system". One reason is that the complexity of modern software makes provable security impossible. Also, the code base is growly rapidly, hence enlarging the attack vector more and more. Today, the most reasonable approach to ensure a certain baseline of security is to establish a *process* to handle security issues. This refers to a procedure to handle the disclosure of problems, ordering them by priority, fixing them, testing the fixes and enrolling the fixes. These procedures grant advantages over less organized ways of handling security issues.

However, even if there are established processes to deal with security issues, these take time. Response to newly discovered vulnerabilities takes several hours in the best case, sometimes up to multiple weeks. This leaves a large window for successfully attacking computers. In case, an adversary discovers an exploitable

[12]One example would be a Man-in-the-Middle attack on the Diffie-Hellmann algorithm.

[13]See, e.g., the havoc which was caused by the OpenSSL-bug in the Debian distribution in 2008.

Type	Cost
Windows Vista Zero-Day Exploit	\$20,000 – \$30,000
Application Level Exploit	\$4,000 – \$5,000
Custom Trojan (stealthy)	\$1,000 – \$5,000
Custom Malware (generic)	up to \$20,000
Fake identity	\$150 – \$800
Account information	\$7 – \$100

Table 6.1: Overview on the cost of vulnerabilities for arbitrary computers

vulnerability and actively makes use of this knowledge, his attack programs is called *zero day exploit*s. It is widely considered that it is generally impossible to defend against this kind of attacks. Therefore, having access to, or being able to develop zero-day exploits, gives an attacker the opportunity to successfully penetrate nearly arbitrary computers.

There is an abundance of programming errors that can be found in virtually every kind of application and operating system, most of them providing an attacker with the ability to exploit them. Due to the prevalence of programs written in C, the security landscape was dominated by buffer overflow vulnerabilities in diverse forms – on the stack, heap, BSS-, or data segment. Together with other vulnerabilities, like the format string vulnerability, this had led to a situation where applications written in C are not considered trustworthy by a majority of computer scientists.

Even with the advent of protection mechanisms like stack guards, the No-eXecution bit to avoid code execution on a program's stack, and even other programming languages, computers are still vulnerable. A famous quote related to this is [Dul00]:

> [..] So it raises the bar for us all ☺ but [that] just might make writing exploits an interesting business again. [..]

Even if an attacker is unable to develop high-potential exploit by himself, he can still buy them on a black market. A list of generic vulnerabilities and the cost of a ready-to-use exploit on the black market are given in table 6.1 (taken from [Nar06]).

Note that even next-generation programming languages did not change the security landscape. While some of them solved problems that existed in former programming languages, most of them introduced a variety of new vulnerabilities. As these programming languages have inherently more possibilities than C, their faults are consequently also more dangerous and easier to exploit then errors in compiled C-code. For example, a widely known study[14] shows that more than 95% of web

[14]See `http://www.webappsec.org/projects/statistics/`

applications, which are mostly developed in a “higher” programming language, contain errors which can be exploited by an attacker.

While attacking server applications was the main focus in the 1990s and in the beginning of the new millennium, today’s exploits are focused on client-side applications. Thus, with an exploit for, e.g., a web browser (a cost of about $5,000) and a website that is engineered to lure a victim into visiting it, it is possible to execute arbitrary code on the victim’s computer. With this methods an attacker can take over control of the victim’s computer and is able to not only get the victim’s IP address and true identity, but, e.g., also read arbitrary information from the hard drive.

That even high-security systems are vulnerable to attacks was prominently shown in the public media in fall 2007: groups of supposed to be Chinese hackers broke into computers of the German chancellor’s office, the Pentagon and British military systems [Bri07]. However, it should also be noted that penetrating high-security systems is harder than attacking the computer of an average end user. This is of concern to anonymizing networks, as the relaying nodes are most often better protected than end user’s computers, but still more vulnerable than high-end systems.

We can therefore conclude that just with the use of software exploits, an attacker is capable of breaking an anonymizing network. To this end, he must either manage to break into a certain amount of network nodes or to exploit the victim’s machine with the means of, e.g. a bait page. We can also see that an attacker can compensate lacking skills with an adequate amount of money.

6.4.2 Breaking Anonymizing Algorithms

Given that the implementation of an anonymizing network’s algorithms are flawless[15], an attacker can still attack vulnerabilities in the *design* of the algorithms. Hence, he can find *inherent* problems with the deployed algorithms which him to infer supposed-to-be secret information.

In this section we will list several well-known and analysed attacks on anonymization algorithms. These can be used by an attacker in order to gain more information on the whereabouts of his target or to increase his chances of getting these.

End-to-end timing attacks make use of the fact that sometimes the same entity might own the first and the last node of a connection through a circuit-switching anonymizing network. In this case the entity can successfully link the data’s originator with the destination. Despite this weakness being publicly documented on the Tor website, there were a couple of publications that re-discovered it [ØS06] or which proved it to be possible under relaxed conditions, as described in [MD05]. While other networks, e.g., AN.ON, have not publicly been proven to be vulnerable to this attack, it is very likely that these attacks work. However, there are

[15]See the previous section, 6.4.1 on page 100, for a discussion on this topic.

some indications that end-to-end timing attacks are not as dangerous as commonly perceived[16].

One of the most dangerous and also most difficult attacks to defeat is the so-called *Sybil-attack* [Dou02]. In this attack, the adversary mimics the behaviour of a complete anonymizing network, tricking the user into believing that he has just joined a set of other users, each one providing cover traffic – see also [KP06] for a description on cover traffic. Variations of these attack are discussed for different types of anonymizing networks in, e.g., [SDS02].

In order to create sufficient plausible identities an attacker is in need of man power and hardware. A commonplace argument to defeat the Sybil attack is to introduce a centralized check in anonymizing networks, to determine the participants are actual humans to avoid automated multiple logins. On the other hand, as Table 6.1 illustrates, it is no problem for an adversary with enough money to provide a sufficiently large set of fake identities.

In a scenario, where the attacker does not control all machines, but rather a high percentage of some arbitrary machines, he is still able to attack networks. Bauer et al. describe in [BMG$^+$07] a method to use fake routing information for subverting an anonymizing network.

Proposed protection mechanisms against these threats include enhanced routing algorithms which choose nodes in a way to make it very costly for an adversary to control all of them. To this end, the assumption is made that even well-suited attackers only have limited resources of special types, e.g., in IP ranges. Hence paths can be chosen, where the relays are located in vastly different IP ranges.

However, from recent studies on bot nets it costs \$0.05 to \$0.10 per node to rent a bot net [Zel07]. This invalidates the above assumption for attackers that are willing to buy, rent or build a bot net.

But even attackers which do not like to get into touch with bot nets have opportunities to invalidate the above assumption: instead of using a multiplicity of slow nodes, they can buy high-performing nodes in several countries. To rent a decent server in most developed countries ranges from \$100 to \$500 per month; virtual servers can be rented for as less as \$10.

Another powerful attack on anonymizing networks is the *predecessor attack* as described in [WALS04]. It uses the fact that in some routing protocols the last node to forward a message is the actual originator of the message with a higher probability than all other nodes in the network. The attacker can use this fact to accumulate this knowledge over time in order to proof sender-recipient relationships. A detailed discussion on the efficiency of this attack can also be found in [PP06a].

The attacks as described in this section require the attacker to have a reasonable coverage of the network, i.e. he is in need of a certain physical influence. Consequently, attackers fulfilling the requirements are in a very powerful position: if an

[16]`http://archives.seul.org/or/dev/Sep-2008/msg00016.html`

adversary is capable of mounting this kind of attacks there is only a very limited set of defense mechanisms to deploy.

6.4.3 Identifying Software

Given that the implementation of an anonymizing network is secure *and* its algorithms are strong enough to resist an attacker, the possibility remains to identify users or their computers on the application layer. As a method to protect information leak on the network layer does not change application layer data[17], it is possible to *profile* users in accordance to this information.

This means that an adversary who is operating an exit node of a network or one of the user's peers, can try to learn information about the anonymous user by finding characteristics in the user's application layer data. This profile can then be used in order to either *re-identify* the user at another place, or shrink the anonymity set to a size which allows an *identification* of the user in real life.

Profiling on the application layer usually works on characteristics of the data that is sent as additional information by client software. For example, web browsers send sets of data to the web server concerning the user's setup, even though this information is not necessary to process the request. These headers (an example is given in Figure 6.2) contain hints about the operating system of the user, software versions, possibly the language of the user, as well as accepted file types and encodings. They also show which web-site the user visited before, so that an adversary can easily follow the trail of a user through the web.

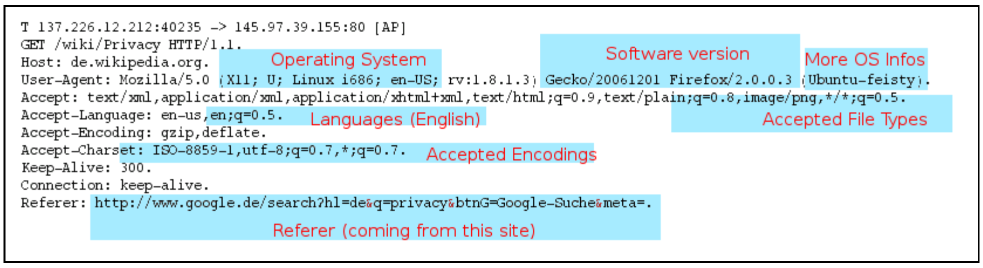
```
T 137.226.12.212:40235 -> 145.97.39.155:80 [AP]
GET /wiki/Privacy HTTP/1.1.
Host: de.wikipedia.org.
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.3) Gecko/20061201 Firefox/2.0.0.3 (Ubuntu-feisty).
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5.
Accept-Language: en-us,en;q=0.5.
Accept-Encoding: gzip,deflate.
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7.
Keep-Alive: 300.
Connection: keep-alive.
Referer: http://www.google.de/search?hl=de&q=privacy&btnG=Google-Suche&meta=.
```

Figure 6.2: An example HTTP-Request with a highlighted set of privacy-related information

While the provided information are rarely unique enough to identify a single person, they allow to cluster users in groups. Statistics over the most prominent field `User-Agent` show an entropy of about 6.5 bit for more than 3 million different user agent strings[18].

[17] Due to adherence to the ISO/OSI-layer or network layer stacks in general.

[18] Data collected for the duration of 15 months on a medium-sized e-Commerce site.

```
<html>
  <head>
    <title>Hallo-Welt-Beispiel</title>
    <STYLE  #header {margin-bottom: 3em;}
      html>body #header {margin-bottom: 1em;}
      body { color: purple; background-color: #d8da3d }
      html>body {color: #d8da3d; background-color: purple;}
    </STYLE>
  </head>
  <body>
    hello
    <font color = "purple"> You are not using Iceweasel </font>
    <font color = "#d8da3d"> Your browser is Iceweasel compatible </font>
  </body>
</html>
```

Figure 6.3: Identification of the user's browser based on its rendering capabilities

However, removing this information does not lead to a situation, where the user is safe from being profiled. Is it trivially possible to profile software by its individual behaviour. The reason for this is that different software acts differently on some kinds of requests, e.g., because it does not implement certain features. An example is shown in Figure 6.3: depending on the user's browser rendering engine only one of the texts is visible. If both texts contained a link, the server would know the user's browser based on which link was clicked.

Typically, this technique makes use of "strange" requests and observes the actual behaviour of the implementation. Amongst other observations, the order of fields and response codes can be used to identify implementations even if obvious information, like user agent strings, are removed or faked.

The most powerful way of analysing a browser is active web content. A good example of information that can be retrieved by active web pages is given at `http://gemal.dk/browserspy/`. This attack vector can be abused to identify users, as can be seen in [ALLP07].

The most privacy invading technology however is Javascript, as was demonstrated with *Spyjax*[19]: a small piece of Javascript code that uses CSS and HTML rendering in an invisible part of the screen. It can be user to determine if the user visited *any* given site recently. Other information to retrieve with Javascript include the user's timezone and the computer's clock skew, i.e. how much its time deviates from the actual time.

Last, but not least, it is possible to identify users based on their writing style, grammar, typos, or similar information (e.g., [ZLCH06]).

[19]`http://www.merchantos.com/makebeta/tools/spyjax/`

Thus, even given the fact that the data streams of an user are anonymized in a perfect way, users will be distinguishable from other each other based on their software characteristics. Therefore, a good technique to hide would be to fake (or better: actually use) the software that is most common in the set of people that comprise the anonymity set.

6.4.4 Summary

In this section we covered the possibilities for an adversary to learn information about a user by means of abusing software. We described three main fields: generic software faults, breaking anonymization specific algorithms and identifying users based on their software.

