AALBORG UNIVERSITY
COPENHAGEN

Trimester:3rd

Title: Generative Internet Technologies Supporting Freedom of Expression

Project Period: Fall 2016

TrimesterTheme:Thesis

Aalborg University Copenhagen
A.C. Meyers Vænge 15
2450 København SV

Coordinator: Reza Tadayoni

Secretary: Maiken Keller

Abstract:

Freedom of expression has come under pressure in recent years from different sides, e.g. massive government surveillance programs, political correctness, social pressure etc.
This is a study of how generative Internet technologies can be combined into an open platform supporting freedom of expression, using the sound approach on which the Internet was based.
Fundamental requirements and a proposed solution are presented.

Supervisor(s):

Henning Olesen

Project group no.:

Members
(do not write CPR.nr.):

Henrik Strøm

Pages: 62
Finished: 2016-10-27

When uploading this document to Digital Exam each group member confirms that all have participated equally in the project work and that they collectively are responsible for the content of the project report. Furthermore each group member is liable for that there is no plagiarism in the report.

Generative Internet Technologies Supporting Freedom of Expression

Henrik Strøm

Contents

1 Introduction

1.1 A Brief History of the Internet

In the early 1970's, the U.S. Department of Defence (DoD) was looking for a way to connect two of their existing networks: the ARPA (Advanced Research Projects Agency) packet radio network and the wired ARPANET, to give users of the ARPA radio network access to services on ARPANET[1, p. 106]. Thus the DARPA (Defence Advanced Research Project Agency) Internet Architecture project was started[1, p. 106]. Rather than building a new, homogeneous network, DoD opted to build an *inter-network*, that could interconnect various kinds of networks[1, p. 107].

Because of the military context of the network, the foremost priority of the network was *survivability* – the network should continue to work despite the loss of networks or gateways, as long as there was *some physical path* through the network[1, p. 107-108].

The network also had to support multiple types of communication, e.g. *remote login*[1, p. 107], the *XNET* protocol for debugging[1, p. 108] and voice[1, p. 108]. These services are very different in nature, e.g. *reliable transmission* is required for services such as remote login and file transfer[1, p. 108], whereas XNET debugging and voice services would work far better in a low-latency but lossy network[1, p. 108]. This led to a fundamental change in the network design – the initial design had been based on the *Transmission Control Protocol* (TCP), a reliable protocol – the new approach was a two-layer protocol stack: *Internet Protocol* (IP), an *unreliable* protocol serving as a *best effort* building block for other services, and on top of that the TCP protocol for services requiring reliability, and a likewise application level *User Datagram Protocol* (UDP) for low-latency applications with no need for reliability[1, p. 109].

Another requirement of the network was that it should be able to interconnect different existing network technologies[1, p. 107]. This was achieved by making minimum assumptions about the network – basically it's just assumed that the network can transport a packet or datagram[1, p. 109]. Capabilities such as reliability, sequenced delivery, prioritization of packets are explicitly *not* assumed[1, p. 109].

Distributed management of the network, the effectiveness of the network, low affordance in connecting to the network and accountability were also requirements, but with lower priorities[1, p. 107].

During the 1970's and 1980's, a growing number of U.S. universities were connected

to what was then known as the *Internet*[2].

The *end-to-end argument* was introduced in 1984 by Saltzer, Clark, and Reed[3] and became a very influential principle in the design of the Internet – later known as the *procrastination principle*[4, p. 31]. The basic idea behind the end-to-end argument is to generally defer implementations from the network to the end nodes – thus procrastinating. The first argument in the article is the *careful file transfer*, showing that a reliable copy made by two hosts can only be sufficiently implemented at the end nodes, because any number of things can go wrong after the file has reached the destination node, so reliability has to be implemented at the end node anyway, and such functionality built into the network will at best be a performance optimization[3, p. 278-282]. The matter of identifying the *real end nodes* is not even trivial, and might be outside the system itself, as seen with voice packets on a network – the real end node might be considered the user, because he can simply ask the person at the other end of the line to repeat if there was a dropout[3, p. 284-285].

Another way we see the influence of the *end-to-end* argument is in that there is no identity layer on the Internet[4, p. 32]. Identity management would be handled by the end nodes rather than the network, and this rhymes perfectly with the low priority of management and accountability[1, p. 107].

Figure 1: The lack of an identity layer on the Internet means nobody knows who you are.

The foundation for use of the Internet outside government and universities was laid with *the world wide web*[5] by Tim Berners-Lee. The *world wide web* was the specification of protocols, document standards, and tools allowing non-technical users to gain access to information via the Internet. *HTML documents* would be transmitted via the text-based *HTTP protocol*, and displayed in the user's *browser*[5]. The user could access documents via their *universal resource locator*, and documents could *link* to each other[5]. Because the Internet was build to support various types of communication, the *HTTP protocol* could be layered on top of the *TCP/IP protocol suite*.

While Tim Berners-Lee was working on the *world wide web*, commercial companies like *America Online* and *Compuserve* were both offering a *walled garden* Internet experience[4, p. 29]. Users could access certain information on these networks, send emails etc., but America Online and Compuserve acted like arbiters of these networks – nobody could offer a service on these networks without a contract with the network operators, and like AT&T before them, they developed a bias for certainty rather than risk-taking innovation[4, p. 25]. These companies did their best to *own the Internet* – I remember trying to set up Windows 95 for Internet access would actually connect you to a similar *MSN Network* rather than the real Internet (users had to set up access to a *custom network instead* to get real Internet access, and fill in their *Internet Service Provider* (ISP) details), and most recently *Facebook* has been trying to do the same in India[6].

Commercial activities on the Internet became legal in 1994 when the U.S. Congress passed legislation allowing this and quickly attracted users. Unlike the walled gardens of America Online, Compuserve, and Microsoft, the Internet was a *free and open network*.

The commercialization of the Internet quickly attracted the existing IT industry companies as well as new *start-ups*, encouraged by loads of venture capital from *retail investors* expecting a quick payoff[7, p. 32], causing the infamous *dot-com bubble* at the turn of the century.

While Tim Berners Lee's *world wide web* was based on *static HTML documents*, the *Web 2.0* wave of the early and mid 2000s focused on *the web as a platform*[8, p. 19], enabling non-technical users to participate in generating content rather than just consuming content. The term *Web 2.0* was coined at a 2004 conference brainstorm session between O'Reilly and MediaLive International[8, p. 17], and quickly became an ambiguous buzzword[8, p. 18]. Tim O'Reilly tried to clarify the term in his 2007 article[8] by distinguishing between *Web 1.0* companies' *design patterns* and *business models* contra those of the *Web 2.0* era. A common characteristic of many *Web 2.0* companies is that their most valuable asset (their data) often is created and augmented by their users, e.g Amazon's user recommendations and reviews, and

data collected by these companies about their users, e.g. buying patterns[8, p. 23]. In hindsight, it is quite ironic how Tim O'Reilly promotes Google's advertising business as *"minimally intrusive, context sensitive, consumer friendly"*[8, p. 21], when in fact, Google has turned the user into the product with a business model that *"consists almost entirely of gathering data about the preferences, locations, and behavior of ordinary people and monetizing that data through the sale of targeted advertisements on the Internet."*[9].

Tim O'Reilly points to the *RSS protocol* (Really Simple Syndication) as one of the most significant elements of Web 2.0[8, p. 24]. An *RSS feed* is basically an XML file hosted on a website, e.g. a blog or a news site, containing key information about posts or articles. By extracting the information, making it available in a machine readable format rather than a human-readable format, RRS builds on the ideas of *the semantic web*[10]. The feed can be read by an *RSS reader*, that can collect RSS feeds from many websites, and present a list of updated posts and articles to the user. This way, the user doesn't have to browse the sites, again and again, to look for updates (virtually a *push* rather than a *pull* of information). RSS is built on the generative technologies of the Internet, and is in itself a generative technology[4, p. 56], allowing any system to use its content for whatever purpose.

Google launched *Google Reader* in 2005[11] – a *cloud-based* RSS reader, that enabled users to reach their RSS subscriptions from any browser, and, more importantly, keep RSS readers in sync for users who user more than one device, something that became even more important with the smartphone revolution in 2007. After eight years, when Google reader had pretty much eliminated all competition, Google discontinued Google Reader[11]. Many have speculated, including former product manager of Google Reader Brian Shih[12], that Google did this to drive users into Google Plus, a proprietary *walled garden* owned by Google.

These properties of the Internet design combines into a *general purpose network*, suitable for the plethora of services we see on the Internet today: voice over IP, video conferencing, media services like Netflix, the World Wide Web etc. because they form a *generative*[4, p. 71] technology that services can be built upon, even though many of these properties are more coincidental than intentional.

1.2 Motivation

When the Internet emerged into mass adoption in the late 1990's and early 2000's, it brought along hope for democracy and freedom – there would be a platform for all voices to be heard, we would have open and transparent democracies around the world[13, Introduction].

While the Internet has been a tremendous commercial success, have influenced nearly all aspects of modern life, and has opened up platforms for some groups of people to voice their opinions, it has also opened the doors for *governments*, *cooperations* and *strongly engaged social* groups to control and invade people's lives.

The Internet has brought along unprecedented opportunities for governments to create an Orwellian surveillance network:

> *There was of course no way of knowing whether you were being watched at any given moment. How often, or on what system, the Thought Police plugged in on any individual wire was guesswork. It was even conceivable that they watched everybody all the time. But at any rate they could plug in your wire whenever they wanted to. You had to live – did live, from habit that became instinct – in the assumption that every sound you made was overheard, and, except in darkness, every movement scrutinized.* ~ George Orwell, 1984[14]

George Orwell's dystopian sci-fi novel *1984* presents a totalitarian and authoritarian mass surveillance society, where every move and sound you make will be monitored by *Big Brother*, or rather the troops of collaboraters that everybody had become. A society where the very idea of a *private life* is a thing of the past.

But how far are we really from this situation today, and how far are we willing to go?

1.2.1 Excessive Government Surveillance

The *Edward Snowden leak* of 2013 shed light on several massive, global scale surveillance programs run by the *National Security Agency* (NSA)[15], a United States government intelligence organization tasked with *signals intelligence* (SIGINT) and *information assurance* (IA)[16].

It turned out that the NSA was cooperating and actively exchanging sensitive information about foreign and domestic citizens with intelligence services all over the world[15, p. 123].

Now – about three years later – while still only a small subset of the leaked information has been made available to the public, it has long been clear that Western governments have put in place all-encompassing surveillance programs to spy on its own as well as foreign citizens *en masse*[15] (see section 2.1.1).

These mass surveillance programs are not only violating fundamental democratic and constitutional rights[17], they also have a *chilling effect*[18] on people participating in the democratic and social debate in society.

1.2.2 Social Media Censorship

In the Western world, a lot of public debate now takes place on social media networks such as *Facebook*, *Twitter* and *YouTube* – among ordinary people as well as politicians and other well-known people.

Over the last few years, we have seen various *strongly engaged social groups*, most notably *third wave feminists*, *social justice warriors*, *GamerGate* and various *religious groups* policing social media networks and press outlets for opposing ideas and ideologies, and then attacking these whenever found with *social media shit-storms*[19], *doxing* or simply trying to get people fired from their jobs[19.

While these groups are not actually organized, they do manage to silence opposing opinions to a large degree by making people impose self-censorship and abide by political correctness by instilling a poisonous and hostile environment, and again – creating a *chilling effect*[18].

Social media networks are not immune to the influence of *strongly engaged social groups*. These networks are big business, and in order to retain users, they sometimes get involved as Twitter did when launching the *Twitter Trust and Safety Council*[20], and actively took side of *3rd wave feminists* and *social justice warrior* groups against *GamerGate* and a number of unaffiliated free speech advocates.