6.5 Network-based Attacks

Attacks on the network layer adhere to the established focus on anonymity researchers. For this reason there is an extensive list of works in the area of attacks on the network layer, mostly focusing on information that can be gained by third parties observing messages.

A small survey of traditional network-based attacks in this area is given by Wright in [WALS02]. However, this work is somewhat outdated and a number of very powerful attacks have been found and published after 2002. With the established attacks discussed first we prepare the ground for an overview on the more advanced attacks.

The attacks we present in this section are listed into increasing order of their required preconditions. Hence we first start with a set of rather simple, but possibly highly efficient denial-of-service attacks. These can usually be mounted even by remote attackers, not involved in running or participating with the attacked network at all.

The number of possible attacks grows with an increasing physical influence of the attacker. Hence we then discuss locally bounded attackers and those which are close to the victim's peer or the peer itself. We then continue with ISP-scale attackers and finally cover attacks which are feasible for a global adversary.

6.5.1 Denial of Service

The objective of *denial of service* (DoS) attacks in our context is to make it as hard as possible for victims to use the anonymizing network; if this succeeds, the victims have to either stop communicating or use plain means of communication which will reveal their peers. Basic methods to achieve a shutdown of the service include shutting down the network or preventing the users from connecting to it in the first place.

There are several ways of setting up denial-of-service attacks. Any user in the Internet can try to shut down a network using central infrastructure by making the servers which are obligatory to connect to inaccessible. Possible attack vectors of the network can be, for example, a central directory service which lists the addresses of the nodes in the network. Another vector are the nodes themselves, in case the attacker can cope with their number.

“Taking down” can be achieved by rather unsophisticated means as bandwidth exhaustion. This means that the attacker sends a high volume of requests to the targeted computers, such that they are too busy to answer legitimate requests. In an extreme case, the network link to the attacked computer would be utilized in such a high manner that no legitimate packets would even reach the attacked computer. Other possibilities include attacks on the routing protocols which change the path of messages such that messages to the target are routed into an empty space, or even redirecting DNS requests by spoofing or poisoning. The biggest problem with these attacks is that it is very difficult, if not impossible, to protect against them. Any networks which rely on centralized infrastructure are highly vulnerable to this kind of attack.

Real-life examples of this attack include attempts by the Chinese government to restrict access to the Tor network and attempts from the Iranian government to stop accesses to AN.ON.

For an local administrator or ISP there are more elegant, but also more trivial means of blocking access to anonymizing networks. In addition to blocking access to certain IPs and (TCP-)ports, an administrator also has the opportunity to inspect the content of a victim’s data streams and decide to redirect or drop connections that look suspicious or similar to a network protocol that is being used for anonymization. Attacks of this type can ultimately only be thwarted by steganography.

ISPs or hosting servers can also cause problems for networks by cutting the lines of nodes. An easy way to detect anonymizing nodes is to take a look at the amount of traffic: participants, but especially also relaying nodes, have a high usage of bandwidth, which is often equally distributed into upstream and downstream. This characteristic is easy to spot and very distinctive. An alternative method is to forbid the use of these services by legal means, i.e. in the contract with the clients. For this see also Appendix A.3 on page 148.

Finally, governments are capable to outrightly make the use of anonymous communication illegal. This has a direct effect on users and operators within the given legislation. The only remaining possibilities would be the use of infrastructures which are not primarily related with anonymity, e.g., open proxies and open relays. Also, the use of other people’s unprotected WiFi access points can be of use in this case.

Finally, governments can try a "*FUD*" tactic, i.e. spread *fear*, *uncertainty* and *doubt* along the lines of privacy enthusiasts in order to break the willingness of contributing nodes to an anonymizing network. Possible actions in this area include confiscating nodes of the network or filing lawsuits against their operators to reduce the size of the targeted anonymization network. However, as long as there is a significant amount of nodes outside the legislation of nations practising this, the effect of these techniques is negligible.

6.5.2 Local Area Networks, and Local Attackers

Anyone between the user and the first hop of the network, like the user's local administrator, can try to identify the type of traffic anonymized by the user with the help of statistical traffic pattern analysis. This has been shown to work for web traffic, to the extent that attackers can identify certain webpages accessed by the user due to characteristics in volume of and the delay between single data packets [Ray00, Hin02].

Follow-up research was conducted by Serjantov and Sewell who analysed the general properties of hiding connections in anonymizing networks [SS03]. They identified a set of preconditions under which packet counting attacks were feasible for an attacker and could be used to identify individual connections. They also discussed briefly countermeasures which, however, are impractical to deploy in large-scale systems.

A new type of attack has been presented by Murdoch and Danezis in [MD05]: A remote attacker can send probing messages through the anonymization network in order to gain information about the path of an victim's message through the network. With the help of this attack, which is nearly transparent to the victim, the attacker can trace him down to the first node of the network. If the victim is participating as a node himself, it is even possible to identify the user. This attack reduces the protection of low-latency networks like Tor, which should provide a rather high practical level of security, to the protection provided by a single proxy hop.

While this attack on its own can only rarely be used to identify end users, it was significantly enhanced in [HVCT07]. The authors show how to accumulate knowledge about the *network latency* between an user and several nodes in the network, as well as how this can be used to clearly identify the physical location of a user.

End-to-end traffic confirmation attacks have been used in [ØS06] to identify IP addresses and identities of location-hidden servers in the Tor network. To this end, an attacker builds several connections to the hidden service, which in return has to start building up virtual circuits through the Tor network. This behaviour is due to the specifications of the Tor protocol for hidden services. By deploying as little as a single node as a relay in the network, the attacker has a certain probability for each of this connections to be chosen as first hop in one of the hidden services' paths, thus learning the true identity of the server providing the hidden service.

6.5.3 Exit Nodes and Peers

Most of the users of an anonymizing network hide their identity from their peer partners while communicating with network protocols like HTTP and e-mail. Even though traffic is encrypted *within* the anonymizing network, it is in often without encryption when it leaves it. This gives the opportunity for the end node operator to read or change all data exiting from the network or being returned as a response to one of the anonymous requests[20].

The reason for this attack to work is that the security model of anonymizing networks only covers routing information. This means that the protocols are designed to keep the relaying nodes from learning the complete path through the network. If the protocol is designed correctly the nodes may be partially untrusted by the user and hence there is no problem, if one node is operated by an untrusted party.

However, the actual content of the message is not encrypted per se. This is not a problem in networks which only transmit messages within themselves, like Freenet. But there are networks, most notably Tor and I2P, which offer access to arbitrary webpages. Here, the protocols of the network itself are insufficient to provide protection because they do not offer protection of the content. While unprotected HTTP can be considered a minor security issue if only professional ISPs are forwarding packets between the two entities, anonymizing networks offer *anyone* the opportunity to forward messages on behalf of others and hence also record their content.

This means that any exit node which might be operated by an untrusted entity, is in a position to eavesdrop personal data from users[21]. The attacker can use this data in order to profile users and commit identity theft or impersonation attacks. This refers not only to personal data deliberately entered by the user: name, address information, his language, etc, but also includes data contained in the header fields of communication protocols (also described in Section 6.4.3).

Given that an attacker does not only resort to passive techniques, he can increase the odds on identifying a victim. An exit node, but also a peer partner, can inject active content into data in order to force the victim's software to bypass the anonymizing network. Practical instances of this class of attacks have been shown to work in, e.g., [For06, GAL$^+$07]. The results were devastating: the success rate was far beyond 90%, allowing to identify an significant amount of the network's

[20] As a side remark it should be noted that there is a share (of unknown size) of users who perceive that anonymizing networks "magically" encrypt and secure all of their communication.

[21] Under rare conditions it is possible to detect attackers logging data [MBG$^+$08]. However, if properly mounted this attack cannot be detected.

users. Encryption of content does avoid that a malicious exit node tampers with the traffic, but a peer partner will still be able to conduct these attacks.

Even if the content is encrypted exit nodes are still able to to tamper with the connections to a certain extent: if the cryptography used is too strong to be broken, the adversary can simply block all encrypted connections, effectively resulting in an denial of service attack. Another possibility includes an active man-in-the-middle attack on the secured connection. While this relies on the user to ignore the warning of a strange SSL-certificate, in fact, studies found that more than 90% of users does so [SDOF07].

6.5.4 ISP-Scale Networks

ISPs have a significant amount of network connections under their control, multinational ISPs may even control a significant part of the complete Internet. This rises significantly the potential of an ISP to prohibit a user from accessing an anonymizing network by means of denial of service attacks.

In addition, it has been shown by Murdoch in [MZ07] that controlling a single central hub of the Internet is sufficient to deploy timing and correlation attacks. This even holds true, if the attacker is able to intercept only a fraction of all traffic running through the hub, like one out of 10,000 data packets.

6.5.5 Wide Area Networks and Global Attacker

For reasons described in Section 5.1 on page 73, the traditional focus of researchers in our area has been an adversary which is able to observe all communication lines in a given network. Therefore, the majority of (older) works on vulnerabilities of anonymizing networks focuses on this view.

Some publications make the simplifying assumption that an attacker cannot see all messages passing between nodes of the network. Hence, in some of these papers the network itself is modelled as a single entity processing messages. The input to this entity consists of messages generated by the users, while the output consists of the users' message to their peers.

Intersection attacks form a very powerful class of attacks in these scenarios. They are capable of extracting information about users based on repeated communication with a similar pattern of behaviour. By exploiting traffic patterns, statistical characteristics, or similar features, it is possible to break even systems which provide perfect protection in each single round.

Seemingly unobtrusive differences like the number of hops which are used to forward messages have been shown to leak enough information for adversaries [BPS00]. Other methods known to work include the analysis of the view which a user has on a given network [GKK05]. If users know different sets of nodes, for example because the directory service of a network only releases a part of the nodes rather than all of them, this can be used to partition the users into smaller anonymity sets.

Kesdogan discovered an attack which identified a user's peers by making use of a distinctive feature of mixes: each output batch contains only a single message from each user [KAP02]. This information allows an attacker to extract the peer partners using a two-staged algorithm: the *disclosure attack*. Mitigation of this attack is very difficult because the basis for each user sending only a single message per batch is an actual security feature to prevent Sybil attacks (see Section 6.4.2).

While the complexity of the disclosure attack was NP-complete, Danezis proposed a probabilistic version of it that possesses polynomial run time [Dan03]. Due to the nature of being a statistical attack, the result could be wrong with a certain probability, based on the amount of observations available to the adversary.

This attack was taken up by Mathewson and Dingledine in [MD04] and considerably improved. Besides relaxing the preconditions of the attack and therefore broadening the scope of applicability, they also showed simulations of this attack in real systems and also in mix systems with other than fixed-sized batch mixes.

The next step was the development of the *hitting set attack* [KP04]. Similar to the previous intersection attacks, this work made use of intrinsic properties of a mix, i.e. the fact that each batch has at least one contribution per participating user. This allowed an even further relaxation of preconditions for intersection attacks. Another major advancement is that an attacker now does not necessarily need to know the number of a victim's peer partners in advance. Also, this attack works, if there are several contributions of a user to a batch. Compared to the disclosure attack, the run-time behaviour was also improved: with the help of an oracle that guessed the peer partners using the available information in polynomial time, it is now possible to check the validity of the guess in $O(n^{\log n})$; therefore, delivering results with less than exponential time effort.

Other attacks on mix networks use signal-detection techniques in order to link incoming and outgoing streams. Danezis showed in [Dan04] how to apply one instance of this attack on continuous time mixes. A side result of this work included the proof that they were actually providing an optimum level of anonymity in the given scenario. There were also other improved results in the same year presented in [DS04].

In contrast to the works listed above, [LRWW04] considered an attacker which would not only record passively messages, but also delay certain messages. The resulting attack could be used in low-latency anonymizing system in order to gain information about specific connections.

Whereas most attacks had been developed and evaluated on simulated users, results on real data were presented in [KPK05]. The applicability of two intersection attacks was shown on data collected from real users. It showed that users actually acted quite diverse and the success of the attack largely depends on the type of user. [KPK05] also provided an empirical classification of users into several groups, ranging from casual surfers to power users and even automated scripts.

The latest improvements on intersection attacks extended the model of mix networks. While formerly only uni-directional communication was considered, the works [DDT07, Pim07] analysed the effects of bi-directional communication. The results were yet another improvement on the speed and time complexity of attacks. Specifically [Pim07], which itself is computationally cheap, can also serve as a pre-processing step for more complex attacks which would not be feasible otherwise.

Summing up the various options of all possible attacks for global adversaries, i.e. timing attacks, intersection attacks, and pattern matching, it can be concluded that it tends to be very difficult, if not impossible, to defend against an attacker who can observe large parts of a network. In fact, the Tor project deliberately names the global attacker as one of the adversaries which they do not try to cope with.