Twitter has allowed the *Trust and Safety Council* to silence anybody opposing their opinions without justification, warning or notice, by simply removing their *tweets* from their followers' timelines.

Because social media networks play such a predominant role in the public debate, this becomes a very effective way of censorship.

1.2.3 Silencing Critique with Copyright Legislation

The *Digital Millenium Copyright Act* (DMCA) is the American law dealing with digital content copyright. Social networks operating in U.S.A. are liable for content on their network abiding by the DMCA and therefore have put various *take down* procedures in place, to remove any copyright violating content.

The takedown happens without warning, and it is then up to the person or organization who uploaded the material to disprove the DMCA violation. This is complicated by the *Fair Use doctrine*, which allows content creators to *reuse* copyrighted materials for a number of purposes, e.g. public critique or non-profit educational use. However, the Fair Use doctrine is extremely vague and ambiguos[21][4, p. 122-123], allowing copyright holders to threaten content creators reusing the content in question under supposed fair use.

While filing false DMCA takedowns is a felony in U.S.A., it is often an *unleveled playing field* between media cooperations or in other cases religious organizations, with lots of resources and lawyers, against private people facing imprisonment or devastating economic sanctions if they would lose the case. This allows for resourceful copyright holders to effectively silence critique.

1.3 User Stories

If we are to regain an environment of free speech, what are the things we need to enable people to do? To answer this question, I have worked out a few user stories (see figure 2) based on Andrew Stellman's model[22].

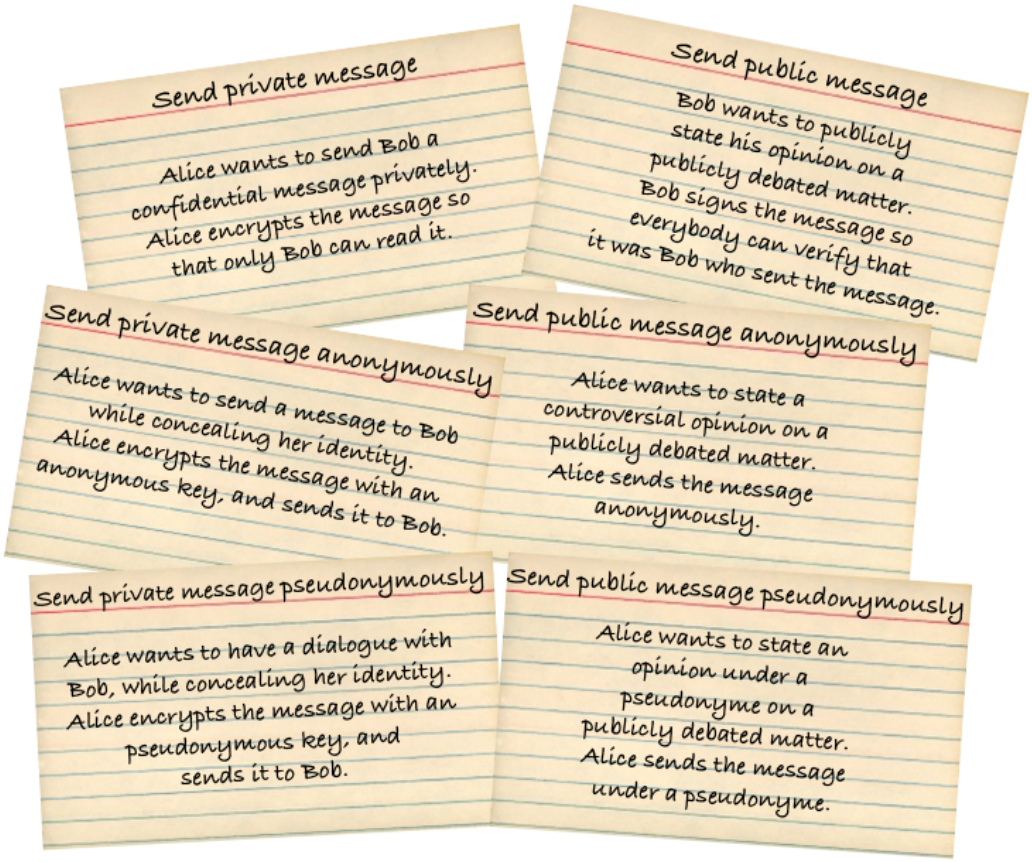

Figure 2: User stories illustrating needed abilities to regain free speech.

US-1 – Send private message: If Alice is able to send a message to Bob in private, and not worry about somebody listening in on the conversation, she can freely

express herself as if she was standing face-to-face alone with Bob. A use case for this user story can be found in section 3.3.1.

US-2 – Send public message: Bob doesn't care about confidentiality, he wants to express his opinion openly. He does, however, care about somebody changing the message to misrepresent him, so he cryptographically signs the message, making it possible for other people to verify the message as sent by Bob, and making it impossible to change the message without introducing a mismatch with the signature. A use case for this user story can be found in section 3.3.2.

US-3 – Send private message anonymously: Alice wants to conceal her identity. Using an anonymous private key, for which Bob can retrieve the public key, but not determine Alice's identity, she can remain anonymous and be confidential. While this might seem like an odd situation, this would likely be the case for a whistleblower contacting a journalist. A use case for this user story can be found in 3.3.3.

US-4 – Send public message anonymously: Alice wants to state a controversial opinion on a public matter. Because she is concerned with reprisals following a controversial statement, she sends the message anonymously, so that nobody can link the message to her. A use case for this user story can be found in 3.3.4.

US-5 – Send private message pseudonymously: Alice wants a sustained, confidential dialogue with Bob while concealing her identity. Using a pseudonymous set of keys, Alice can assure Bob he is communicating with the same person between multiple messages. A use case for this user story can be found in 3.3.5.

US-6 – Send public message pseudonymously: Alice wants to state an opinion on a publicly debated matter. She uses a pseudonym, so people can link Alice's messages together, while Alice can still conceal her identity. A use case for this user story can be found in 3.3.6.

1.4 Vision and Preliminary Design Objectives

So how can we mitigate these issues? We need people to be able to voice their opinions without fear of reprisal, we need people to be able to have a confidential conversation, and we need to ensure that these fundamental democratic rights can't be taken away from us.

People must be able to send messages anonymously, for public view. If people can be truly anonymous, they don't have to fear reprisals unless they give information away themselves that makes them identifiable. This would help whistle-blowers, people who want to raise an unpopular opinion, and even people living under dictatorships voice their opinions.

People also need to be able to send a message with confidence that nobody can listen in or alter the message, and that sending this message can't be used against them later, even if elements of the security is later compromised, or they are talking to somebody else than they think they are.

The fundamental vision of this system is perhaps best formulated by Edward Snowden in the missive send along with his manifesto and the archive of documents to the journalists who would work on the NSA leak[15, p. 23]:

> *Let us speak no more of faith in man, but bind him down from mischief by the chains of cryptography.*[15, p. 24]

1.5 Research Question

This project started out with the goal of finding ways for people to communicate, in a confidential and anonymous manner. However, midway through my project, and well after my initial State of the Arts analysis, Riffle[23] was published, solving pretty much all the problems I was looking at better than I thought possible.

Because I couldn't unlearn what I had learned, I had to take this project in a new direction. My original research question was:

> ***Is it feasible to construct an independent communication system, that offers anonymity, confidentiality and integrity, which is impossible to censor or take control of, and is open and available to practically anybody – and if so, what would be the basic requirements for such a system?***

So, knowing that much of this problem was in fact solved by Riffle, I decided to investigate how this technology could be used to form a generative technology, that other technologies could be built upon. Thus, my new research question is:

> ***How can existing communication technologies be combined and enhanced into a generative technology, supporting freedom of expression?***

The term *freedom of expression* is meant in a broad context, rather than the more narrow constitutional (state/citizen) context.

1.6 Methodology

1.6.1 Desktop Research

Because this is an analytical project, the primary research method used though out the project is desktop research.

In the beginning, conceptual and exploratory research was conducted to get an overview of the research domain. The fundamental problem of challenges to freedom of expression was quite clear to me, giving the *motivation* (see section 1.2) and the *research question* (see section 1.5). These fed into the *State of the Art* chapter (see chapter 2), giving an objective technology overview. Then the *analysis* is performed (see chapter 3), and we can deduct the *requirement specification* (see section 3.6) based on all of the preceding work, and then arrive at a proposed solution (see chapter 4). This process is illustrated in figure 3.

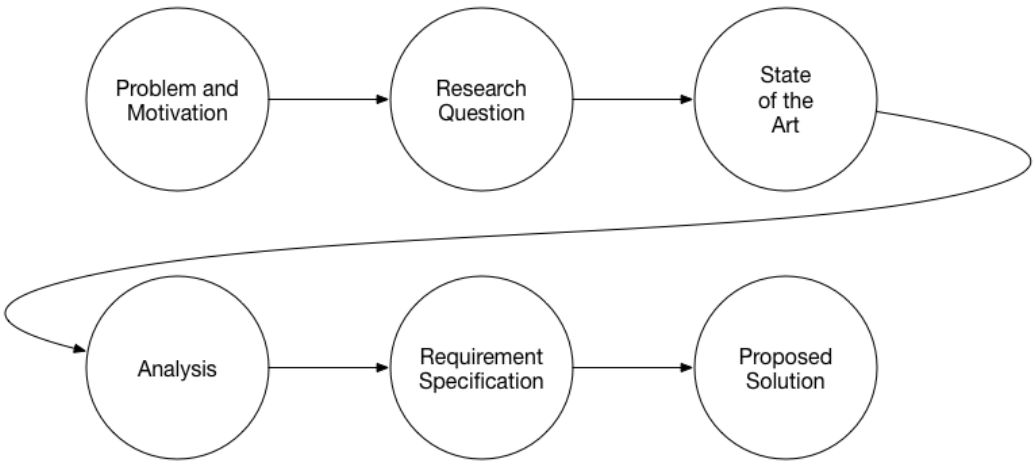

Figure 3: The flow of the report, showing how the requirement specification and proposed solution is deducted.

Finding relevant literature and references, and understanding the technologies used by different systems have also been conducted throughout the project.

1.6.2 User Stories and Use Cases

User Stories have been used to communicate the concept to the reader of the report – (what is it the user needs?) – and Use Cases to further detail the usage of the system – (how will the system behave helping the user with that need?)[22].

1.6.3 Interview

A single interview (see section 3.5) was conducted as *semi-structured interviews*. This allows for a natural flow of conversation and gives the interviewee opportunity to bring up topics or details that I might not know or have thought of.

Rather than having specific questions to ask the interviewee during the interview, I have organized the topics of interest in multiple levels in a mind map. I also use this mind map to make short notes during the interview.

The interview was audio recorded to enable me to directly quote the interviewee, and to liberate myself from having to write down everything during the interview.

As the interview was conducted at a time that I knew of Riffle but had not yet decided to change the research question, it should be seen in that context. The fundamental questions about the legality of the system, however, is just as valid as with the original research question. Unlike the rest of this report, it didn't make sense to rewrite this section after the change of research question, so I have left it as is.

1.7 Expected Outcome

The major outcome of this report is a high-level requirement specification for a solution and a proposed solution based on this requirement specification.

1.8 Limitations

As usual, when dealing with security, there are no *silver bullets*, rather it is more like an *arms race*: one side is constantly trying to increase the security of some system, while someone else is trying to break that increased security. What is adequate for security today, may be inadequate tomorrow. This report will take threats and opportunities that we know of today into account, but these circumstances are rapidly changing.

1.9 Ethical Implications

There are some obvious ethical and legal implications of developing technologies supporting anonymity and confidentiality to support freedom of expression.