6.6 Non-technical attacks

This section covers a short list of attacks on anonymizing networks that do not require an attacker to learn technical details of the system, or to keep the amount of technical parts in the attack to an absolute minimum.

The main reason for including this section is twofold: besides being an eye-opener for real-world vulnerabilities there are sometimes technical means that can be used in order to mitigate or at least reduce the impact of some of these attacks.

One of the major threats in general IT security is the fact that virtually all systems can be accessed by and are controlled by human beings. Therefore, manipulating the humans which control the machines is sometimes the easier way to get access to the secrets kept in the computer. This can be done in several ways, depending on the effort, time and resources available to an attacker.

While non-technical attacks are usually of very high potential, they can only rarely be automated by an attacker like a computer attack can be automated. Thereby the cost of this attacks, especially the cost of repeated or multiple non-technical attacks, is quite high compared to technical attacks.

6.6.1 Physical Attacks

Getting physical access to the computers has been discussed in Section 6.2. Similar powerful attacks can be conducted, if the adversary gets physical hold of some computer's administrator.

A ruthless attacker is capable of using physical force in order to coerce an administrator to provide him access to the machine. Possibilities range from slight psychological force on one side to delivering arbitrary amounts of physical harm in the extreme case. A famous example of excessive force happened in March 2005, when in Malaysia thieves not only stole a car, but also cut off the owner's finger as well, since the car was protected by a biometric system[22].

On top of physical treatment, practices like blackmailing, taking hostages, any other way of extortion (e.g., if the target person owes a lot of money, has uncommon diseases, or is addicted to anything) could be used in order to put administrators under pressure.

In our context this attack is usually directed at the operators of the network's nodes or central infrastructure. The more power and influence a single person has to the network, like operating the directory service or a significant amount of nodes, the more an attacker might be tempted to attack this person. If the attacker knows one of the communication's peer partners, then this attack can also be applied to this person in order to learn the identity of the second.

Mitigation of these attack is easily possible for node operators, i.e. staff which relays messages on behalf of others. With dedicated hardware and a special setup it is possible to run anonymization software on a node such that it is autonomous, keeps no logs and destroys all material and data if the computer is shut down or disturbed. However, this is only possible, if the software of the network does not require signed certificates or access to the private part of a public/private key pair.

Peer partners that know (parts of) the true identity of a possible victim cannot easily be protected by means of IT technology and have to rely on plausible deniability instead.

6.6.2 Legal Attacks

Another way for an attacker to force node operators or end users to hand over information is to force them to comply by legal force[23].

The easiest way to learn a piece of desired information which is not accessible due to a certain technical protection is to ban the use of this technique. This also applies to anonymizing systems. The effect is that all law-abiding users within the legislation have to either communicate without such a system or face the potential consequences. This is similar to a denial-of-service attack on the system (see Section 6.5.1 on page 106), but technically allows people to continue usage in face of possible consequences.

An alternative for governments to control the use of anonymizing systems is *key escrow*, i.e. a set of laws to force users to decrypt enciphered data streams

[22] `http://news.bbc.co.uk/2/hi/asia-pacific/4396831.stm`

[23] There might be countries, where the actions described in the previous Section 6.6.1 on the preceding page can be considered legal means for secret services or the police. See `http://www.amnesty.org/` or `http://www.hrw.org/` for further information.

on request of law enforcement agencies[24]. Since anonymizing networks heavily rely on cryptographic techniques in order to achieve their security properties, the protocol is getting transparent for an attacker who is able to decrypt the messages.

However, practical limitations of this include that depending on the implementation of the network, relaying node might not log any data which can be used to trace connections back. Therefore the attacker, in this case a government, would have to store the content of suspected anonymous communication streams himself. A technical measure against decryption of stored traffic data is the use of encryption algorithms with perfect forward secrecy (see Section 2.2.3 on page 21 for an explanation). This is a feature which is supported by, e.g., Tor, but not AN.ON.

On the other hand, in order to avoid "problems" with perfect forward secrecy, some countries like, e.g., France and the United Kingdom [Koo08], introduced penalties for users which are not capable of decrypting data streams at a later time. Therefore, it might not be desirable in these countries to relay data which is subject to perfect forward secrecy.

A third way for an attacker with access to legislation is to implement a country wide interception of data traffic, log-file surveillance, or similar techniques. While these methods are usually not useful for instant identification of anonymized communication streams, they can be a suitable help to gain more insight into the network if any other data is leaked. A prominent example of this approach is the European directive on data retention [Eur06] and especially its German implementation.

6.6.3 Social Engineering

Finally, attacks that manipulate human behaviour without the victim recognizing this as an attack are the most powerful version of non-technical attacks. This class includes so-called *social engineering*. The potential of social engineering was shown and documented by, e.g., Kevin Mitnick [MS05].

In the course of an social-engineering attack the attacker pretends to be a different person or to have different motives than his actual ones. The goal of this attack is to trick administrators or users into arbitrary actions which support the goal of the attacker.

The range of possible actions which can be achieved start with users revealing passwords and access codes. This is commonly pulled off by the attacker pretending to be a legitimate system operator or official person. Faked surveys have also been proving to be an excellent method [ORBO04].

Another option is to ask users to install a certain piece of software which was modified by the attacker and thus gives him access to the users data. For example,

[24] Note that there might be legal inferences due to the right of not having to self-incriminate oneself.

the user can be told that this particular software is a better instant messenger, or a quality-enhanced version of another program the user has installed.

Even experienced, alert and security-aware users might still unintentionally reveal single pieces of critical information. This includes IP addresses, personal information, telephone numbers, or the like. With the help of this, an attacker can either identify the weakest point in a defense system, or he can impersonate that person and attack other persons. Commonly, information acquired by social engineering is also used to ease brute forcing attacks on passwords or encryption keys.

Tracing and defending against social-engineering attacks is very difficult, due to several factors: obviously, the attacked persons often do not recognise the attack in the first place. As has been shown in Table 6.1, the cost for a fake identity is also marginal. Thus even if an adversary has to enter buildings in order to enhance the effect of his attack, he is able to impersonate arbitrary fake identities. Hence, once he left the physical premises, there might be no more trace of the attack left.

The high potential of social-engineering attacks is also underlined by the following fact: even in professional penetration tests it is easy to get information with this kind of attacks. However, the working morale of the staff will be destroyed, if the results of the penetration test will be published within the company. Even though anyone could have been the target, the single person which was successfully exploited is socially isolated. Due to this, the Federal German Office for Security in Information System (BSI) recommends *not* to use social-engineering practices in penetration testing [Bun06], with the sole exception for national high-security areas.

6.7 Theoretical Results on Attacks

Despite this work's focus on real and deployed networks, we discuss a selected set of theoretical results on attacks in this section. The presented set was selected based on their significance for either confidentiality as a general concept or anonymizing systems in special. One difference between these works and others attacks on anonymizing networks is that these works *measure* the basic feasibility of attacks, regardless of the involved system and algorithms. Hence, their results can be used to estimate firm lower bounds on the security of systems.

Shannon analysed the theoretical feasibility of attacks on cryptographic systems in [Sha49]. He showed that there is a bound, the so-called *unicity distance*. All messages which are shorter than or equal to this length and enciphered with a secret key, are safe from being attacked. In this seminal work he uses information theoretic measures to prove the security of encryption algorithms.

A similar idea was taken up in [KP05], where a lower bound has been shown to exist for mix networks. This bound, which consequently was also referred to as “unicity distance”, was compared to existing attacks on the same abstraction layer. The results showed that it had an adequate behaviour for a lower bound. The results also proved that for users with a fixed communication profile there is no perfect protection possible by a mix network: at some point in time the profile can eventually be extracted from pure passive observation.

These results were backed up in [KAPR06], where it was shown with a different method that a lower bound for security exists and that it can be used to make a couple of security-related statements about mix networks.

Finally, a set of open theoretical problems to be solved can be found in [Ray00].

6.8 Summary

We have seen in this chapter that there is a multiplicity of attacks on anonymizing networks. In addition to the attacks on the network layer which are discussed in the traditional literature on anonymizing networks, we have listed attacks on hardware, the operating system, software and non-technical attacks.

It is interesting to notice that a couple of these attacks could have be mitigated or the impact could have been reduced by proper design of the anonymizing network.

Future research in this area includes at least:

- Of course, defense mechanisms against the most serious attacks would need to be found.

  Is there anything which can be done to avoid the severe impact of faults in “lower layers” of a system? This includes hardware, the operating system and software issues.

  To which end is it actually possible to limit the abuse potential of active web content?

- Identification of even more attacks on deployed systems would also help to design better systems in the future. It is likely that more places leak information.

  Especially the application layer has yet not undergone extensive research and it is unclear to which extent the plethora of individual software is compromising network layer protection.

  Also, local and partial attackers have not received extensive attention yet. Fingerprinting websites is probably only one method to extract information.

- Users seem to be one of the weakest part in the system for two reasons: first, they are highly susceptible to be influenced; but also due to the typical repeating patterns of their communication.

In order to make it more difficult for attackers system designers would need to know typical schemes of user behaviour and find methods to protect them.

- Security properties are not additive in the general case, but maybe attacks are. Is there a way to combine arbitrary attacks into more dangerous attacks?
- Identification of upper- and lower bounds of attacks would be of significant help.

In the next chapter we will take the input from this and the previous chapter, i.e. our observations on attacks and attackers in order to do a risk analysis and security evaluation of anonymizing networks.

Chapter 7

Considerations and Conclusion

In this chapter we use the input and results of the previous chapters for two purposes: first we elaborate the holistic security of anonymity systems. Second, we use this information to draw conclusions about the importance of future research areas within this topic.

We finish this work with a summary.

7.1 Attack Tree

Based on the chapters on attacker models (Chapter 5) and attacks (Chapter 6) we build and evaluate a holistic attack tree in this section. To this end we make a brief analysis of the attack tree under various conditions. For one, we will identify the attacks which pose the biggest threat today. But we also try to see beyond that and identify possible scenarios of serious future attacks.

For us it is not a viable approach to list all attacks and order them by their cost. This is because of the huge number of possible attacks and resulting combinations. Hence, any such list would not be meaningful and comprehensible.

Instead, once we identified the most effective attack we remove essential nodes from the attack graph such that this attack is not possible any more. The evaluation is iterated on the resulting graph until no more successful attacks are found. This provides us with a more legible list of attacks. To a certain extent, this also models the behaviour of system designers and programmers which discover vulnerabilities of their system and start to fix these. However, it is impossible to model the impact of yet-unknown weaknesses.

The survey on attacks on anonymizing networks provides us with one part of the input needed to build an *attack tree*. This is a generic representation of different ways to compromise a system. It was first proposed by Schneier in [Sch00] as a tool for *risk analysis* and evaluation of security properties.

An attack tree can be used to find ways of minimal effort to attack systems. The disadvantage of attack trees is, however, that there is no methodology to ensure its completeness. This means that if the creator of an attack tree has disregarded an attack (possibly because it was not known publicly by the time of the creation) the results of the attack tree's evaluation can be misleading. Therefore, one should always keep in mind that there could be a number of new attacks to a system, which are to be published in the future, but could already be known to an adversary of the system.

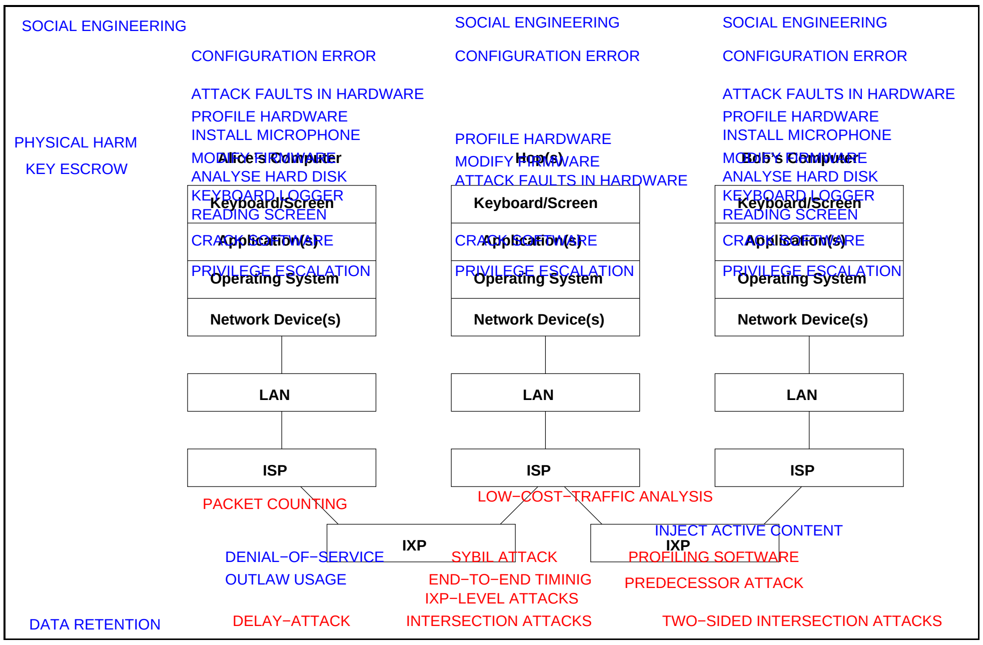

Figure 7.1: Attacks on Networks

To give a short overview over the attacks which we presented in the previous sections, we created Figure 7.1. It bases on Figure 6.1 on page 92, but we inserted the attacks as listed in the previous chapter into the picture of the message's flow. Red attacks can be found in the "traditional" literature on anonymizing networks, while the blue ones are the attacks which we added in this work.