As far as the legal issues are concerned, I will get into this in section 3.5.

As for the ethical issues, there is a need for a fundamental decision to be made: *these technologies could be abused for criminal activities, and even terrorism – should there be a back door to address such issues?*

When Apple announced enhanced security as part of their mobile operating system iOS 8, that would make data stored on an iPhone inaccessible to anybody but the user knowing the pin-code or password used for device encryption, *The Washington Post* suggested a *compromise* to allow law enforcement agencies to get access in emergency situations[24], by implementing a *golden key* that would unlock any iPhone. As Chris Coyne pointed out on *Keybase's* blog[25], there are a number of reasons this is a bad idea: *it could get compromised*, allowing hackers access to extremely sensitive information we all accumulate on our mobile devices, *human error* could expose your data, and it would *take away your control of your data.*

There have also been examples of *very unfortunate* situations, where device vulnerabilities exposed users to hackers, such as *Cisco's NSA linked zero-day attack*[26, 27] which exposed Cisco customers for years, and even a *leak of NSA tools*[28].

All experience show[15, p. 98] that back doors will be abused. The NSA mantra of *collect it all*[15, p 98] shows that authorities will not respect people's rights to communicate anonymously and confidentially, and even if they did, their employee will not, as both NSA[29] and the police[30] have had issues.

In the physical world, we wouldn't accept government agencies to constantly peek over the fence, creeping around outside our windows, and entering our homes going through our personal items every time we went out. Obviously, this wouldn't scale very well in the physical world, but just because it is possible to do so in the digital world, we shouldn't accept it any more for that reason.

Of course, law enforcement agencies need to do their jobs, and they need tools to do so. *Targeted* surveillance of *suspects* for *reasonable amounts of time* is fully acceptable, but *en masse* surveillance of *everybody, all the time* is not. When we are constantly under surveillance, law enforcement agencies tend to access the gathered information for petty crimes[31]. When warrants are not needed to dig out people's most intimate secrets, there is no paper trail, and nobody will be held accountable for the need to access this information.

The former head of the Danish *Politiets Efterretningstjeneste* (PET, ~ Police Intelligence Service) Jacob Scharf states in his new book that gathering all this information is of very little use in investigations[32]. This aligns perfectly with an answer given by the Danish Minister of Justice Søren Pind, when asked to justify reinstating a controversial *session logging program*, about what examples could be given on the use of the suspended session logging program in police investigations. The minister was only able to mention four cases, some of which were unlikely to

have been solved because of the session logging program.

So the answer to the question of whether there should be a back-door or a golden key must be a clear unequivocal *NO!*

2 State of the Art

2.1 Examples of Freedom of Expression Challenges

2.1.1 The Edward Snowden Leak

Edward Snowden is kind of an odd person. Feeling unchallenged in high school, he spent his time on the Internet rather than in school, and eventually dropped out of high school[15, p. 40]. Like many other Americans, his political views was changed in more patriotic direction after the 9/11 terror attack in New York and Pentagon[15, p. 40]. He enlisted in the U.S. Army in 2004, to go to Iraq and free the Iraqi people, but was deeply disillusioned by his fellow soldiers' eagerness for *killing Arabs*[15, p. 40]. After a training accident, where both his legs were broken, he was forced out of the U.S. Army, and sought a career in a federal agency[15, p. 40].

Though he didn't have a high school diploma, his tech skills allowed him to find work in IT, and got a *Microsoft Certified Systems Engineer* certification[15, p. 40]. He started working as a security guard at a building used by NSA, to get his *foot in the door*, for an opportunity to do technical work[15, p. 41]. This worked, and he was soon contracted as a technical expert for the CIA, which were desperate for tech savvy people[15, p. 41].

Snowden quickly raised through the ranks, became a full-time employee of the CIA, and in 2007 he was stationed undercover in Switzerland with diplomatic credentials[15, p. 41]. In this position, he started to see the back side of the intelligence world, e.g. a failed operation targeting a Swiss banker, from whom CIA wanted information about certain people's financial transactions. The agents set him up and practically destroyed his life for no reason[15, p. 42].

Meanwhile, Snowden tried to make his superiors aware of *"problems in computer security he thought skirted ethical lines"*, but this was ill received[15, p. 42]. These concerns would always be rebuffed, and he quickly developed a reputation of raising too many concerns[15, p. 42].

The lack of oversight and accountability lead Snowden to leave the CIA, disillusioned once more, and he began considering how to make the public aware of these problems[15, p. 42], but two things stopped him at this point: he had hopes that the election of Barack Obama as president of U.S.A. would change things, but Obama actually expanded on the surveillance programs – he also thought leaking CIA secrets would be too dangerous for the people involved, whereas leaking secrets about the NSA programs would not[15, p. 43]. So he started working for NSA, and was stationed in Japan, with yet higher security clearance, and he got access to

even more disturbing secrets and began to realize the true extend of the surveillance programs NSA was running, and it became more clear to him, that the public had to know what was going on[15, p. 43]. Snowden still advanced, and continuously received an even higher level of training and pay[15, p. 46], and finally ended up in a position in Hawaii, contracting for Booz Allen Hamilton, where he stayed until he *blew the wistle*[15, p. 48].

Edward Snowden leaked evidence of a global surveillance network, including 37 countries other than U.S.A. itself. These countries were not only Western democracies but also totalitarian, repressive regimes such as *Saudi Arabia*, *The United Arab Emirates* and *Pakistan*[33] (see figure 4).

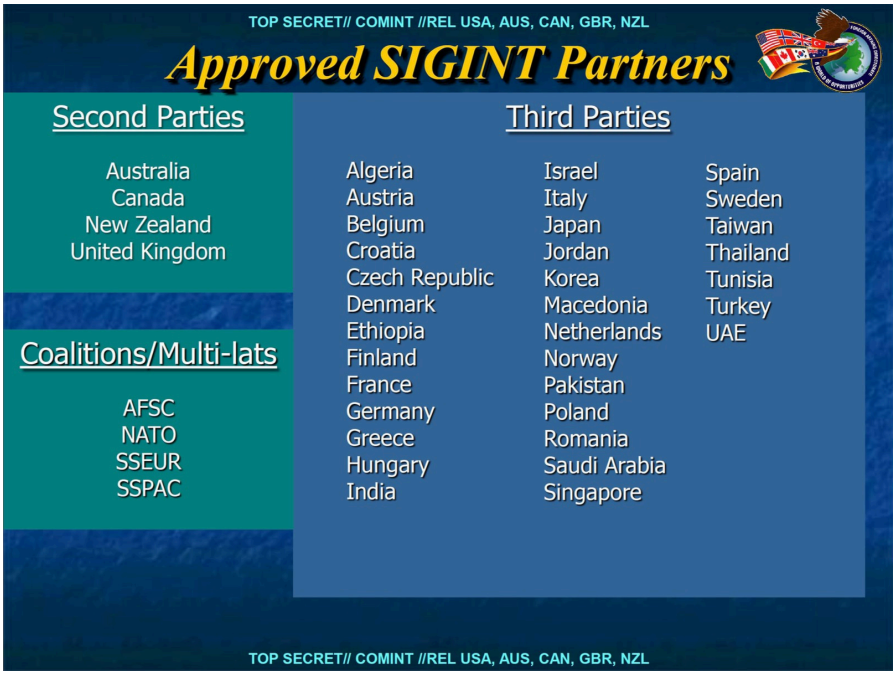

Figure 4: The global scale of the NSA SIGINT programs. Image courtesy of NSA.

NSA also formed a network of strategic partner companies, covering chipset manufacturers, hardware manufacturers, software companies, and telecoms[33] (see figure 5).

Figure 5: NSA strategic partners. Image courtesy of NSA.

NSA are using these partnerships to gain access to central Internet and telecommunications network systems all over the world[33] (see figure 6).

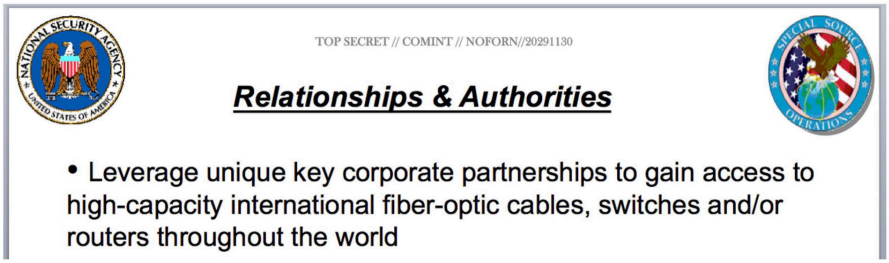

Figure 6: NSA use of SIGINT and strategic partners. Image courtesy of NSA.

The following examples of government agency spy programs is compiled from Glenn Greenwald's *No Place To Hide* book[15, chapter 3] and website[34], *The Guardian's* collection of NSA stories[35], and *The Daily Dot's* guide[36] to the NSA spy programs.

This is by no means a comprehensive list, in fact, it is only a selected few examples, but it illustrates the character and magnitude of these programs.

PRISM is probably one of the most well-known surveillance programs. The PRISM program collects user data and metadata such as emails, files, file transfers, VoIP calls, login and access activity, and even video conference calls[33] (see figure 7).

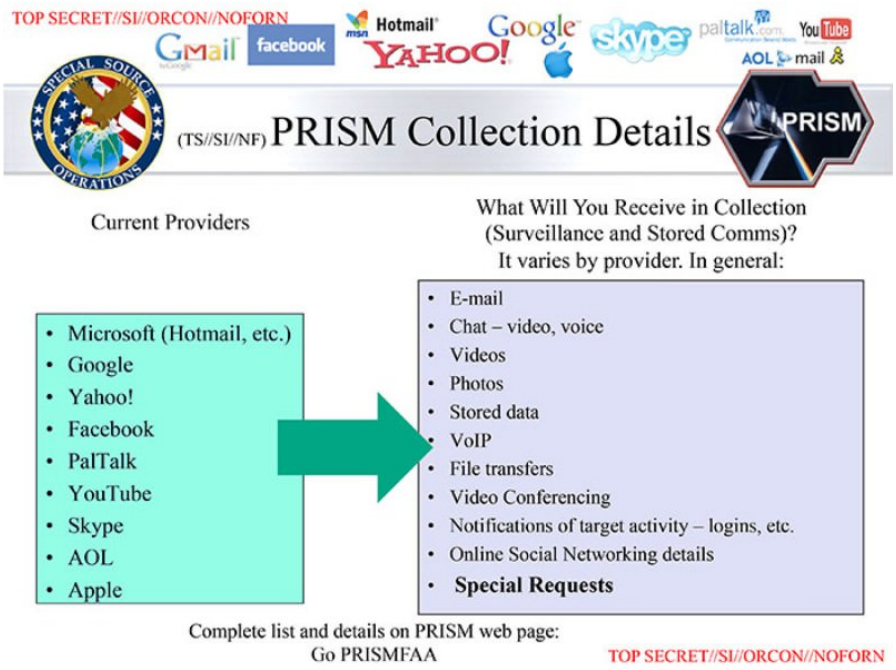

Figure 7: The NSA PRISM program. Image courtesy of NSA.

The PRISM program is done directly in cooperation with the major cloud service providers, such as Facebook, Google (search, Gmail, Docs, YouTube), Yahoo (Mail, search), Microsoft (Office, Skype), Apple, and others (see figure 7).

FAIRVIEW is a global wiretapping program, giving access to *cables, routers and switches* of the Internet backbone[33] (see figure 8).