With this input, we build the attack tree as presented in Figure 7.2 on the facing page. This is a graphical representation which is constructed from the list of attacks. Note that due to visual cluttering of the complete data, we clustered attack techniques with similar preconditions and outcome into single points of the graph. It should be noted that all nodes are "or-style" nodes. This means that it is sufficient for an attacker to fulfill a single out of several prerequisites in order to proceed a path with several possible input situations.

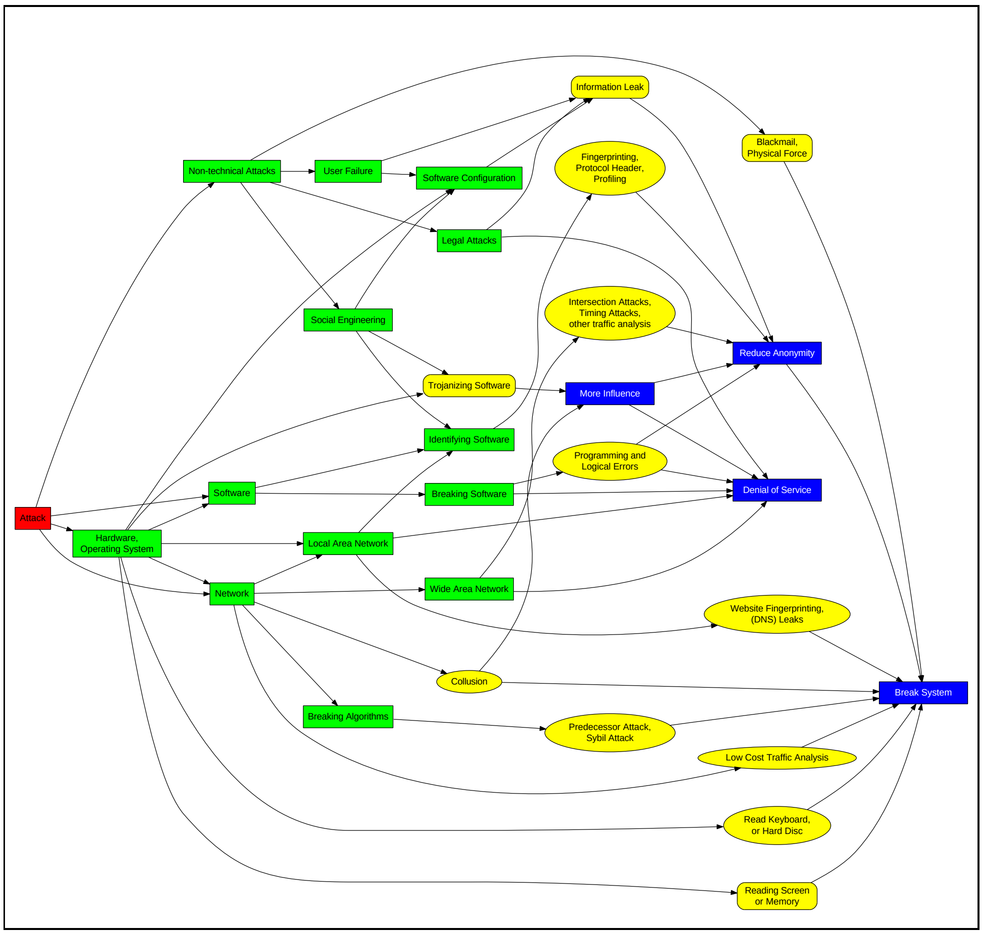

Figure 7.2: An attack tree on anonymizing networks

The attack tree's source is the general opportunity and motivation of the attacker to attack the system, depicted in red on the left-hand side. Starting there, we list basic categories and subcategories in green. These stand for classes of attacks as resembled by the structure of Chapter 6. The actual attacks are depicted in yellow and white boxes. Attacks which are very unlikely to be noticed by a victim are painted yellow; those which have a non-negligible probability to be detectable, are painted in white. The paths finally lead to possible results of attacks, which are painted in blue.

We identified four different types of results for an attack, which can be divided into two groups: the first consists of the results "More Influence" and "Denial of Service". While increasing influence on the anonymizing network does not help an attacker per se, these attacks can be used as step stones to either simplify denial-of-service attacks, or to reduce the degree of anonymity and continue with more elaborate attacks. If an attacker chooses to deny users access to the network, he has reached an end point in the graph. From this point on it is up to the user to decide whether to continue communicating or not. While in the first case his profile will be revealed to the attacker, the information remains secret in the latter case – however, the user is prone to have no means of communication until the adversary is defeated or can be circumvented.

The other group of results are formed by “Reducing the Degree of Anonymity” and “Breaking the System”. Both are instances of the same result, but with different degrees of success. Namely a “Break of the System” means that the adversary has learned all necessary information and is in a position where he can accomplish all his goals. The other is a partial result, where the attacker can exclude a number of users from the anonymity set, or has other information that can be used to link a couple of information. Hence, repeated, continuous, or several attacks with partial results can possibly be combined into a single total attack that breaks the system.

The main difference between these two groups of attack results is that the user is aware of ongoing attacks in the case of denial-of-service attacks. This gives the user an opportunity to stop communication before a critical system breach is achieved. However, in the other case, it might not be noticeable to the user that an attacker has just compromised the system and learns his identity as well as the identity of his peers. Even if the user noticed such an attack, it is too late for any reaction.

7.1.1 Data Conversion

The attack tree as given in Figure 7.2 on the previous page, is only for limited use in algorithms. Therefore, we need to refine the conceptual attack tree into a more detailed one. We also have to estimate the cost and effort for each single attack in order to calculate the cost and effectiveness of a *staged attack*, i.e. a series of single attacks, where the previous attacks are used as step stones for the latter attacks.

There are two main reasons for focusing on staged attacks: first, only a very limited set of attacks on anonymizing networks will lead to immediate success. Fortunately, these attacks have strong preconditions and thereby they cannot be easily mounted on real systems. Second, there is nearly no academic coverage on compound attacks on anonymizing systems, i.e. the effect of an attacker that is capable and willing to launch a set of attacks.

For this evaluation we defined a *format* in which attacks could be formally described in terms of preconditions and requirements, as well as their outcome and results. As an example, the definition of an end-to-end timing attack is given in Figure 7.3 on the facing page.

The format starts with a unique identifier, in this case “EndToEndTiming”. This is followed by a set of preconditions. These are represented by a list of attributes, which can be compared to any other attribute or numerical constant; boolean values can be represented by, e.g., the numerical values zero and one.

```
EndToEndTiming \
  requires PresenceLocal=1, PresencePeer=1, ManPower>=1, \
           ComputationalPower>=2 \
  results  ManPower-=1, ComputationalPower-=2, TotalBreak=1
```

Figure 7.3: Example declaration in the attack-tree definition

In this example, the attacker needs to be present at the first hop of the circuit (`PresenceLocal=1`) and the end of the circuit (`PresencePeer=1`). He needs to have one node for each of these (`ComputationalPower>=2`), and an operator (`ManPower>=1`) for both. The outcome of this attack is that the time of the operator has been spend (`ManPower-=1`[1]), the computers were busy (`ComputationalPower-=2`), but the system is broken with respect to this victim (`TotalBreak=1`).

One part of the resulting attack trees can be seen in Figure 7.4 on the next page.

The attack trees themselves, however, need to be set within the context of a specific anonymizing network, in order to produce results. To this end, we chose a representation of attributes for a representation of the network which is compatible to the notation used above. We also consider the setup of the networks to be static for the duration of an attack.

In order to analyse attack paths with multiple attacks, we need to *iterate* through the attack tree multiple times: for one given input we get a set of possible follow-up situations, one for each possible attack. Due to the multiplicity of outputs and the resulting exponential state explosion, we need to cap the searching depth to a fixed limit. In our case we could evaluate all attack trees which considered staged attacks of up to nine single attacks.

After an attack with minimal cost has been identified, the node which is most relevant to this attack is removed from the attack tree and another iteration is started to find the next best attack. This is repeated until no more attacks are found to be possible.

7.1.2 Results

An overview of the results is given in Tables 7.1 to 7.4. The tables list the best attacks a given attacker can mount on a certain anonymizing system.

These tables are then summarized in Table 7.5 on page 125. To this end, we created a weighted sum of the occurrences. As numerical results can be misleading, we labelled the results according to the relative result of this calculation.

It is immediately obvious from Table 7.5 that a number of these attacks are definitely out of the scope of our topic: developing a secure operating system and

[1]We use compound assignment operators to denote increasing or decreasing values. “`-=`” is for decreasing values, whereas “`+=`” increases values.

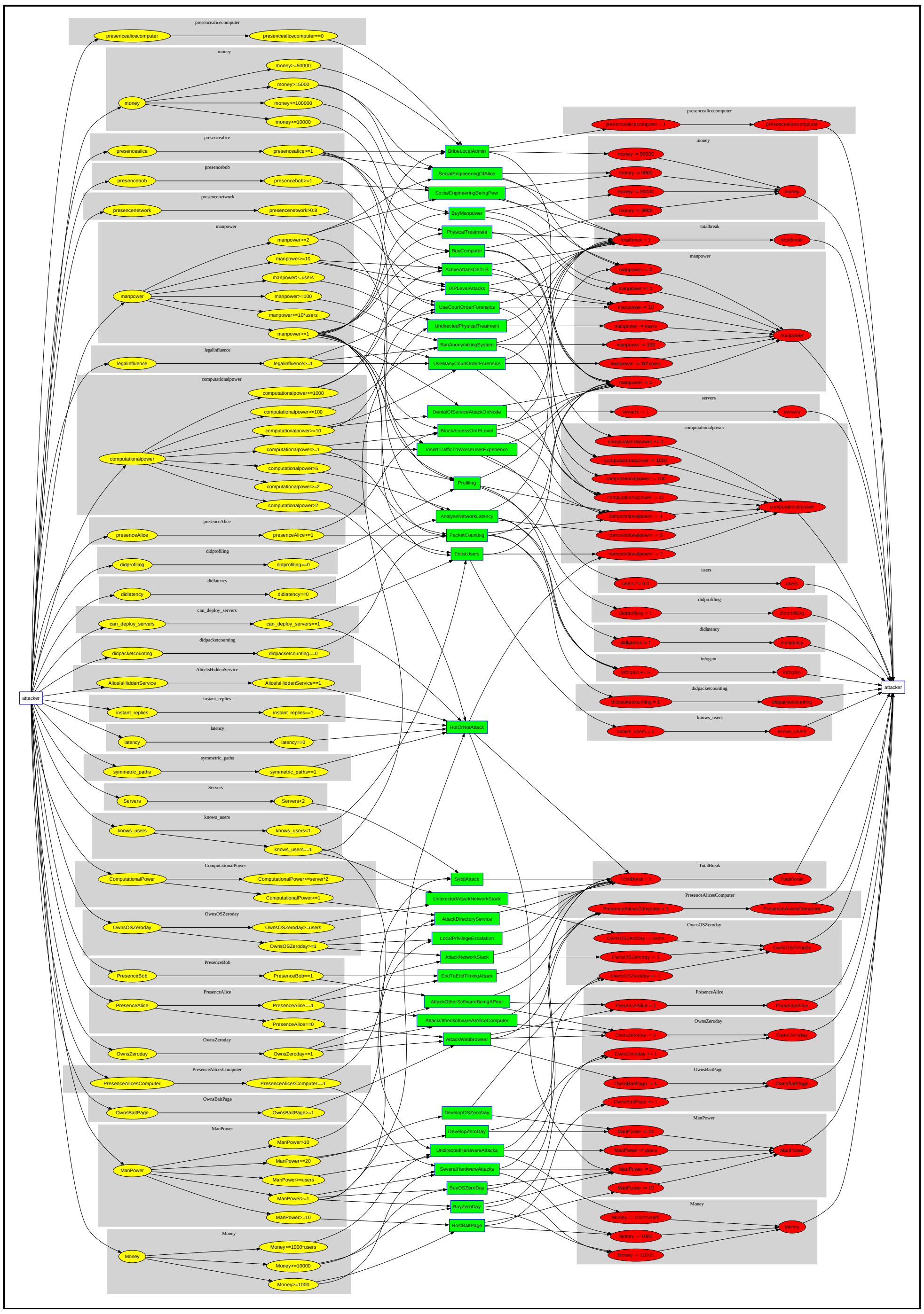

Figure 7.4: A part of the modified attack tree.

Attacker	Attacks
External Party	3x Denial Of Service Attack On Node
Peer	Analyse Network Latency, Profiling
Local Administrator	Packet Counting, Analyse Network Latency
Local Administrator	Blocking Access On IP Level
Local Administrator	Physical Treatment
ISP	2x Denial Of Service Attack On Node, Sybil Attack
ISP	Active Attack On TLS
ISP	Attack Network Stack, Develop OS Zero Day
Secret Service	Develop OS Zero Day, Attack Directory Service
Secret Service	Ban Anonymizing System
Secret Service	IXP Level Attacks

Table 7.1: Best attacks on “AN.ON”

Attacker	Attacks
Peer	Analyse Network Latency, Profiling
Local Administrator	Packet Counting, Analyse Network Latency
Local Administrator	Blocking Access On IP Level
Local Administrator	Physical Treatment
ISP	Active Attack On TLS
ISP	Attack Network Stack, Develop OS Zero Day
ISP	Insert Traffic To Worse User Experience, Undirected Physical Treatment, Enlist Users
Secret Service	Undirected Physical Treatment, Enlist Users
Secret Service	Use Many Court Order Forensics
Secret Service	Develop OS Zero Day, Attack Directory Service
Secret Service	Ban Anonymizing System
Secret Service	IXP Level Attacks

Table 7.2: Best attacks on “I2P”

secure applications are two of those. However, mitigating the *consequences* of these is a valid target.

Attacker	Attacks
Peer	Analyse Network Latency, Profiling
Local Administrator	Packet Counting, Analyse Network Latency
Local Administrator	Blocking Access On IP Level
Local Administrator	Physical Treatment
ISP	Active Attack On TLS
ISP	Attack Network Stack, Develop OS Zero Day
ISP	Insert Traffic To Worse User Experience, Undirected Physical Treatment, Enlist Users
Secret Service	Undirected Physical Treatment, Enlist Users
Secret Service	Use Many Court Order Forensics
Secret Service	Develop OS Zero Day, Attack Directory Service
Secret Service	Ban Anonymizing System
Secret Service	IXP Level Attacks

Table 7.3: Best attacks on “Mixmaster”

Attacker	Attacks
Local Administrator	Blocking Access On IP Level
Local Administrator	Physical Treatment
ISP	Active Attack On TLS
ISP	Attack Network Stack, Develop OS Zero Day
Secret Service	Develop OS Zero Day, Attack Directory Service
Secret Service	Ban Anonymizing System
Secret Service	IXP Level Attacks

Table 7.4: Best attacks on “Tor”

7.2 Conclusion

In the last section, we identified a number of dangerous attacks on anonymizing systems. The three most dangerous attacks are vulnerable software, a central directory service and blocking access to the network. Another dangerous class of attacks are physical attacks.

Attacks which are relevant to “traditional” research in this area, i.e. network layer analysis and active attacks on the network layer, were only marked with “medium” severeness.

In order to enhance the security of anonymizing networks we therefore propose to conduct research to find countermeasures against the more dangerous threats.

Potential	Attack
very high	Buy OS Zero Day
very high	Attack Directory Service
very high	Blocking Access On IP Level
high	Physical Treatment
medium	Analyse Network Latency
medium	Denial Of Service Attack On Node
low	Attack Network Stack
low	Social Engineering Of Alice
low	Active Attack On TLS
low	Packet Counting
low	Develop OS Zero Day
low	Profiling
low	Ban Anonymizing System
low	IXP Level Attacks
low	Attack Other Software At Alice Computer
low	Several Hardware Attacks
rather low	Develop Zero Day
rather low	Buy Zero Day
rather low	Attack Webbrowser
rather low	Undirected Physical Treatment
rather low	Enlist Users
rather low	Use Many Court Order Forensics
rather low	Host Bait Page
rather low	Insert Traffic To Worse User Experience
rather low	Sybil Attack

Table 7.5: Rating of attacks based on tables 7.1 to 7.4

While it is beyond the scope of this work to derive and analyse solutions to all of these problems, there are some basic directions which seem promising:

Missing software security and vulnerable software can possibly be handled by similar solutions as untrusted node operators. Distributing the trust amongst a high variety of different software platforms is a good start to avoid that a single vulnerability can compromise large parts of the network.

For similar reasons it seems promising to have a fair number of completely different implementations of a given service.

Also, keeping the complexity low is another point to ensure that software will not become too complex, not manageable and hence vulnerable. Using well-known building blocks and libraries instead of proprietary solutions can help to reduce the effort of building secure software.

The most advanced system with regards to these points today is Shallon.

Directory Services are amongst the most weak points of an anonymizing system. The use of a central system does not only comprise a single point of failure for availability reasons. It also trivially allows to identify users of the system. In the worst case, if the central directory is compromised, an attacker is able to commit arbitrary harm to any of the users.

In any case, there has been no publicly known research dedicated to creating a secure, performing and anonymous directory service. This lack should be filled rather sooner than later.

Moving to a de-centralized structure, like Tor, where a set of globally distributed people run the directory seems to be a good idea. Also, the use of distributed hash tables for the distribution of information should be more thoroughly investigated.

Blocked Access is, as can be seen in China and Iran, already a hot topic today. There also exists a small amount of research related to the problem to make anonymizing networks harder to detect and more difficult to block.

However, perfect protection is likely to be infeasible: as long as an adversary can get access to the software there will be a way to find and shut down or block nodes of the system. Still, methods might be identified to make this as difficult as possible.

Physical Assaults on users go hand in hand with unobservability. If an adversary is not able to identify the users of a network, and also if the traffic of a user cannot be classified into belonging to an anonymizing network, it is highly unlikely that an adversary will resort to physical assaults.

This means that for those users which actually have to defend themselves against adversaries which do not refrain from physical force, there is a demand for unobservability.

7.3 Summary

In this chapter we used the input of the previous work to discuss and identify a set of serious vulnerabilities in today's anonymizing networks.

We found that contemporary systems suffer from highly dangerous attacks despite the fact that these are usually considered to be “out of scope”. However, in our opinion it is possible to defend against these by broadening the scope of research on enhanced anonymizing networks.

Bibliography

[Acq04] Alessandro Acquisti. Privacy in electronic commerce and the economics of immediate gratification. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 21–29, New York, NY, USA, 2004. ACM.

[AGL05] Andre Adelsbach, Ulrich Greveler, and Sven Löschner. Anonymous Data Broadcasting by Misuse of Satellite ISPs. Berlin, Germany, 2005. Proceedings of 22C3 Chaos Computer Club (CCC) Congress.

[ALFH04] Christer Andersson, Reine Lundin, and Simone Fischer-Hübner. Privacy-enhanced WAP Browsing with mCrowds - Anonymity Properties and Performance Evaluation of the mCrowds System. In *Proceedings of the Fourth annual ISSA 2004 IT Security Conference*, Johannesburg, South Africa, July 2004.

[ALLP07] Timothy G. Abbott, Katherine J. Lai, Michael R. Lieberman, and Eric C. Price. Browser-Based Attacks on Tor. In *Privacy Enhancing Technologies*, pages 184–199, 2007.

[Amo94] Edward G. Amoroso. *Fundamentals of Computer Security Technology*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994. ISBN: 0-13-108929-3.

[Ant08] ANTS File Sharing. `http://antsp2p.sourceforge.net/`, 2008. last visited September 2008.

[ARP82] RFC 826: An Ethernet Address Resolution Protocol. `http://www.ietf.org/rfc/rfc826.txt`, November 1982.

[BB03] David Brumley and Dan Boneh. Remote timing attacks are practical. In *In Proceedings of the 12th USENIX Security Symposium*, pages 1–14, 2003.

[BdB90] Jurjen Bos and Bert den Boer. Detection of disrupters in the DC protocol. In *EUROCRYPT '89: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in* *cryptology*, pages 320–327, New York, NY, USA, 1990. Springer-Verlag New York, Inc. ISBN: 3-540-53433-4.

[BFK00] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.

[BG03] Krista Bennett and Christian Grothoff. GAP – practical anonymous networking. In Roger Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*, pages 141–160. Springer-Verlag, LNCS 2760, March 2003.

[Bis02] Matt Bishop. *Computer Security Art and Science*. Addison-Wesley Professional, December 2002. ISBN: 978-0201440997.

[BMG+07] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-Resource Routing Attacks Against Anonymous Systems. Technical Report CU-CS-1025-07, University of Colorado at Boulder, February 2007. `http://www.cs.colorado.edu/department/publications/reports/docs/CU-CS-1025-07.pdf`.

[BPS00] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45. Springer-Verlag, LNCS 2009, July 2000.

[Bri07] Volker Briegleb. “Cyber-Krieg” in vollem Gange. `http://www.heise.de/newsticker/meldung/95552`, Sep 2007.

[Bro02] Zach Brown. Cebolla – Prgamatic IP Anonymity. Ottowa Linux Symposium, 2002.

[Bun06] Bundesamt für Sicherheit in der Informationstechnik. *IT-Grundschutz-Kataloge*. 2006. ISBN 3-88784-915-9, `http://www.bsi.bund.de/literat/bsi_standard/`.

[CCR+03] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003.

[CDK01] Richard Clayton, George Danezis, and Markus G. Kuhn. Real World Patterns of Failure in Anonymity Systems. In Ira S.

Moskowitz, editor, *Proceedings of Information Hiding Workshop (IH 2001)*, pages 230–244. Springer-Verlag, LNCS 2137, April 2001. `http://www.cl.cam.ac.uk/~rnc1/Patterns_of_Failure.pdf`.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private Information Retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995.

[Cha81] David L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84 – 88, Feb 1981.

[Cha88] David L. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, (1):65 – 75, 1988.

[Col04] Collard Brigitte. *Les langages secrets dans l'Antiquitè grèco-romaine*. Folia Electronica Classica (Louvain-la-Neuve) - Numèro 7, published by Licencièen langues et littèratures classiques, Januar 2004.

[Com02] Douglar Comer. *Computer networks and Internet*. Pearson, Munich, 3 edition, 2002. ISBN: 3-8273-7023-X.

[Cor07] Intel Corp. Intel Core 2 Duo Errata. `http://download.intel.com/design/processor/specupdt/31327914.pdf`, 2007.

[CSWH00] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore 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*, pages 46–66, July 2000. `http://citeseer.nj.nec.com/clarke00freenet.html`.

[DA99] Tim Dierks and Christopher Allen. The TLS Protocol Version 1.0. Internet Engineering Task Force: RFC 2246, Januar 1999. `http://www.ietf.org/rfc/rfc2246.txt`.

[Dan03] George Danezis. Statistical Disclosure Attacks: Traffic Confirmation in Open Environments. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.

[Dan04] George Danezis. The Traffic Analysis of Continuous-Time Mixes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, pages 35–50, May 2004.

[Dav89] Fred D. Davis. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, 13(3):319–340, 1989.

[DCN03] DC net chat program. `http://crypt21.sourceforge.net/`, 2003. visited Aug 2007.

[DD08] George Danezis and Claudia Diaz. A Survey of Anonymous Communication Channels. Technical Report MSR-TR-2008-35, Microsoft Research, January 2008.

[DDM03] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, Washington, DC, USA, May 2003. IEEE Computer Society.

[DDT07] George Danezis, Claudia Diaz, and Carmela Troncoso. Two-sided Statistical Disclosure Attack. In Nikita Borisov and Philippe Golle, editors, *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, Ottawa, Canada, July 2007.

[DFM00] Roger Dingledine, Michael J. Freedman, and David Molnar. The Free Haven Project: Distributed Anonymous Storage Service. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*. Springer-Verlag, LNCS 2009, July 2000.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[Din08] Roger Dingledine. Main Tor specification. `http://freehaven.net/tor/cvs/doc/tor-spec.txt`, July 2008.

[DM] Roger Dingledine and Nick Mathewson. Tor Control Protocol Specification.
https://www.torproject.org/svn/trunk/doc/spec/control-spec.txt.

[DM06a] Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In *The Fifth Workshop on the Economics of Information Security (WEIS 2006)*. Robinson College, University of Cambridge, 2006.

[DM06b] Roger Dingledine and Nick Mathewson. Anonymity Loves Company: Usability and the Network Effect. In *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)*, Cambridge, UK, June 2006.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[Dou02] John Douceur. The Sybil Attack. In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, March 2002.