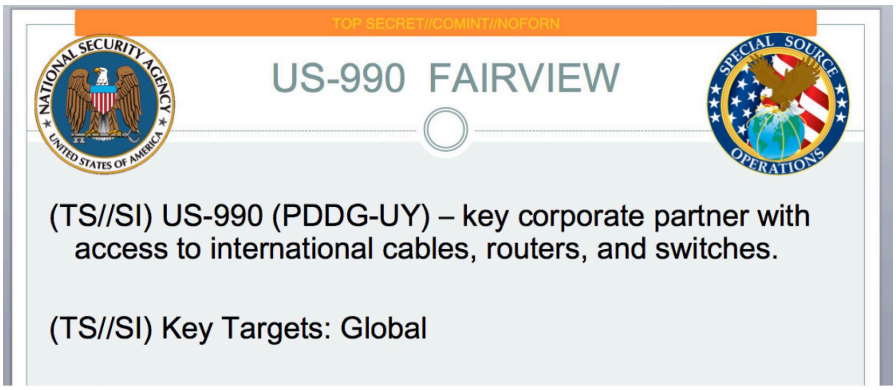

Figure 8: The FAIRVIEW program gives access to Internet backbone systems on a global scale. Image courtesy of NSA.

XKEYSCORE is a system that facilitates easy access to data from various systems. As an example, simply entering an email address and a reason for accessing the system lets the agent view collected emails from said account[33] (see figure 9).

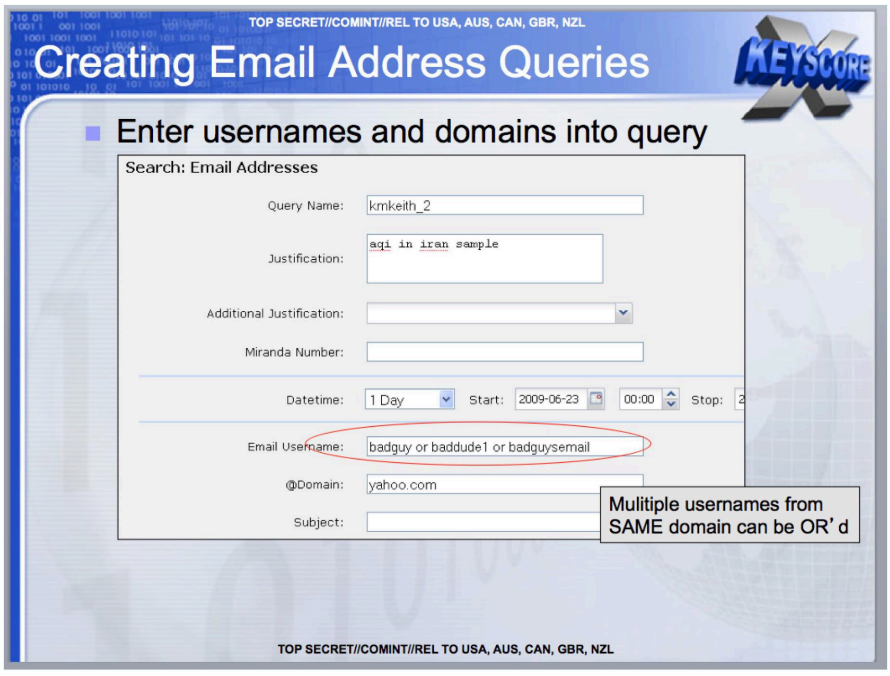

Figure 9: Accessing data with the XKEYSCORE system. Image courtesy of NSA.

BOUNDLESSINFORMANT is a system using *big data* analysis to do near-realtime business intelligence analysis on what amount of metadata is available.

Other activities include intercepting hardware deliveries, such as servers and routers, and install *beacon implants*[33] that subsequently allows the intelligence agencies to access the equipment (see 10).


TOP SECRET//COMINT//NOFORN

June 2010

SID today

(U) Stealthy Techniques Can Crack Some of SIGINT's Hardest Targets

By: (U//FOUO) NAME REDACTED, Chief, Access and Target Development (S3261)

IMAGE REDACTED

(TS//SI//NF) Not all SIGINT tradecraft involves accessing signals and networks from thousands of miles away… In fact, sometimes it is very hands-on (literally!). Here's how it works: shipments of computer network devices (servers, routers, etc.) being delivered to our targets throughout the world are ***intercepted***. Next, they are ***redirected to a secret location*** where Tailored Access Operations/Access Operations (AO – S326) employees, with the support of the Remote Operations Center (S321), enable the ***installation of beacon implants*** directly into our targets' electronic devices. These devices are then re-packaged and ***placed back into transit*** to the original destination. All of this happens with the support of Intelligence Community partners and the technical wizards in TAO.

Figure 10: Intercepting deliveries to install beacon implants. Image courtesy of NSA.

2.1.2 Social Media Censorship

Social media plays a big role in the public media, and much public debate takes place on Facebook, Twitter, and YouTube. The social media platforms are privately owned, and have the freedom the decide the terms of use. In doing so, they sometimes end up censoring material, that is perfectly legal to say in any public setting.

Facebook use image analysis software to censor images containing nudity and other things the general public in U.S.A. finds inappropriate. Thus, when the Norwegian Prime Minister posted the iconic picture of a naked Vietnamese girl running from the napalm flames during the Vietnam War, the picture was removed by Facebook[37]. After much media attention, Facebook decided to accept the picture.

Other people have found themselves *quarentined* from Facebook for exercising freedom of expression, well within the bounds of the law[37].

Twitter is not much different. Right-wing provocateur Milo Yiannopoulos found himself permanently banned from Twitter, after a clash actress Leslie Jones[38]. Milo Yiannopoulos' remarks leading to the ban was well within the bounds of the law.

Social Autopsy[39], headed by Candace Owens, launched a *Kickstarter* campaign[40] in early 2016.

Social Autopsy's mission was to put an end to *cyberbullying* and *trolls* on the Internet, by putting their wrongdoings on public display, like a modern pillory. They put a video on YouTube[41], explaining how this would work. Apparently, screenshots are all that is needed[41, T2:05].

Social Autopsy does mention a review process, there are no details available on this process. I contacted Social Autopsy for information (see figure 11), but I never got an answer.

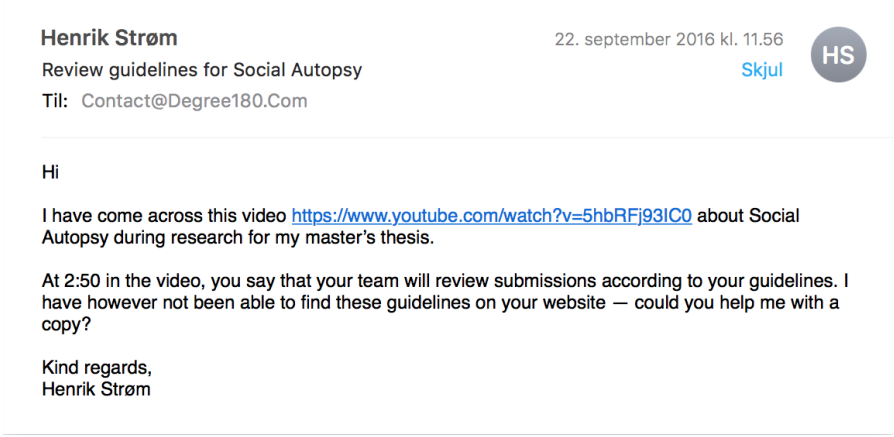

Figure 11: Mail sent to Social Autopsy to obtain a copy of their review guidelines.

It is extremely easy to manipulate screenshots, especially from a browser (see figures 12 and 13), so there is a risk that people could get accused of saying things they never said.

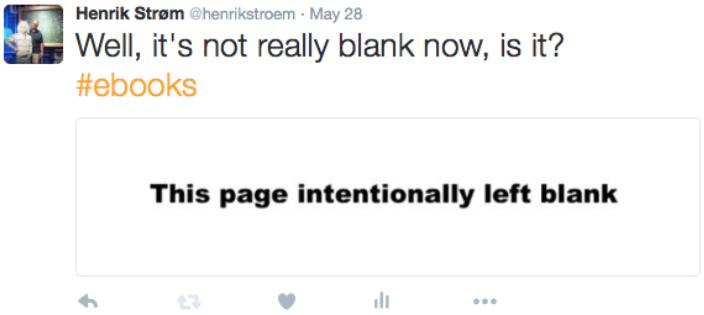

Figure 12: An original post on Twitter.

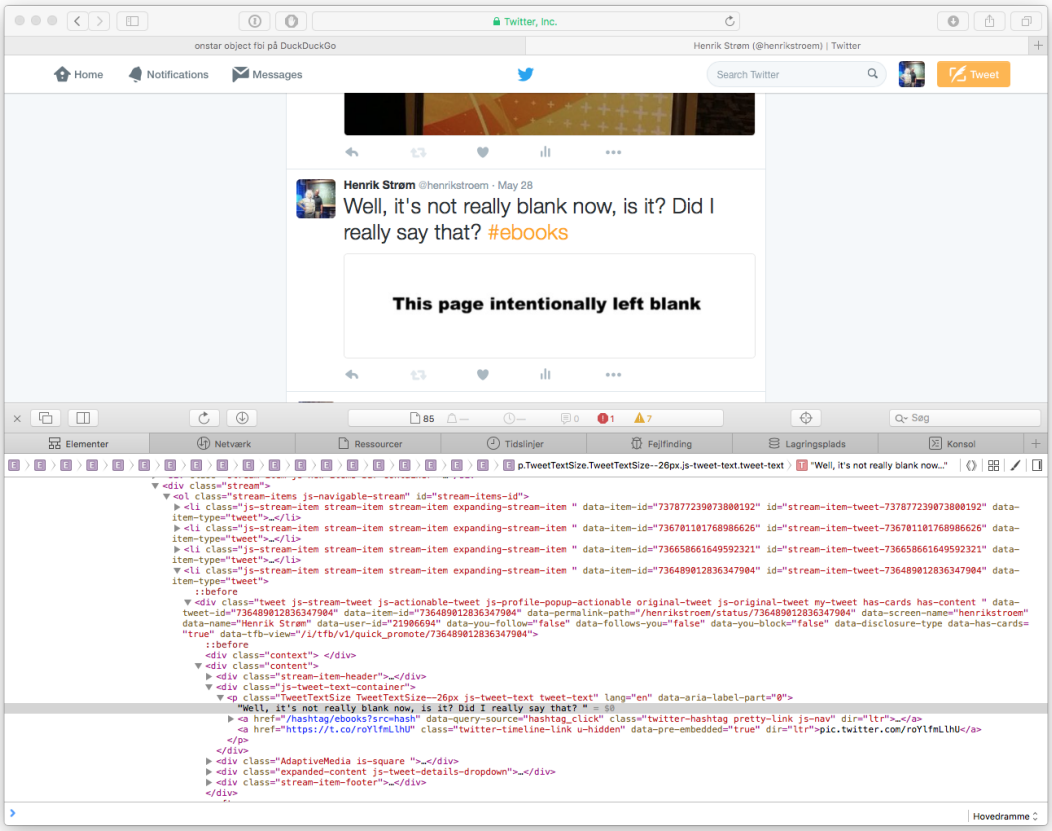

Figure 13: Just by opening the browser inspector, it is possible to change a post on Twitter to say anything you want.

2.2 Message Technologies

2.2.1 Email

Email was one of the earliest kinds of electronic messaging systems. Douglas Engelbart demonstrated a messaging system very similar to modern email already in 1968, in the legendary *Mother of all Demos* [42, T64:23], where users could write messages to each other.

Email is defined by a set of protocols for transmission between mail servers and mail clients, and for then contents of the emails.

Email transmission in its current form, however, dates back to 1981, with the *Simple Mail Transfer Protocol* (SMTP) defined in RFC 788 [43], which has later been succeeded by numerous RFCs refining the protocol, but the basic principles remain the same.