[DPN08] Thomas Deselaers, Lexi Pimenidis, and Hermann Ney. Bag-of-Visual-Words Models for Adult Image Classification and Filtering. In *Proceedings of 19th International Conference on Pattern Recognition, ICPR 2008*, Tampa, Florida, USA, December 8-11 2008.

[dR07] Theo de Raadt. Mailinglist post openbsd-misc: Intel Core 2. `http://marc.info/?l=openbsd-misc\&m=118296441702631\&w=2`, June 27th 2007.

[DS04] George Danezis and Andrei Serjantov. Statistical Disclosure or Intersection Attacks on Anonymity Systems. Proceedings of the 6th Information Hiding Workshop (IH2004), LNCS, Toronto, 2004.

[DSD04] Claudia Díaz, Len Sassaman, and Evelyne Dewitte. Comparison Between Two Practical Mix Designs. In *Proceedings of 9th European Symposium on Research in Computer Security (ESORICS)*, LNCS, France, September 2004.

[Dul00] Thomas Dullien. Future of Buffer Overflows?, 2000. `http://diswww.mit.edu/menelaus/bt/17418`, Posting on Bugtraq.

[Eur06] European Parliament. Directive 2006/24/EC, on ”the retention of data generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks and amending Directive 2002/58/EC”. `http://europa.eu.int/eur-lex/lex/LexUriServ/site/en/oj/2006/l_105/l_10520060413en00540063.pdf`, 15 March 2006.

[FD04] Nick Feamster and Roger Dingledine. Location Diversity in Anonymity Networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, Washington, DC, USA, October 2004.

[Fed05] Hannes Federrath. Privacy Enhanced Technologies: Methods - Markets - Misuse. In Sokratis K. Katsikas, Javier Lopez, and Günther Pernul, editors, *TrustBus*, volume 3592 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2005.

[Fed06] Hannes Federrath. Anonymität.Online: Technik – Szenarien – Geschäftsmodelle. Starke Anonymität und Unbeobachtbarkeit im Internet (AN.ON), Abschlußbericht, 2006.

[FGM$^+$99] Robert Fielding, Jim Gettys, Jeff Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext Transfer Protocol: HTTP/1.1. Internet Engineering Task Force: RFC 2616, June 1999.

[Fin06] Manfred Fink. Gäste-Überwachung in Hotels durch staatliche und private Schnüffler. `http://events.ccc.de/congress/2006/Fahrplan/events/1505.en.html`, December, 27th 2006.

[FM02] Michael J. Freedman and Robert Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.

[For06] FortConsult. Practical Onion Hacking: Finding the real address of Tor clients. October 2006. `http://packetstormsecurity.nl/0610-advisories/Practical_Onion_Hacking.pdf`.

[FV04] Dan Farmer and Wietse Venema. *Forensic Discovery*. Addison Wesley Professional, 2004. ISBN: 020163497X.

[GAL$^+$07] Timothy G, Abbott, Katherine J. Lai, Michael R. Lieberman, and Eric C. Price. Browser-Based Attacks on Tor. In Nikita Borisov and Philippe Golle, editors, *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, Ottawa, Canada, June 2007. Springer. URL: `http://petworkshop.org/2007/papers/PET2007_preproc_Browser_based.pdf`.

[GHMP04] Dennis F. Galletta, Raymond Henry, Scott McCoy, and Peter Polak. Web site delays: How tolerant are users? *Journal of the Association for Information Systems*, 5(1):1–28, 2004.

[GKK05] Marcin Gogolewski, Marek Klonowski, and Miroslaw Kutylowski. Local View Attack on Anonymous Communication. In *Proceedings of ESORICS 2005*, September 2005. URL: `http://www.im.pwr.wroc.pl/~klonowsk/LocalViewAttack.ps`.

[GM05] Simson L. Garfinkel and Robert C. Miller. Johnny 2: a user test of key continuity management with S/MIME and Outlook Express. In *SOUPS ’05: Proceedings of the 2005 symposium on Usable privacy and security*, pages 13–24, New York, NY, USA, 2005. ACM. ISBN: 1-59593-178-3.

[Gnu01] Gnutella project. http://www.gnutella.com/, 2001. last visited May 2004. Website down since March 2008.
Protocol Specification still available from http://wiki.limewire.org/index.php?title=GDF.
Software still available from e.g. http://www.gnutelliums.com/.

[GRPS03] Sharad Goel, Mark Robson, Milo Polte, and Emin Gun Sirer. Herbivore: A Scalable and Efficient Protocol for Anonymous Communication. Technical Report 2003-1890, Cornell University, Ithaca, NY, February 2003.

[GS03] Mesut Günes and Otto Spaniol. Ant-routing-algorithm for mobile multi-hop ad-hoc networks. In *Network control and engineering for Qos, security and mobility II, ISBN:1-4020-7616-9*, pages 120 – 138. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[GSG02] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002)*, Marseille, France, November 2002.

[Hac07] Hacktivismo Project. Six/Four System. http://www.hacktivismo.com/projects/, 2007. last visited September 2008.

[Hec05] Attacker Classification to Aid Targeting Critical Systems for Threat Modelling and Security Review. http://www.rockyh.net/papers/AttackerClassification.pdf, 2005. visited July 2006.

[Hin02] Andrew Hintz. Fingerprinting Websites Using Traffic Analysis. In Roger Dingledine and Paul Syverson, editors, *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.

[HJW03] Andreas Hirt, Michael J. Jacobson, and Carey Williamson. Survey and Analysis of Anonymous Communication Schemes. Submitted to ACM Computing Surveys, Department of Computer Science, University of Calgary, December 2003.

[HM04] Greg Hoglund and Gary McGraw. *Exploiting Software – How to Break Code*. Addison Wesley, 2004.

[How97] John D. Howard. *An Analysis Of Security Incidents On The Internet 1989-1995*. PhD thesis, Carnegie Mellon University, 1997.

[HSH$^+$08] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: cold boot attacks on encryption keys. *USENIX Security Symposium*, 2008. http://citp.princeton.edu/pub/coldboot.pdf.

[Hüt70] Erich Hüttenhain. Einzeldarstellungen aus dem Gebiet der Kryptologie. Bayerische Staatsbibliothek, Handschriftensammlung, January 1970.

[HVCT07] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-Tin. How Much Anonymity does Network Latency Leak? In *Proceedings of the 14th ACM Conference on Computer and Communications Security (ACM CCS)*, Alexandria, Virginia, USA, October 2007.

[I2P07] I2P project. http://www.i2p.net/, 2007. visited Aug 2007.

[iG07] Leipzig ipoque GmbH. Internet Study 2007. http://www.ipoque.de/media/internet_studies/internet_study_2007, 2007.

[Int01] Assurex International. Kidnappings for Ransom on the Rise. http://www.buzzle.com/editorials/10-7-2001-5152.asp, July 2001.

[IP89] RFC 1122: Requirements for Internet Hosts – Communication Layers. http://www.ietf.org/rfc/rfc1122.txt, October 1989.

[IPV81] RFC 791: INTERNET PROTOCOL, DARPA INTERNET PROGRAM, PROTOCOL SPECIFICATION. http://www.ietf.org/rfc/rfc791.txt, September 1981.

[ISO] ISO/IEC 13335: Information Technology – Guidelines for the Management of IT Security. http://www.iso.org/.

[KAP02] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of Anonymity in Open Environments. In *Information Hiding, 5th International Workshop*. Springer Verlag, 2002.

[KAPR06] Dogan Kesdogan, Dakshi Agrawal, Vinh Pham, and Dieter Rautenbach. Fundamental Limits on the Anonymity Provided by the MIX Technique. In *The 2006 IEEE Symposium on Security and Privacy, Oakland, California, USA*, May 2006.

[KAS04] Kilger, Arkin, and Stutzman. *The honeynet project know your enemy: learning about security threats (second edition)*. Addison Wesley, Boston, USA, 2004.

[KCmW$^+$06] Samuel T. King, Peter M. Chen, Yi min Wang, Chad Verbowski, Helen J. Wang, and Jacob R. Lorch. Subvirt: Implementing Malware with Virtual Machines. In *In IEEE Symposium on Security and Privacy*, pages 314–327, 2006.

[KEB98] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-Go-Mixes Providing Anonymity in an Open System. In D. Aucsmith, editor, *Information Hiding 98 - Second International Workshop*, pages 83 – 98. Springer Verlag, 1998.

[Ker83] Auguste Kerckhoffs. *La Cryptographie Militaire*. le Journal des Sciences Militaires, 1883.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO '96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, pages 104–113, London, UK, 1996. Springer-Verlag.

[Koo08] Bert-Jaap Koops. Crypto Law Survey. `http://rechten.uvt.nl/koops/cryptolaw/`, last visited August 2008.

[Köp06] Stefan Köpsell. Low Latency Anonymous Communication - How Long Are Users Willing to Wait? In Günter Müller, editor, *ETRICS*, volume 3995 of *Lecture Notes in Computer Science*, pages 221–237. Springer, 2006.

[KP03] Dogan Kesdogan and C. Palmer. The Past Present and Future of Network Anonymity. Network Security, Special Issue of Computer Communications Journal, Elsevier, 2003.

[KP04] Dogan Kesdogan and Lexi Pimenidis. The Hitting Set Attack on Anonymity Protocols. In *Proceedings of Information Hiding, 7th International Workshop*. Springer, 2004.

[KP05] Dogan Kesdogan and Lexi Pimenidis. The Lower Bound of Attacks on Anonymity Systems – A Unicity Distance Approach. In *Proceedings of 1st Workshop on Quality of Protection, Colocated at ESORICS*, Milan, Italy, September 2005. LNCS.

[KP06] Dogan Kesdogan and C. Palmer. Technical challenges of network anonymity. *Computer Communications*, 29(3):306–324, February 2006.

[KP08] Anton Kapela and Alex Pilosov. Stealing The Internet. Talk given at DefCon16, `http://blog.wired.com/27bstroke6/files/edited-iphd-2.ppt`, 2008.

[KPK05] Dogan Kesdogan, Lexi Pimenidis, and Tobias Kölsch. Intersection Attacks on Web-Mixes: Bringing the Theory into Praxis. In *Proceedings of 1st Workshop on Quality of Protection, Colocated at ESORICS*, Milan, Italy, September 2005. LNCS.

[Lee] Ying-Da Lee. *SOCKS: A protocol for TCP proxy across firewalls*. http://archive.socks.permeo.com/protocol/socks4.protocol.

[LL06] Marc Liberatore and Brian Neil Levine. Inferring the Source of Encrypted HTTP Connections. In *Proceedings of the 13th ACM conference on Computer and Communications Security (CCS 2006)*, pages 255–263, Alexandria, Virginia, USA, October 2006.

[LNW07] Dezernat 3.2 Landeskriminalamt Nordrhein-Westfalen. *Polizeiliche Kriminalstatistik 2007*. Völklinger Str. 49, 40221 Düsseldorf, 2007. ISSN 0171 - 2802.

[LPW$^+$07] Olaf Landsiedel, Lexi Pimenidis, Klaus Wehrle, Heiko Niedermayer, and Georg Carle. More: A Peer-To-Peer Based Connectionless Onion Router. In *Proceedings of IEEE GLOBECOM, Globalcommunications Conference*, Washington DC, USA, November 26th 2007.

[LRWW04] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright. Timing Attacks in Low-Latency Mix-Based Systems. In Ari Juels, editor, *Proceedings of Financial Cryptography (FC '04)*. Springer-Verlag, LNCS 3110, February 2004.

[Lyo] Gordon Lyon. Nmap, a free and open source utility for network exploration or security auditing. http://nmap.org/. last visited September 2008.

[Mac01] Judith Mackay. Global Sex: Sexuality and Sexual Practices Around the World. *Sexual and Relationship Therapy*, 16(1):41–51, 2001.

[MAFH06] Leonardo A. Martucci, Christer Andersson, and Simone Fischer-Hübner. Chameleon and the Identity-Anonymity Paradox: Anonymity in Mobile Ad Hoc Networks. In *Proceedings of the 1st International Workshop on Security (IWSEC 2006)*, Kyoto, Japan, 23–24 Oct 2006.

[Mas91] Simon Mason. *Secret Signals: The Euronumbers Mystery*. Tiare Publications, 1991. ISBN 0-936-65328-0.

[MBG$^+$08] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining Light in Dark Places: Understanding the Tor Network. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 63–76, Leuven, Belgium, July 2008. Springer.

[McC04] Steve McConnell. *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press; 2nd edition, July 2004. ISBN: 978-0735619678.

[MCPS03] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. IETF Internet Draft, July 2003. `http://www.abditum.com/mixmaster-spec.txt`.