The SMTP protocol, in its original form, was an extremely *naïve* protocol, in that there was no build in authentication, that only came in 1999 with *RFC 2554* [44] – until then, anyone could *relay* an email on a SMTP mail server, and specify the *sender's* email address as any email address whatsoever. SMTP also doesn't have any transport security, but can be combined with SSL/TLS [45], which is usually the case nowadays.

SMTP is capable of both sending and receiving emails and is used exclusively between mail servers on the Internet. Clients usually access their mail servers through *Post Office Protocol* (POP3) [46] or the more advanced*Internet Message Access Protocol* (IMAP4) [47], but still send emails via SMTP. As with SMTP, both POP3 and IMAP4 depend on SSL/TLS for transport security.

Email was originally restricted to sending contents as a one part ASCII message, as described in a long list of RFCs, e.g. *RFC 822* [48], however modern email is send in *Multipurpose Internet Mail Extensions* (MIME) format [49, 50, 51], allowing multi part messages in international character sets (e.g. UTF8), attachments etc.

Like the SMTP protocol, the MIME protocol offers no protection of the contents. To compensate for this, a number of standards have emerged over the years, e.g. *Secure MIME* (S/MIME) [52, p. 518], *Privacy-Enhanced Mail* (PEM) [52, p. 518], *Message Security Protocol* (MSP) [52, p. 519], and *Pretty Good Privacy* (PGP) [52, p. 519], of which S/MIME and PGP is still widely used today.

For many years, email has pretty much been the *default* way of sending messages on the Internet, especially in a professional context. Younger generations, however, tends to adopt communication via *social media* to a higher extent.

Unlike social media, email is not a platform but a technology, or more specifically *a set of protocols*, as we have just seen, thus nobody *owns* email, anybody can buy a domain name, configure the necessary `MX` records in DNS, and set up an email server.

Because email became the most widely used communication technology on the Internet, it obviously attracted some abuse. *Spam* has long been a serious annoyance with email, and without filtering, users' inboxes are most likely filled with far more spam than real emails, however filters have become very effective, and users on many emails services are likely not experiencing much of a problem.

2.2.2 Instant Messaging

Instant messaging comes in many forms, usually, it simply enables users to write short messages to each other, that instantly appear at the recipient. This kind of communication is often referred to as *chatting*.

One of the earlier examples is the Unix *talk* command, which first appeared in 4.2BSD[53], released in 1983[54]. Talk allows users to communicate with any other user on the same host, or users on other hosts using the familiar `user@host` scheme. Combined with the *Secure Shell* (SSH) transport protocol[55], users can communicate directly between two hosts with a very high level of security, only intermediated by a *Domain Name System* (DNS) lookup, which could even be eliminated for security reasons using the *hosts* file, in a pure *peer-to-peer* fashion.

Short Message Service (SMS) or simply *texting*, originally a service protocol used for e.g. alerting about new voice mails, became a popular way of instant messaging in the mid-'90s, as GSM and similar mobile phones became available to most people. SMS is limited to 160 characters, though several messages can be linked together behind the scene.

With the arrival of smartphones, instant messaging started moving to mobile apps, accessing owned platforms such as Signal (Open Whisper Systems), iMessage (Apple), WhatsApp (Facebook), WeChat (Tencent) offering a wide variety of services other than just instant messaging, e.g. payments, voice clips, attached documents or images etc. Some of these services offer strong end-to-end encrypting.

2.2.3 Off-the-Record Communication

Borisov et. al.[56] presents an *off-the-record* protocol primarily suited for instant messaging. This protocol has a number of noteworthy features, such as *perfect forward secrecy*, *reputability* and *forgeability*.

This means that even if the keys are compromised, it is not possible to decrypt or authenticate recorded messages by any third part, or even the participants themselves. This is achieved by negotiating a shared secret using the *Diffie-Hellman* protocol, and repeatedly changing encryption keys.

Use of a *message authentication code* (MAC) enables the participants of the communication to verify the authenticity of the messages, however, this makes it possible to establish proof that *some communication* has taken place between the participants.

The use of a *malleable stream cipher* means that not only is repudiability preserved, the protocol is *forgeable* – if somebody can guess the content of a message, they would be able to craft a message of the same length that would be indistinguishable from a message sent by one of the participants – meaning *anybody could have written the message*, thus the sender can claim *plausible deniability*.

2.2.4 The Tor Network

The Tor network[57] has long been one of the most popular and effective ways of accessing the web and other Internet services, while retaining a good level of anonymity.

Tor offers low latency web browsing[23] on the *ordinary Internet*, and access to special `.onion` sites, which are often referred to as the much hyped *Dark Web*. The Tor Project even offers a customized *Firefox web browser*, with the Tor functionality built in, so make it easier for non-technical users to use the Tor network.

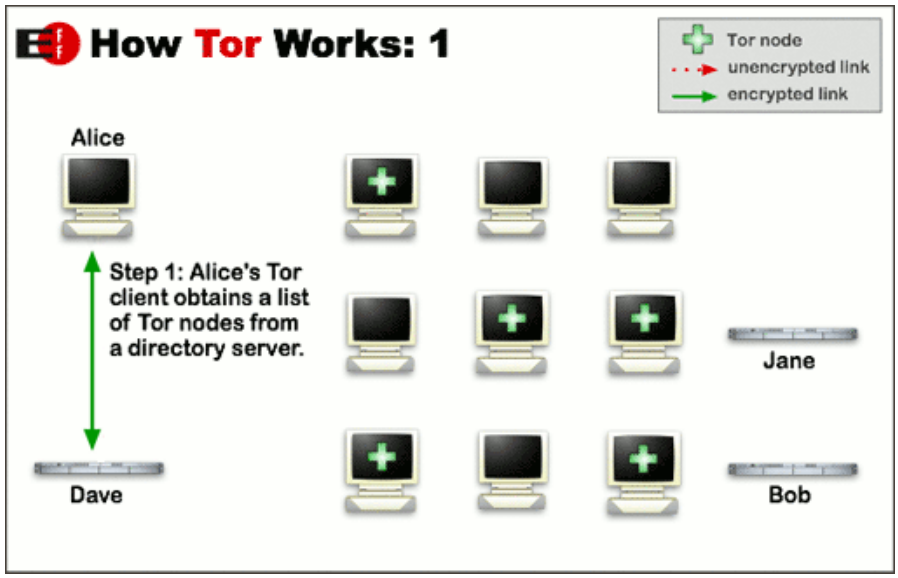

Figure 14: Tor client (Alice) obtaining node list from directory server (Dave). Image courtesy of The Tor Project.

A client connecting to the Tor network initially connects to a *directory server* to obtain a list of potential *nodes* in the Tor network (see figure 14), then selects a random route through the network (see figure 15). This process is known as *onion routing*, as layers of the onion are removed throughout the routing process.

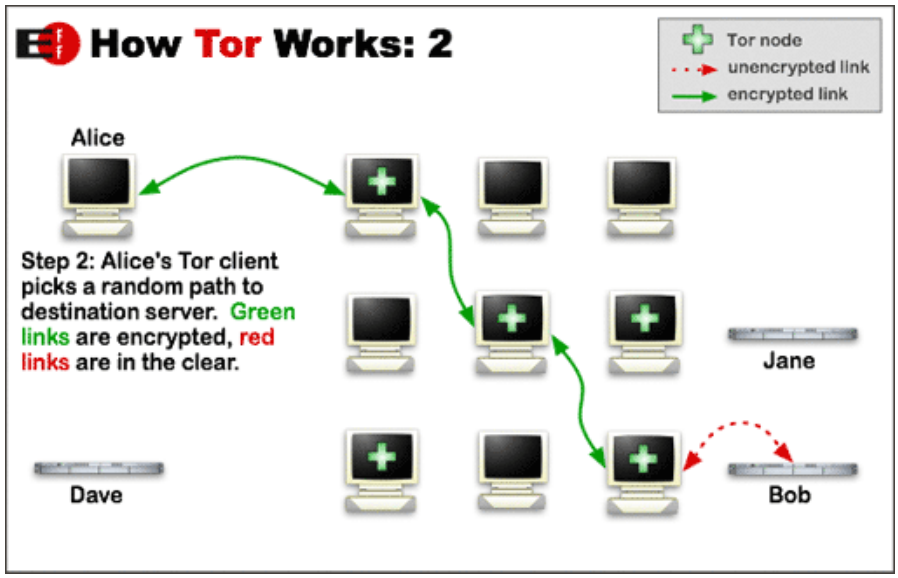

Figure 15: Tor client (Alice) accessing resource a server (Bob) via randomized route. Image courtesy of The Tor Project.

The nodes along the route only have information on the preceding and succeeding nodes in the route, so no single node can know the complete path a data packet has traveled through the Tor network[57].

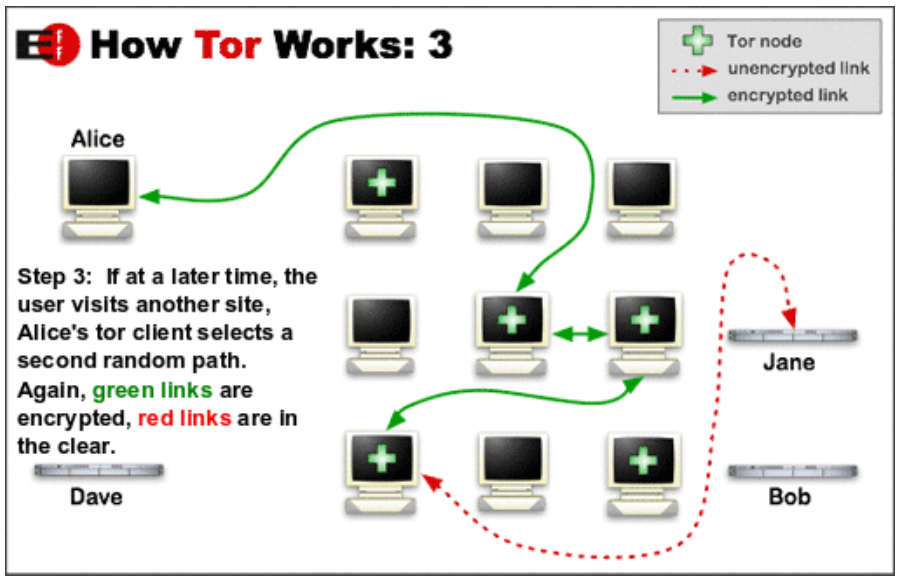

Figure 16: Tor client (Alice) accessing resource at another server (Jane) via a new, randomized route. Image courtesy of The Tor Project.

For efficiency reasons, the same route is used for some period of time, typically around 10 minutes, before a new route is randomized trough the network[57] (see figure 16).

Tor also contains *Hidden Services Directories*, or *HSDirs*, which lists the `.onion` sites mentioned earlier. Sanatinia and Noubir recently gathered evidence, that at least 110 of the approximately 3,000 HSDirs in the Tor network were dishonest, by deploying a number of *honeypot onions* or *HOnions* to detects these dishonest servers[58].

It has been known for some time, that Tor is vulnerable to *traffic analysis*[23] from powerful adversaries such as *governments* and *Internet Service Providers* (ISPs), and even from weaker adversaries analyzing the user's traffic[23]. Methods of traffic analysis include *site fingerprinting* and *general machine learning algorithms*[23].

2.3 Relevant Theory

2.3.1 Primary Security Goals

The primary goals of information security are to ensure *confidentiality*, *integrity*, and *availability*.

Confidentiality is the concept of ensuring that only authorized principals[1] can access information[59, chapter 1]. Access to information in this context includes reading, viewing, printing and in some cases even knowing about the existence of information[59, chapter 1]. This is what people in layman's terms refer to as *privacy* or *secrecy*[59, chapter 1], though privacy actually has a different meaning in information security.

Integrity means that only authorized principals can modify information in an authorized manner[59, chapter 1]. Modification includes writing, changing, changing status (e.g. metadata), deleting and creating information[59, chapter 1].

Availability is about ensuring that authorized principals have access to information[59, chapter 1]. Availability of information includes performance (access in timely manner), future availability (e.g. backups), fault tolerance (e.g. clusters and standby systems) etc[59, chapter 1].

2.3.2 Public Key Cryptography

Public key cryptography makes use of two mathematically related, asymmetric cryptographic keys: a *public key* and a *private key*[52]. The public key can freely

[1] A principal is an entity, e.g. a user or process, that can act on or access information.

be given to anybody, while the private key must be kept secret.

The keys must be used in tandem – a message encrypted with a public key must be decrypted with the corresponding private key, and vice versa.

Asymmetric keys offer *confidentiality*, *authenticity*, *integrity* and *non-repudiation* by combining two users' public and private keys. This is best illustrated using an example:

> Alice needs to send a message to Bob. She will encrypt the message using Bob's public key so that Bob's private key is necessary for decryption, thus making the message *confidential*. She will also encrypt the message using her own private key, making her public key necessary for decryption, thus *authenticating* the message as written by her. Because any tampering with the encrypted message will make the message cryptographically invalid, the *integrity* of the message is ensured, and because she used her own private key, *non-repudiation* is also provided.

If not all these features are needed, the use of the keys can be changed accordingly, and it is also possible to use the private key for *signing* a message, leaving the message in clear text, but providing a *hash* of the message, that can be confirmed using the public key.

Cryptographic keys usually have an expiration date as part of the key itself, limiting the implications of keys being compromised without the owner's knowledge. Another important element of public key cryptography is the ability to *revoke* compromised keys. This requires continuous verification every time somebody else's public key is used, because the key itself does not change when revoked.

Various *public key infrastructures* (PKI) can be used to facilitate the use of public key cryptography, e.g. access to public keys and revocation status, as we shall see in the following sections.

2.3.3 Organizational/Authoritative Trust

Organizational or authoritative trust rests on the foundation that there is some entity in which we have a common trust.

In a global setting, a number of companies have established themselves as *certificate authorities* (CA). They sign various kinds of certificates on behalf of other entities, e.g. web server certificates used with the TLS protocol for secure browsing and other communication, and different levels of hierarchical *root certificates* that can

be used to sign other certificates, e.g. public/private key pairs. *X.509* is an example of such a key[60].

In a commercial setting, we see large enterprise organizations establishing internal *trust centers* or CAs, who can issue public/private key pairs to their employees. They might have an external CA such as *VeriSign* issue a root certificate used to sign certificates subsequently, enabling the use of the certificates with the world outside the organization, or they might use *self-signed* certificates, which only would allow internal use. Companies usually keep copies of employees public and private keys, enabling them to access encrypted information in case the employee was no longer around, however, this means that the *non-repudiation* element of the private key is questionable because the employee is no longer the only person with access to the private key.

We also see national states operate as CAs, e.g. the Danish *NemID* system, facilitating a nationwide system for login authentication and signing of digital documents (electronic signature). In this system, the users do not have the private key themselves, it is stored centrally in the NemID system, making the *non-repudiation* element questionable.

These are examples of authoritative trust – if we trust this authority, we implicitly trust everybody trusted by this authority. This is a pragmatic and simple solution to the obvious *who to trust* problem, but it lacks granularity.

2.3.4 Distributed Trust

An alternative to organizational/authoritative trust is *distributed trust*. This is still based on having mutual trust in "something", but not necessarily the same thing – rather than trusting a commercial organization like Verisign, or a national state, users trust *each other*, creating webs of interconnected trusts. An example:

> Alice and Bob trust each other. Charlie doesn't know Bob, but he trusts Alice, and thus Alice vouch for Bob because Alice trusts Bob – Charlie can implicitly trust Bob.

Pretty Good Privacy (PGP) was developed in 1991 by Phil Zimmermann[60] – a package of cryptographic tools, that can be used to encrypt all kinds of digital information but most notably used for email encryption.

PGP is not based on a central CA that we must all trust but is organized by users generating their own keys, and signing public keys of other users (endorsement) that they know and trust[52, p. 519-520]. This is in contrast to *S/MIME*, that

uses an authoritative trust model for key distribution[59, chapter 7]. PGP doesn't provide any facilitating *infrastructure* however, making the exchange of signed keys cumbersome, even though key servers can be used for key exchange.

However, if Dennis, who trusts Charlie, now needs to trust Bob – can he do that? The answer is *probably not*. It is widely acknowledged that only first-degree trust relationships can be trusted[60].

There is also the problem of new keys[60] – if a user generates a new key, it is yet to be endorsed by other users before it can be trusted, and if users only trust first degree endorsements, many endorsements might need to be made.

Keybase (https://keybase.io) is an attempt to ease the burden of managing keys. It offers a web application and a command line interface to enable users to upload their public key, and *follow* other users they might be interested in exchanging encrypted information with, taking care of the public key handling.

Each user has a profile page on the Keybase homepage (see figure 17) with some basic information about them, but how can you know they are who they say they are?

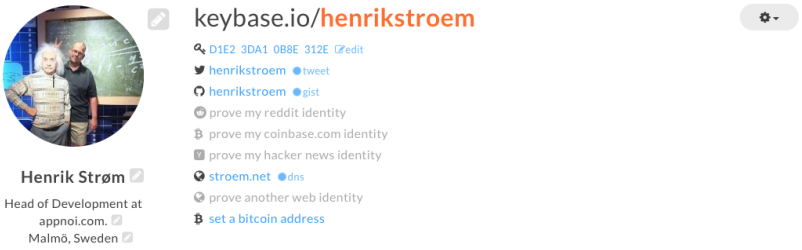

Figure 17: My user profile page on Keybase.

Keybase solves this problem by letting users verify themselves using social media posts, DNS entries, gists on Github, Bitcoin addresses etc. as showed in figure 18. While none of these are perfect for authenticating a user by themselves, they can piece together an identity with a certain level of confidence, and Keybase also assists users in finding the right public keys, by linking them with social network profiles (there might be many public keys belonging to somebody named John Smith).

Figure 18: My verification of my Keybase identity via Twitter.

As pointed out by Wilson and Ateniese[60], Keybase does not currently use the established identities to strengthen the trust in a key, e.g. via blockchain certificates, but this would be of obvious interest.

Users are still required to keep their private key safe and secret themselves, which is still a challenge for many users.

2.4 Candidate Technologies

2.4.1 Riffle

Riffle is an *anonymization technology*, described in the 2016 paper *Riffle – An Efficient Communication System With Strong Anonymity* by Albert Kwan et. al.[23].

A common problem of anonymization technologies is that they have to weigh *anonymity* against *efficiency*, in the form of communication or computation[23, p. 1].

For example:

- the *Tor Network* offers very efficient communication and scales well, but as I demonstrated in section 2.2.4, it is susceptible to traffic analysis,
- *Dining Cryptographer networks* (DC-Nets) offers strong anonymity and protection from traffic analysis, but suffers from low communication efficiency, due to the use of broadcast, that is, bandwidth is proportional to number of clients rather than just size of message[23, p. 1],

- *Verifiable mixnets* also offer strong anonymity and protection from traffic analysis, but suffers from a high computational overhead, due to the use of asymmetric cryptography[23, p. 1].

For these reasons, anonymization technologies usually don't scale well, and they can't support sharing large files.

Riffle is different, however. It was developed with a focus on bandwidth minimization for clients, by not making use of broadcast, and minimizing the computational overhead for servers, by using symmetrical rather than asymmetrical encrypting in the critical path, in what is referred to as *the common case*[23, p. 2].

To achieve this, Riffle introduces two new primitives[23, p. 2]:

- for upstream: a new, verifiable shuffle,
- for downstream: private information retrieval (PIR) based on *anytrust*

Anytrust is a networking model, organized in *a small number of servers* and *a large number of clients*. In the anytrust model, as long as *at least one honest server exists*, two *honest clients* communicating are guaranteed anonymity[23, p. 1].

Because only one honest server is needed to guarantee anonymity, servers are assumed to be operated by different administrative entities, commercial or non-profit. Clients each have a *primary server* that they connect to[23, p. 2].

Unlike the Tor Network, with which you can use the Internet at large, Riffle is not designed for general browsing but designed to exchanged messages between a group of clients.

Communication is carried out in rounds. In each round, each client will send a message to its primary server, whether it has "something to say" or not[23, p. 4], otherwise it would be easy to exclude all clients not having sent any messages in a traffic analysis attempt. Messages are fixed sized, to hinder analysis of message size, so smaller messages must be padded, and larger messages must be sliced into smaller messages[23, p. 4].

The messages must be sent via an *authorized* and *encrypted* channel, such as TLS because Riffle doesn't provide this functionality itself. Confidentiality is also not supported, some other encryption technology must be used for this purpose[23, p. 4].

The full Riffle message protocol is shown in figure 19[23, p. 8].

Algorithm 3 Riffle Protocol

1. **Setup**:
   (a) **Shuffle Keys**:
      i. Each server S_i generates public keys pairs, and publishes the public key p_i to the clients. S_i also generates permutation π_i.
      ii. Each client C_j generates key k_{ij} for S_i, and encrypts them with keys $p_1, \ldots, p_i$ for $i = 1, \ldots, m$. C_j submits m onion-encrypted $\{k_{ij}\}_{i\in[m]}$ to S_1 via the primary server.
      iii. From S_1 to S_m, S_i verifiably decrypts the encrypted keys. S_i then retains $\{k_{ij}\}_{j\in[n]}$, verifiably shuffles the other keys using π_i, and sends the shuffled keys to S_{i+1}. The servers verify the decryption and shuffle.
   (b) **Share Secrets**: Every pair of S_i and C_j generates pairwise secrets m_{ij} and s_{ij}, used for PIR (Algorithm 2) in the download stage.
2. **Communication**: In round r,
   (a) **Upload**: C_j onion-encrypts the message M_j^r using authenticated encryption with $\{k_{ij}\}_{i\in[m]}$ and r as a nonce: $AEnc_{1,\ldots,m}(M_j^r) = AEnc_{k_{1j},r}(\ldots(AEnc_{k_{mj},r}(M_j^r))\ldots)$. C_j then sends $AEnc_{1,\ldots,m}(M_j^r)$ to S_1 via C_j's primary server.
   (b) **Shuffle**: From S_1 to S_m, S_i authenticates, decrypts, and shuffles ciphertexts using the π_i, and sends the shuffled $\{AEnc_{i+1,\ldots,m}(M_j^r)\}_{j\in[n]}$ to S_{i+1}. S_m shares the final plaintext messages with the all servers.
   (c) **Download**: The clients download the plaintext message(s) through PIR or broadcast.

Figure 19: The Riffle message protocol. Image courtesy of Albert Kwan et. al.

File sharing in Riffle is based on *BitTorrent*[61], however it works as a *client-server* setup rather than *peer-to-peer*, with the servers acting as *torrent trackers*[23, p. 12]. Clients sharing a file generates a torrent file, a small file containing hashes of all the file blocks the shared file is divided into[23, p. 12]. The upstream and downstream process is a performance optimized variation of the Riffle message protocol.

The file sharing in Riffle is anonymous, but if the files are confidential, they need to be encrypted before the torrenting process is started, by some cryptographic tool.

For a technology offering strong anonymity, Riffle is quite efficient. For file sharing, Riffle tests show a 100KB/s per user for file sharing in a group of 200 users, and in the messaging scenario (called *microblogging*), the system could handle 100,000 users with less than 10 seconds latency.