[MD04] Nick Mathewson and Roger Dingledine. Practical Traffic Analysis: Extending and Resisting Statistical Disclosure. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, LNCS, May 2004.

[MD05] Steven J. Murdoch and George Danezis. Low-Cost Traffic Analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, Oakland, California, USA, May 2005. IEEE Computer Society Press.

[Moc87] P. Mockapetris. Domain Names – Implementation and Specification. Internet Engineering Task Force: RFC 1035, November 1987.

[MS05] Kevin D. Mitnick and William L. Simon. *The Art of Intrusion : The Real Stories Behind the Exploits of Hackers, Intruders & Deceivers*. Wiley, December 2005.

[Mur06] Steven J. Murdoch. Hot or Not: Revealing Hidden Services by their Clock Skew. In *13th ACM Conference on Computer and Communications Security (ACM CCS)*, Alexandria, Virginia, USA, 2006.

[Mut08] MUTE File Sharing. `http://mute-net.sourceforge.net/`, 2008. last visited September 2008.

[MW08] Steven J. Murdoch and Robert N.M. Watson. Metrics for Security and Performance in Low-Latency Anonymity Systems. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 115–132, Leuven, Belgium, July 2008. Springer.

[MZ07] Steven J. Murdoch and Piotr Zielinski. Sampled Traffic Analysis by Internet-Exchange-Level Adversaries. In *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, Ottawa, Canada, June 2007.

[Nar06] Ryan Naraine. Hackers Selling Vista Zero-Day Exploit. `http://www.eweek.com/article2/0,1895,2073611,00.asp`, Dec 2006.

[Nat77] National Bureau of Standards. *Data Encryption Standard*. U. S. Department of Commerce, Washington, DC, USA, jan 1977.

[Nat01] National Bureau of Standards. *The Advanced Encryption Standard*. U. S. Department of Commerce, Washington, DC, USA, 2001.

[OFB97] Donald P. Orr, J. Dennis Fortenberry, and Margaret J. Blythe. Validity of Self-Reported Sexual Behaviors in Adolescent Women Using Biomarker Outcomes. *Sexually Transmitted Diseases*, 24(5):261–266, 1997.

[ope] The OpenVPN-project. `http://openvpn.sourceforge.net/`.

[ORBO04] Gregory L. Orgill, Gordon W. Romney, Michael G. Bailey, and Paul M. Orgill. The Urgency for Effective User Privacy-education to Counter Social Engineering Attacks on Secure Computer Systems. In C. Richard G. Helps and Eydie Lawson, editors, *SIGITE Conference*, pages 177–181. ACM, 2004. DOI `http://doi.acm.org/10.1145/1029533.1029577`.

[ØS06] Lasse Øverlier and Paul Syverson. Locating Hidden Servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*. IEEE CS, May 2006.

[Per] Mike Perry. TorFlow. `https://www.torproject.org/svn/torflow/`.

[Pfi89] Andreas Pfitzmann. *Diensteintegrierende Kommunikationsnetze mit Teilnehmer-überprüfbarem Datenschutz*. IFB, Springer Verlag, 1989. Dissertation Fakultät für Informatik, Universität Karlsruhe.

[Pfi04] Andreas Pfitzmann. Security in IT Networks: Multilateral Security in Distributed and by Distributed Systems, October 2004. Script for the lectures “Security and Cryptography I+II”.

[PH06] Andreas Pfitzmann and Marit Hansen. Anonymity, Unlinkability, Unobservability, Pseudonymity, and Identity Management - A Consolidated Proposal for Terminology. Draft, version 0.28, May 2006.

[Pim07] Lexi Pimenidis. On Anonymity Loss in Open Environments. In *Proceedings of 12th Nordic Workshop on IT-Security, NordSec*, Reykjavik, Iceland, Oct 2007.

[PK05] Lexi Pimenidis and Tobias Kölsch. Transparent Anonymization of IP Based Network Traffic. In *Proceedings of 10th Nordic Workshop on Secure IT-systems, October 2005*, Tartu, Estonia, October 2005.

[PP06a] Andriy Panchenko and Lexi Pimenidis. Fundamental Limits of Anonymity Provided by Crowds. In *8th International Symposium on System and Information Security SSI'2006*, Sao Paulo, Brazil, 8-10 November 2006.

[PP06b] Andriy Panchenko and Lexi Pimenidis. Towards Practical Attacker Classification for Risk Analysis in Anonymous Communication. In *Communications and Multimedia Security, Lecture Notes in Computer Science 4237*, pages 240–251, Heraklion, Greece, October 2006.

[PP07a] Andriy Panchenko and Lexi Pimenidis. Crowds Revisited: Practically Effective Predecessor Attack. In *Proceedings of the 12th Nordic Workshop on Secure IT-Systems (NordSec 2007)*, Reykjavik, Iceland, October 2007.

[PP07b] David R. Piegdon and Lexi Pimenidis. Targeting Physically Addressable Memory. In *Proceedings of Fourth GI International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment*, pages 193–212, 2007.

[PP08] Andriy Panchenko and Lexi Pimenidis. Onioncoffee, a free Java implementation of the Tor protocol. `http://onioncoffee.sourceforge.net/`, 2008.

[PPR08] Andriy Panchenko, Lexi Pimenidis, and Johannes Renner. Performance Analysis of Anonymous Communication Channels Provided by Tor. In *Proceedings of the Third International Conference on Availability, Reliability and Security (ARES 2008)*, Barcelona, Spain, March 2008. IEEE Computer Society Press.

[Ray00] Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.

[Rel04] Reliable. `http://www.bigfoot.com/~potatoware/reli/`, 2004. Remailing software, last visited January 2008, site was shutdown in summer 2008.
Software is still available from `http://www.iks-jena.de/mitarb/lutz/anon/privacy/potato.html`.

[RES03] Gregory M. Rose, Roberto Evaristo, and Detmar Straub. Culture and Consumer Responses to Web Download Time: A Four-Continent Study of Mono and Polychronism. In *IEEE Transactions on Engineering Management*, volume 50, pages 31–44, Feb 2003.

[Rog05] Marcus Rogers. The Development of a Meaningful Hacker Taxonomy: A Two Dimensional Approach. In *NIJ National Conference 2005*, volume Marcus Rogers, Washington, DC, 07 2005. National Institute of Justice. CERIAS TR 2005-43.

[Rol06] Stephen Rollyson. Improving Tor Onion Routing Client Latency. Technical report, Georgia Tech College of Computing, 2006.

[RP02] Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, pages 66 – 92, April 1998.

[Rut07] Joanna Rutkowska. bluepillproject.org. `http://bluepillproject.org/`, 2007.

[SB08] Robin Snader and Nikita Borisov. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *Proceedings of the Network and Distributed Security Symposium - NDSS '08*. Internet Society, February 2008.

[Sch96] Bruce Schneier. *Applied Cryptography (Second Edition)*. John Wiley & Sons, Inc, 1996. ISBN 0-471-11709-9.

[Sch00] Bruce Schneier. *Secrets & Lies: Digital Security in a Networked World*. John Wiley & Sons, Inc., New York, NY, USA, 2000.

[SDOF07] Stuart E. Schechter, Rachna Dhamija, Andy Ozment, and Ian Fischer. The Emperor's New Security Indicators. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 51–65, Washington, DC, USA, 2007. IEEE Computer Society. URL: `http://usablesecurity.org/emperor/`.

[SDS02] Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a Trickle to a Flood: Active Attacks on Several Mix Types. In Fabien Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, October 2002.

[Sec] Tenable Network Security. Nessus, vulnerability scanning software. `http://www.nessus.org/`. last visited September 2008.

[Sha49] C. E. Shannon. Communication Theory of Secrecy Systems. In *Bell System Technology Journal*, volume 28, pages 656 – 715, 1949.

[Shi02] Debra Littlejohn Shinder. *Scene of the Cybercrime – Computer Forensics Handbook*. Syngress, Shinder Books, 2002.

[SKW$^+$99] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. *The Twofish encryption algorithm: a 128-bit block cipher*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[SLB07] Micah Sherr, Boon Thau Loo, and Matt Blaze. Towards Application-Aware Anonymous Routing. In *HOTSEC'07: Proceedings of the 2nd USENIX Workshop on Hot Topics in Security*, pages 1–5, Berkeley, CA, USA, 2007. USENIX Association.

[Spi03] Sarah Spiekermann. Die Konsumenten der Anonymität - Wer nutzt Anonymisierungsdienste? *Datenschutz und Datensicherheit*, 27(3), 2003.

[SS03] Andrei Serjantov and Peter Sewell. Passive Attack Analysis for Connection-Based Anonymity Systems. In *Proceedings of ESORICS 2003: European Symposium on Research in Computer Security (Gjøvik), LNCS 2808*, pages 116–131, October 2003.

[STRL00] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an Analysis of Onion Routing Security. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.

[Syv03] Paul Syverson. The Paradoxical Value of Privacy. In *2nd Annual Workshop on Economics and Information Security (WEIS 2003)*, College Park MD, 2003.

[Tan03] Andrew S. Tanenbaum. *Computer networks*. Pearson, Munich, 4 edition, 2003. ISBN: 3-8273-7046-9.

[TCP81] RFC 793: TRANSMISSION CONTROL PROTOCOL, DARPA INTERNET PROGRAM, PROTOCOL SPECIFICATION. `http://www.ietf.org/rfc/rfc793.txt`, September 1981.

[TECA07] Janice Tsai, Serge Egelman, Lorrie Cranor, and Alessandro Acquisti. The Effect of Online Privacy Information on Purchasing Behavior: An Experimental Study. In *Proceedings of Workshop on the Economics of Information Security*, Pittsburgh, USA, June 7-8 2007.

[Tem01] Temporary Committee on the ECHELON Interception System. A5-0264/2001, Report on the existence of a global system for the interception of private and commercial communications (ECHELON interception system). Published at `http://www2.europarl.eu.int/omk/OM-Europarl?PROG=REPORT&L=EN&PUBREF=-//EP//TEXT+REPORT+A5-2001-0264+0+NOT+SGML+V0//EN`, July 2001. Editor: Gerhard Schmid.

[Tora] Tor Network Status. `https://torstatus.kgprog.com/`.

[Torb] Tor Node Status. `https://torstat.xenobite.eu/`.

[TzuBC] Sun Tzu. *The Art of War (translation 1963, S.Griffith(translator))*. Oxford University Press, Oxford, 6th cent. B.C.

[UDP80] RFC 768: User Datagram Protocol. `http://www.ietf.org/rfc/rfc768.txt`, August 1980.

[VSV05] Antti Vaha-Sipila and Teemupekka Virtanen. BT-Crowds: Crowds-Style Anonymity with Bluetooth and Java. In *HICSS '05: Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 9*, page 320.1, Washington, DC, USA, 2005. IEEE Computer Society.

[WALS02] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. An Analysis of the Degradation of Anonymous Protocols. In *Proceedings of the Network and Distributed Security Symposium - NDSS '02*. IEEE, February 2002.

[WALS04] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. In *ACM Transactions on Information and System Security TISSEC'04*, volume 7 (4), pages 489 – 522. ACM Press, November 2004.

[Wes08] Benedikt Westermann. Entwicklung und Evaluation eines einfachen und effizienten Anonymisierungsverfahrens basierend auf offenen Standards. Master's thesis, RWTH Aachen, Januar 2008.

[WHF07] Rolf Wendolsky, Dominik Herrmann, and Hannes Federrath. Performance Comparison of low-latency Anonymisation Services from a User Perspective. In Nikita Borisov and Philippe Golle, editors, *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, Ottawa, Canada, June 2007. Springer.

[WT99] Alma Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *8th USENIX Security Symposium*, 1999. `citeseer.ist.psu.edu/whitten99why.html`.

[WWP07] Karel Wouters, Brecht Wyseur, and Bart Preneel. Lexical Natural Language Steganography Systems with Human Interaction. In *Proceedings of the 6thth European Conference on Information Warfare and Security*, 2007.

[Zel07] Lenny Zeltser. Emerging Information Security Threats. Information Security magazine, `http://www.zeltser.com/emerging-threats2007/`, May 2007.

[ZLCH06] Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. A framework for authorship identification of online messages: Writing-style features and classification techniques. *J. Am. Soc. Inf. Sci. Technol.*, 57(3):378–393, 2006.

Appendix A

Mailinglist Extracts: Hosting provider misbehaviour

The following emails were send on the open german mailing list `exitnodes@lists.ccc.de`. However, as there is no publicly available archieve, we included emails which are of relevenace to this work in this appendix. Where appropriate, the emails were shortened in order to save space; in this case the original text has been replaced with “`[..]`”.

As the major nodes of anonymizing services are not run by academic institutes, mailinglists are the major source of information on events around Tor nodes.