2.4.2 PGP

PGP has already been discussed in section 2.3.4. I must just add that PGP is a candidate technology, as it will enable strong encryption without trust in a central authority, that could be compromised e.g. by government surveillance.

3 Analysis

3.1 General, Generative Message System

We can begin to see some features that our technology should support:

- Everybody should be able to join the system – this is a fundamental feature of a system supporting freedom of expression
- Everybody should be able to set up their own primary server, and join the global system, just as you can with email servers – you might not trust other operators
- The system should be generative, in that the system should be open to new technology being build upon it, in perfect harmony with the *end-to-end argument* discussed in section 1.1
- It should be a *general communication technology*, meaning users can perform the same high-level tasks one would expect from any communication system, e.g. send, receive messages to and from any user

The generative element is very important. Imagine giving some kids each a set of the same LEGO bricks, and ask them to build a house. Given no other instructions, the kids would most likely build very different houses, some with features you might not have imagined. Building generative technologies creates open, thriving technologies, bigger than the sum of their parts.

Of course, sometimes decisions must be made that will limit generativity, e.g. it is very unlikely that a technology such as this could support *video streaming*, and it is perfectly ok that such functionality is cut off in order to achieve the primary objectives.

3.2 Identified, Pseudonymous, and Anonymous

The user stories in section 1.3 showed that we need three levels of *identifiability*:

1. *Identified* – an identified user is a user presented by his real name, we can send messages to him if we know his name, and we can link messages to his account
2. *Pseudonymous* – a pseudonymous user is a presented by a pseudonym the user has chosen, rather than his real name, but we can still send messages to him, and link messages from the same pseudonym user together

3. *Anonymous* – an anonymous user need not be presented as anything but *Anonymous* because we can't reply to his messages, and we can't link his messages together

The difference between *psedonymous* and *anonymous* users may seem subtle, so why bother to differ between them? The answer is, that they offer some different but important properties.

Anonymous accounts are *one off* accounts – you only use an anonymous account one time, and the private key is immediately destroyed. This means, even if the user's computer is succeedingly compromised, the user can't be linked to the anonymous message. This offers a high level of refutability, however, its use is limited, because the user can't sustain a dialogue, and other users can't reply to an anonymous message.

Pseudonymous accounts are still not linked to any given person, and allows to sustain a dialog and reply to messages. However, because the private key is saved, the user can be linked to the messages.

3.3 Use Cases

In section 1.3 we went through user stories showing a typical use of a communication system supporting elements of confidentiality, integrity, anonymity, and pseudonymity. We will now go through the corresponding use cases. The use cases are based on Andrew Stellman's model[22].

3.3.1 UC-1: Send Private Message

The user story *Send private message* from section 1.3 showed that Alice could send a private message to Bob. The corresponding use case goes as follows:

Name	UC-1: Send private message
Summary	Alice sends Bob a message that only Bob can read
Rationale	When users are concerned with 3rd party eavesdropping, encryption can be used to ensure confidentiality of the message
Users	All users
Preconditions	1. Alice and Bob are both registered users and

Name	UC-1: Send private message
	2. Alice and Bob's public keys are published
	3. Alice knows Bob's public username
Basic Course	1. Alice selects the private message function
of Events	2. Alice selects Bob as recipient
	3. Alice types her message to Bob
	4. Alice types her pass phrase
	5. The message is sent encrypted to Bob
	6. Bob receives the message
	7. Bob enters his pass phrase,
	and can read the message
Alternative Paths	3a. Alice selects files to be attached
Postconditions	1. Bob - and only Bob - can read the message

Table 1: UC-1: Send private message use case

3.3.2 UC-2: Send Public Message

The use case for the user story *Send public message* from section 1.3:

Name	UC-2: Send public message
Summary	Bob sends a message with his public username, signing the message so everybody can confirm it's Bob who wrote the message
Rationale	With services such as Social Autopsy (see section 2.1.2), users might want to protect themself from misrepresentation by signing messages
Users	All users

Name	UC-2: Send public message
Preconditions	1. Bob is a registered user
	2. Bob's public key has been published
Basic Course	1. Bob selects the public message function
of Events	2. Bob types the message
	3. Bob types his pass phrase
	4. The message is sent with Bob's signature
Alternative Paths	2a. Bob selects files to be attached
Postconditions	1. Everybody can confirm the message was written by Bob

Table 2: UC-2: Send public message as self

3.3.3 UC-3: Send private message anonymously

The use case for the user story *Send private message anonymously* from section 1.3:

Name	UC-3: Send private message anonymously
Summary	Alice sends a private message anonymously, so that nobody can tell that it was Alice who sent the message, and so that only Bob can read it
Rationale	A user might want to conceal her identity, until some agreement has been reached with the recipient, to protect herself
Users	All users
Preconditions	None
Basic Course	1. Alice selects the anonymous private message
of Events	function
	2. Alice types the message

Name	UC-3: Send private message anonymously
	3. The message is sent anonymously
	4. The anonymous public key is published
	5. Bob receives the message, he can read the message, but not determine Alice's identity
Alternative Paths	2a. Alice selects files to be attached
Postconditions	1. Nobody can tell the message was written by Alice, and only Bob can read the message
	2. The anonymous cryptographic keys are deleted

Table 3: UC-3: Send public message anonymously

3.3.4 UC-4: Send Public Message Anonymously

The use case for the user story *Send public message anonymously* from section 1.3:

Name	UC-4: Send public message anonymously
Summary	Alice sends public a message anonymously so that nobody can tell that it was Alice who sent the message
Rationale	Users might want to state an unpopular or controversial opinion, but be afraid of reprisals - sending the messages anonymously will protect people from such reprisals
Users	All users
Preconditions	None
Basic Course of Events	1. Alice selects the anonymous public message function
	2. Alice types the message

Name	UC-4: Send public message anonymously
	3. The message is sent anonymously
Alternative Paths	2a. Alice selects files to be attached
Postconditions	1. Nobody can tell the message was written by Alice
	2. The anonymous cryptographic keys are deleted

Table 4: UC-4: Send public message anonymously

3.3.5 UC-5: Send Private Message Pseudonymously

Name	UC-5: Send Private Message Pseudonymously
Summary	Alice pseudonymously sends a private message to Bob, so that Alice can conceal her identity, but Bob can still link messages from Alice's pseudonym together, and send messages back to Alice
Rationale	Alice wants to have a dialogue with Bob, while concealing her identity
Users	All
Preconditions	1. Alice has registered a pseudonym user 2. The pseudonym user's public key has been published 3. Alice knows Bob's user name
Basic Course of Events	1. Alice selects the pseudonymous public message function 2. Alice types the message 3. Alice types her pass phrase 4. The message is sent pseudonymously 5. Bob receives the message, he can read the

Name	UC-5: Send Private Message Pseudonymously
	message, but not determine Alice's identity, however, he can link the message with the pseudonym
Alternative Paths	2a. Alice selects files to be attached
Postconditions	1. Bob can't tell the message was written by Alice, but he can link the message to the pseudonym

Table 5: UC-5: Send Private Message Pseudonymously

3.3.6 UC-6: Send Public Message Pseudonymously

Name	UC-6: Send Public Message Pseudonymously
Summary	Alice wants to state an opinion without being identified, but still enable users to reply and link messages to the pseudonym
Rationale	Create a public alter ego
Users	All
Preconditions	1. Alice has registered a pseudonym user 2. The pseudonym user's public key has been published 3. Alice knows Bob's user name
Basic Course of Events	1. Alice selects the pseudonymous public message function 2. Alice types the message 3. Alice types her pass phrase 4. The message is sent pseudonymously 5. Bob receives the message, he can read the message, but not determine Alice's identity,

Name	UC-6: Send Public Message Pseudonymously
	however, he can link the message with the pseudonym
Alternative Paths	2a. Alice selects files to be attached
Postconditions	1. Nobody can tell the message was written by Alice, but they can link the message to the pseudonym

Table 6: UC-6: Send Public Message Pseudonymously

3.4 Stakeholders

> *"If you're not paying for something, you're not the customer, you're the product being sold." – Andrew Lewis*[58, chapter 1].

Users have come to expect most Internet services to be free. However, running a social media platform is expensive, and companies obviously do it make money, rather than just making users happy. The platform owners will operate the platform in a way that produces the biggest profit.

Social networks like YouTube, Twitter, and Facebook are based on a *multisided platform business model*[62]. These platform owners basically run *advertising network* – users get free content, but they pay by being exposed to targeted advertising. While some content creators work on an amateur basis or receive donations from *Patreon*[63], some are professional, and are paid by the platform owner when users consume their content, and thereby are exposed to advertising (see figure 20).

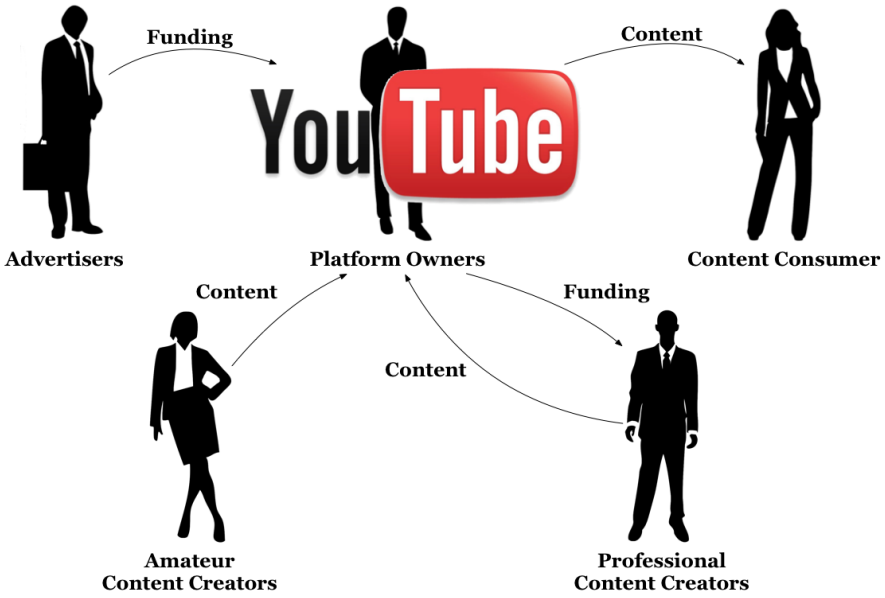

Figure 20: Most social media platforms are multisided platforms, funded by advertisement.

As companies, the platform owners have the freedom to decide on the terms of the platform, as long as they operate within the law. The authorities can hand the company a *National Security Letter* or similar, and the platform owner will have to cooperate.

For freedom of expression not to be suppressed on a truly democratic platform, there must be no platform owner and no commercial interest. This way, there is no entity to hand a National Security Letter to, and the platform can be operated with freedom of expression in mind, rather than commercial interests.

3.5 Legal Issues – Interview with Lawyer

As shown in section 3.4, our Independent Communication System has *users* who use the system, and *server operators* who run the servers the system is based on.

Both these groups might have concerns about the legality of the system, and what consequences it might have for them to participate. This obviously depends on where these people live – living in a Western democracy would be very different from living in a totalitarian or authoritarian regime.

It is also interesting if it would be considered illegal to run such a system, and the system could be forced to shut down. In this case, it would obviously be a waste of

time constructing such a system, at least from a technical perspective.

To get an insight to the legality of the Independent Communication System, I had an interview with Henrik Udsen[64], a professor at and head of *Center for Informations- og Innovationsret* at Copenhagen University in Denmark. While Henrik Udsen's expertise is Danish law, this insight is still strongly indicative of the law in other Western democracies. For obvious reasons, I have not been able to conduct a similar interview with a lawyer from other kinds of societies.