A.1 Thread on OVH

```
Date: Tue, 05 Aug 2008 14:57:02 +0200
From: "Karsten N." <tor-admin@privacyfoundation.de>
To: exitnodes@lists.ccc.de
Subject: Warnung vor OVH

Hallo Exit-Node-Liste,

vor kurzem haben wir noch den ISP OVH aufgrund der guten Anbindung und
des bisher problemlosen Betrieb von TOR-Servern empfohlen.

Seit ein paar Tagen sind die TOR-Exit-Nodes "gpfTOR4" und
"humanistischeunion1" von OVH blockiert.

Die Server sind nicht gekündigt, sie laufen noch, aber OVH blockiert
vollständig den Zugriff auf die Server. Es kommt kein Bit mehr durch.
Diesen Zustand will OVH bis zum Ende der Vertragslaufzeit aufrecht
```

erhalten.

Grund für die Blockade von gpfTOR4 ist der Download einer einzigen
urheberrechtlich geschützten Datei via BitTorrent über TOR. Folgender
Auszug einer E-Mail reichte OVH, um die Blockade zu veranlassen:

```
> > Title: Call of Duty 4: Modern Warfare
> > Infringement Source: BitTorrent
> > Infringement Timestamp: 10 Jul 2008 16:40:18 GMT
> > Infringement Last Documented: 10 Jul 2008 16:40:18 GMT
> > Infringer Username: Infringing Filename: Call of duty 4 [PC-DVD]
[English] [www.topetorrent.com]
> > Infringing Filesize: 6789794354
> > Infringer IP Address: 91.121.26.150
> > Infringer DNS Name: gpftor4.privacyfoundation.de
> > Infringing URL: http://tracker.prq.to/announce
```

Es gibt keine Möglichkeit der Stellungnahme von unserer Seite zu den
Vorwürfen und keinen Hinweis, wer da behauptet, die Datei wäre auf
unserem TOR-Server bereitgestellt worden.

Dieses kundenunfreundliche Verhalten soll ab sofort Standard bei OVH
werden und betrifft nicht nur BitTorrent Downloads, siehe:

   http://forum.ovh.de/showthread.php?t=4356

Beide Server liefen mit der Default-Exit-Policity, die typische
BitTorrent-Ports blockiert.

Karsten N.

From: Sven Anderson <tor@kaputtendorf.de>
To: exitnodes@lists.ccc.de
Subject: Re: Warnung vor OVH
Date: Tue, 5 Aug 2008 17:26:48 +0200

[..]

So eine Mail ist bei mir auch schon mal angekommen, und ich habe es
zunächst auch für eine Falschaussage gehalten, da ein Tor-Client ja
nur aktiv Verbindungen aufbauen kann und so nichts zum Download
anbieten kann. Ein Test hat dann aber ergeben, dass selbst Clients
hinter NAT oder eben Tor sehr wohl Dateien zum Download anbieten

können. Bittorrent-Clients bekommen über den Tracker mitgeteilt,
welche anderen User noch einen bestimmten Teil einer Datei benötigen
und bauen dann gelegentlich selbst eine Verbindung zu diesem anderen
Client auf, um das Dateisegment hochzuladen. So gesehen kann auch ein
Torrent-Client hinter Tor eine Datei zum Download anbieten.

>Beide Server liefen mit der Default-Exit-Policity, die typische
>BitTorrent-Ports blockiert.

Das funktioniert aber leider nicht. Die P2P Verbindungen, also die
eigentlichen Dateitransfers, laufen auf beliebigen Ports, und können
so nicht blockiert werden.

Da mich dieser Filesharing-Abuse sowieso genervt hat, habe ich im
Wesentlichen nur noch Port 80 und 443 in der Exit Policy geöffnet.

Sven

Date: Wed, 06 Aug 2008 14:39:59 +0200
From: "Karsten N." <tor-admin@privacyfoundation.de>
To: exitnodes@lists.ccc.de
Subject: Re: Warnung vor OVH

Dr. Morpheus schrieb:
> Ein Unding; ich zahle bei Hetzner im Monatszyklus.
>
> Dann musst du wohl außerordentlich kündigen und Herausgabe der
> restlichen Monatsraten fordern (§ 812 BGB), etwaigen Schadensersatz
> würde ich mir auch nicht entgehen lassen.

OVH hat soeben fristlos gekündigt:

Zitat aus der Kündigung: "Ein Anspruch auf Erstattung der
vorrausbezahlten Beträge besteht nicht!"

> Schnapp dir einen Anwalt, denn SO geht das nicht.

Das werden wir tun.

Karsten

A.2 Hosting provider kills processes

```
Date: Thu, 3 Jul 2008 16:14:48 +0200
From: Jens Kubieziel <maillist@kubieziel.de>
Cc: exitnodes@lists.ccc.de
Subject: Re: Tor-freundliche ISPs?

* Martin Schobert schrieb am 2008-06-21 um 16:47 Uhr:
>   mich wuerde interessieren, bei welchen ISPs Ihr Tor-Knoten betreibt und
> inwiefern Eure Erfahrungen mit den ISPs aussehen. Habt Ihr Stress, weil
> Ihr zu Traffic verbraucht? Hostet jemand bei einem ISP, der nicht

Ich habe bei Xantron Tor auf einem vServer laufen gehabt. Irgendwann
starb der immer ab. Nachdem ich zuerst von einem Bug in Tor ausging (war
aktuelles SVN), kam ich dann drauf, dass der Provider offensichtlich
einen Cronjob (sic!) laufen hatte, der alle Prozesse mit dem Namen "tor"
killte. Andere Leute haben mit dem Provider ähnliche Erfahrungen
gemacht.
Momentan läuft ein Server bei Manitu (hostblogger.de). Da das noch recht
neu ist, kann ich keine pos. wie. neg. Angaben machen.

> behauptet, der Traffic sei flat, sondern konkrete Limits angibt, z.B.
> Host-Europe? Ist so ein Provider ggf. entspannter?

Diverse Provider reden von Flat, meinen das aber nicht. Ich hatte vor
längerer Zeit mal eine Umfrage gemacht. Nahezu alle gaben mehr oder
weniger offen zu, dass man als Kunde dann eher unerwünscht ist.

> Benutzt jemand Methoden, um den Traffic zu reduzieren, um so dem
> Provider oder dem eigenen Geldbeutel entgegenzukommen? Oder ballert ihr

Ich begrenze den Traffic so, dass ich monatlich an das monatliche
Maximum komme (i.d.R. <2TB) und dem Netz mind. 100kB zur Verfügung
stelle (sofern das die Leitung hergibt).

> Welche ISPs wuerdet Ihr empfehlen und von welchen abraten?

Richte dich nach der Liste auf
<URL:https://wiki.torproject.org/noreply/TheOnionRouter/GoodBadISPs> und
pflege die nach Gelegenheit auch.
```

```
Date: Fri, 4 Jul 2008 00:11:41 +0200
From: "Hendrik P." <mlists@zankt.net>
```

```
To: Jens Kubieziel <maillist@kubieziel.de>
Cc: Martin Schobert <martin@weltregierung.de>, exitnodes@lists.ccc.de
Subject: Re: Tor-freundliche ISPs?

Ich habe auch seit ca 2 jahren einen Tor-Middelnode bei Xantron laufen
und kann es nur allen Abraten.
Wie Jens schrieb werden Prozesse mit dem Namen "tor" gekillt.
Wenn der Traffic weiterhin anhaelt wird der traffic auf 100kByte/s
gedrosselt, Wenn die Technik/der Service irgendwann mal auf die Mails
antworten wird es dementiert...
Incoming ist nix gedrosselt, outgoing auf 100kByte/s nennt meinereiner
durchaus gedrosselt. Wenn man per mail 5 mal den Techniker losschickt
und nichts findet geht auf einmal outgoing 200kByte/s und das obwohl
"nix" gefunden wurde und die traffic einbussen wohl durch die anderen
nutzer kommen...

Also es wird alles versucht einen los zu werden, da es dem tor-prozess
dennoch geling ein bisschen traffic zu machen lauft der kasten halt
weiter.

Um Xantron als serioesen Hoster zu bezeichnen muessen die noch einiges
tun :)

[...]
```

A.3 Hosting providers do not allow anonymizing services

```
Date: Thu, 03 Jul 2008 16:40:18 +0200
From: Muelli <Muelli@cryptobitch.de>
To: Jens Kubieziel <maillist@kubieziel.de>
Cc: Martin Schobert <martin@weltregierung.de>, exitnodes@lists.ccc.de
Subject: Re: Tor-freundliche ISPs?

[...]

Nun die Antwort von Manitu:
On 03.02.2008 13:53 manitu (Support) wrote:
 > Hallo Herr Müller,
 >
 > um es kurz zu machen: Wir sind zwar für den Datenschutz (das wissen
 > Sie), allerdings haben wir uns aus anderen Gründen gegen TOR-Server in
 > unserem Rechenzentrum entschieden.
 >
 > Die Nutzung von Servern als TOR-Server ist entgegen unserer AGB.
 >
 > Ich vermute, dass sich die restlichen Fragen damit erledigt haben,
 > oder?
 >
 > Viele Grüße
 > Manuel Schmitt
```

```
From: Felix Eckhofer <felix@eckhofer.com>
To: exitnodes@lists.ccc.de
Subject: Re: Tor-freundliche ISPs?
Date: Fri, 4 Jul 2008 18:39:05 +0200

Hi.

On Thursday, 3. July 2008, Olaf Selke wrote:
> die Motivation waere spannend zu erfahren. Liegt es am Traffic oder
> am Aufwand fuer das Abuse Handling falls als Exit Node betrieben? Ich
> vermute letzteres.

Bei mir sah die Antwort auf eine ähnliche Anfrage so aus:

-----------------------------------------------------------------------
>   * Ich plane, eine TOR-Node[1] zu betreiben. In Ihren AGB konnte
>     ich keine Ausschlussklausel o.ä. finden, da ich aber ungern
>     allzu häufig umziehe: Ist das für Sie ein Problem?

ehrlich: Ja. Prinzipiell ist es nicht ausgeschlossen, aber wir haben
hier ungern die Kripo rumlaufen ... :-|

Gegenfrage: Ist das für Sie ein Problem?
-----------------------------------------------------------------------

Mein TOR-Server läuft seitdem bei EUserv. Seit einem DDOS auf den
Rechner (auf den EUserv mit IP-Wechsel reagiert hat, der leider einige
Tage gebraucht hat) vorerst allerdings nurnoch als middleman.

Die Story dazu war allerdings ganz lustig: Ein paar Tage vorher gabs
eine Beschwerde an die Abuse-Adresse mit dem Zusatz, man werde falls
nötig selbst dafür sorgen, den bösen Rechner aus dem Netz zu kicken.
Meine Vermutung: Da hatte sich wohl jemand mittels TOR mit einem
Botnetz-Betreiber angelegt...
```

Index

Curriculum Vitae
Alexis Pimenidis

Alexis Pimenidis
Krakaustraße 3
52064 Aachen, Germany

Professional Experience

04/2008 - now	**University of Siegen, Germany** **Chair for IT Security** Research Assistant
01/2007 - now	**Freelance Consultant** Consulting in IT security
03/2004 - 04/2008	**RWTH Aachen University, Germany** **Chair for Communication and Distributed Systems** Research Assistant
01/2004 - 03/2004	**RWTH Aachen University, Germany** **Chair for Distributed Systems** Technical Administrator
08/1997 - 12/2003	**EDV Büro Jones, Aachen, Germany** Software Development, Administration, Database Design, Project Lead, Project Acquisition
1995 - 1996	**Systemhaus Ahrens, Lüdinghausen, Germany** Software Development and Database Design

Education

02/2009	– Defense of PhD thesis at RWTH Aachen University
03/2004 - 02/2009	– PhD student at RWTH Aachen University
09/1996 - 11/2003	– Diploma student in computer science at RWTH Aachen University
10/1995 - 07/1996	– Military service
07/1987 - 07/1995	– Abitur (university-entrance diploma), Gymnasium Canisianum, Lüdinghausen

Organisation and Participation of Events

12/2008	– Invited panel member at 25C3 congress in Berlin, Germany
09/2008	– Reviewer for IFIP/FIDIS summerschool in Brno, Czech Republic
09/2008	– Reviewer for "CAST Next-Generation Day", Darmstadt, Germany
08/2008	– Organisation and host of the IT security exercise "Cipher 4"
02/2008	– Program committee member of the German IT Security Conference 2008, Track: Privacy Enhancing Techniques
08/2007	– Co-Foundation and organisation of the first PET-Con, a German Convention for PhD-students researching Privacy Enhancing Techniques
2005 – 2007	– Host of IT security exercises "Cipher", "Cipher 2" and "Cipher 3"