The interview took place on August 2nd, 2016 at Henrik Udsen's office. After a short introduction to the concept, I asked whether users and server operators participating in the network could be held responsible:

> *[T10:03] In general, no – they can't.*
>
> *[T11:21] In order to be liable for criminal action, you would have to have willingness to do crime, so to speak. And you can also be liable in the way that you contribute to the criminal act of another person, so if you drive the car and somebody go in and rob the bank, you can be liable as well. But still you would have to know that that other person actually did something that was criminal and intended to do so. And that you intended to assist in that. And that's not the case in your example here, because I don't know what's on my computer, I don't know the content of these messages because they are encrypted, I can't decrypt them, I assume. So you don't have any intent, as a starting point.*

As we can see, participating in the Idenpendent Communication System would not be considered a criminal act. The keyword here is *intent* – in order to be liable for a criminal act, you have to take part *actively* and *knowingly*. Merely unknowingly facilitating a crime is not a criminal act.

Another issue is whether you would have to react if you as a server operator somehow found out that the system contained illegal material, and this information was stored on your server.

> *[T14:28] When you are actually informed that there might be something illegal on your server, then you might have an obligation to react on that, to remove the illegal content. It might be different if you are operating a system, and you are kind of encouraging people to use it in an illegal way. Pirate Bay for instance, they encouraged their users to share files illegally. If you do things like that, obviously you can become liable. But the fact that you are just providing space for users is not enough to be liable.*

> *[T16:36] As a starting point you can not, as I explained [sic.] the other person [sic.] is you can not, if the police comes, it could also be a court order (..) say you have to delete this, give us some information or whatever. And then of course you would be responsible for complying with that request, according to the rules. But if it's not possible, because the way the system is set up technically it's not possible, you can't be held liable for not doing something you are actually not able to do.*

One would of course still be able to remove material by taking down the entire system, or at least the server operator's part of the system.

> *[T17:16] Eh – yeah, perhaps. I think it depends on how the system works. If you have a server there, and you can see – I don't know how you should be able to do that because you said the whole point is you can't get access to the information, but even if you could (..) you can't read the content, but even if you could, and you found out that it's primarily illegal content which is on this server, so the server you provide is almost only hosting access to illegal materials – maybe, I am not sure.*

So in the case of Pirate Bay, it was almost exclusively being used for copyright infringing materials, and because they actively encouraged users to use Pirate Bay for this purpose. Also, the name left little doubt what their intentions were.

> *[T18:30] Yeah I think it would depend on, you know, how much legal materials are there, and how illegal. So the fact that you found three legal files is not very interesting. But again, it wouldn't necessarily be enough that there's a lot of illegal materials, because in the Pirate Bay case you know, they kind of contributed themselves, and encouraged people to do these kinds of things. Because we all know that there's a lot of illegal stuff on YouTube, but that wouldn't mean that you would be able to close down YouTube, in general.*

In the case of illegal materials on YouTube, the rights holder would be able to file a DMCA, and YouTube would be responsible for removing the content.

> *[T19:15] Yes. And you would, in that system as well [the Independent Communication System] .. message number 4 is illegal, it's stored on your server and it contains child pornography or whatever, obviously*

> *you would be able to remove it, or you would be obligated to remove it, or you could be. That's not the same as saying, well because there's a lot of illegal material on your server, so now we close down the whole server.*

If the system made it impossible for the server operator to remove the material, or the material would simply come back from other servers during synchronization, the server operator would be in a situation that he would not be able to comply.

> *[T19:57] In that situation I don't think you would have an obligation, again as I said, if it's technically not feasible you couldn't put a legal obligation on anybody doing something that is technically not possible.*

Demonstrating that the system was primarily being used for illegal purposes would probably not be enough to have the system shut down.

> *[T20:20] Yeah, and I'm not sure that would be enough, one thing is that it is actually used in a way like that, but was that also the purpose? You know – I think if you have some people behind the system, and the whole purpose was actually to make a network or system that would allow ISIS to communicate, that system would be shut down one way or another, that's law as well, you know what is common sense here. But if it's a system which is actually a general system, which is made for people you know, which is discussing political issues where it's not allowed to do so or whatever, I don't think you would shut down a system like that, when we talk Denmark, you know.*

3.6 Requirement Specification

We can now present a requirement specification for an implementation. Please note, that the requirement specification priorities are based on *an ideal solution*. This means, some requirement might be prioritized as a *must*, but it might not be possible to implement such system with the technology available today. In chapter 4 (Proposed Solution) I will point out which requirements can be met.

The fields in the requirement specification table are described in the table below.

Field	Values	Description
ID	Serial	Unique ID identifying the requirement

Field	Values	Description
F/N	F or N	Indication of functional/non-functional requirement
Origin	Reference	Reference to requirement origin, e.g. section
Priority	M, S, C or W	Priority according to MoSCoW (Must have, Should have, Could have, Won't have)
Title	Name	Title of the requirement

Table 7: Requirement Specification fields explained

For reasons of compactness and readability, the requirements are each explained following the table. The order of the requirements does not indicate priority.

ID	F/N	Origin	Priority	Title
R00	F	3.1	M	Support generativity
R01	F	2.4.1	M	Upstream transport anonymity
R02	F	2.4.1	M	Downstream transport anonymity
R03	F	1.3	M	Anonymous posting
R04	F	1.3	M	Pseudonymous posting
R05	F	1.3	M	Identified posting
R06	F	1.3	M	Confidentiality
R07	F	1.3	M	Integrity
R08	F	3.1	M	Send message to any user
R09	F	1.3	M	Send public message
R10	F	1.3	M	Send private message
R11	F	1.3	S	Share file public
R12	F	1.3	S	Share file private
R13	F	3.1	M	Receive private message
R14	F	3.1	M	Read public messages
R15	F	3.1	W	Receive video streaming

ID	F/N	Origin	Priority	Title
R16	F	4.2	S	Suppress spam messages
R17	F	2.4.1	M	Strong protection from traffic analysis
R18	F	3.1	M	Enable anybody to become a client
R19	F	3.1	S	Enable anybody to run a server

Table 8: Requirement Specification

R00 – Support of generativity means the technology can potentially be used as a basis for other technologies we have not yet thought of.

R01 – Upstream transport must be anonymized, so users posting can't be linked to individual posts by analyzing the transport stream.

R02 – Downstream transport must be anonymized, so users can't be linked to individual posts they retrieve, by analyzing the transport stream.

R03 – Users must be able to post messages using anonymous, one-off accounts, that are truly anonymous, and deleted immediately after posting a message.

R04 – Users must be able to post messages under pseudonyms, which can be replied to, and messages from the same pseudonym can be linked to each other, but not to the real user

R05 – Users must be able to post messages as an identified user, which can be replied to, and messages from the same identified user can be linked to each other, and to the real user

R06 – Messages can be encrypted to support confidentiality, so only the intended user can decrypt and read the messages

R07 – Messages can be signed, so messages can't be altered without mismatching signatures

R08 – Any user can send a message to any user he knows the identity of (identified or pseudonym)

R09 – Users can send public messages for all to see

R10 – Users can send private messages to be received only by the intended user

R11 – Users can share files with the public

R12 – Users can share files privately with selected users

R13 – Users can connect to their primary server, and collect private messages

R14 – Users can connect to their primary server, and collect public messages

R15 – Users can stream videos from their primary server

R16 – The system must be able to mitigate the problem of spam messages

R17 – The system must offer strong protection from traffic analysis

R18 – All people must be able to connect to the system and become a user

R19 – All people must be able to set up their own primary server and join the system

4 Proposed Solution

In this chapter, I will briefly discuss some of the most important elements of a proposed solution. Some of the requirements listed in the requirement specification (see section 3.6) raises some problems, and I will also discuss these problems here.

4.1 Number of Servers

In section 3.1, I argued that everybody should be able to set up their own primary server and join the global system, just as it is possible with email servers. This is based on a democratic approach.

From a technical perspective, however, this would probably not work, as long as the system is based on Riffle. Because all servers take part in the verifiable shuffle permutation, a large number of servers would make the system inefficient.

It helps that you actually don't have to trust your primary server – as long as just one server is honest, you'll be fine.

4.2 Spam

As I showed in section 2.2.1, spam has turned out to be a major problem in email. It's very simple: because there is virtually no cost of sending an email, sending a flood of emails can be a viable marketing strategy, even if just a small number of the recipients respond.

While many products and services are available to mitigate these issues in the domain of emails, they would not work with encrypted messages. Also, it is extremely limited what data we can associate with any user, given that the whole basis of our solution is to offer various degrees of anonymity.

Given that some stakeholders would rather not have a solution like this one around, they could simply flood the system with spam messages to the point that it would become intolerable. So this is an important problem to solve.

The spam problem would be a problem for us on two levels: first, like with emails, users will be annoyed by spam messages – second, because this technology still has a much higher level of computation related to message handling than emails servers, spam could have a significant impact on the performance of the system.

I will point to two possible solutions to the spam problem.

First of all, a *computational cost* for sending a message. This could be presented as a *challenge*: before accepting a message, the receiving server would send this

challenge to the client, that would have to spend a not insignificant amount of time on solving the challenge before the server would accept the message. Obviously, this should be constructed with very little overhead for the server.

Second, we could present the user with a *human-only* challenge, i.e. a challenge that is easy to solve for a human being, but difficult for computers to solve, e.g. type the text from a skewed image as in a *captcha* (see figure 21). This will, however, limit how the system can be used in a generative manner, e.g. as an underlying transport system, because human intervention always would be necessary.

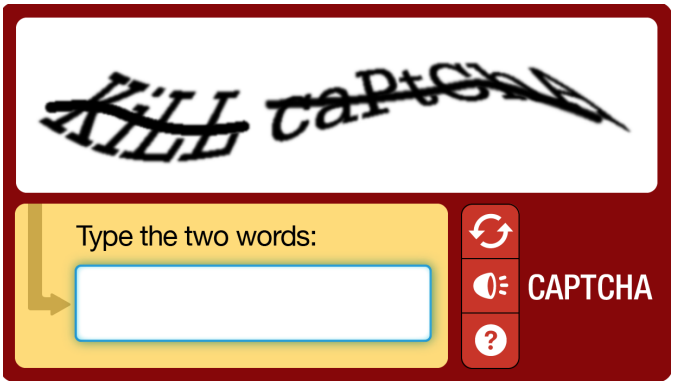

Figure 21: Capcha presenting the user with a challenge. This should be easy for a human being to solve, but hard for a computer.

These solutions could be used in combination, and both would make it significantly harder to spam the system, but spam would most likely be an issue that would need regular attention.

4.3 Handling Users, Identities and Keys

A distributed trust model, as discussed in section 2.3.4, relieves us from having to trust certain entities. Because our adversaries might be powerful governmental surveillance programs (see section 2.1.1), we can't trust a central authority.

We need a way to handle *users*, *identities* and *keys*, and even general identifiers (e.g. for messages). Because the client can't trust his primary server, the identifiers must be created by the client, but no two clients must ever come up with the same identifier. This is, however, a common problem, and using a `UUID` or *Universal Unique Identifier* is an obvious solution, as these can easily be generated on any client, with negligible risk of collision. So – *everything is a UUID*.

Using PGP, clients can generate their own public/private key pairs for encrypting, decrypting, signing and verifying messages. The public keys can be uploaded to the primary server and synchronized between the servers.

We don't need a central concept of *users*, because we actually don't want to link identities in the system together. Owning an identity is thus simply a matter of processing the private key and associated pass phrase, and it is completely up to the user to handle these keys. This also means, if a user loses his keys, he will no longer have access to that identity.

4.4 Technology Stack

We now arrive at a technology stack based on Riffle, PGP and some custom *identity handling* (see figure 22).

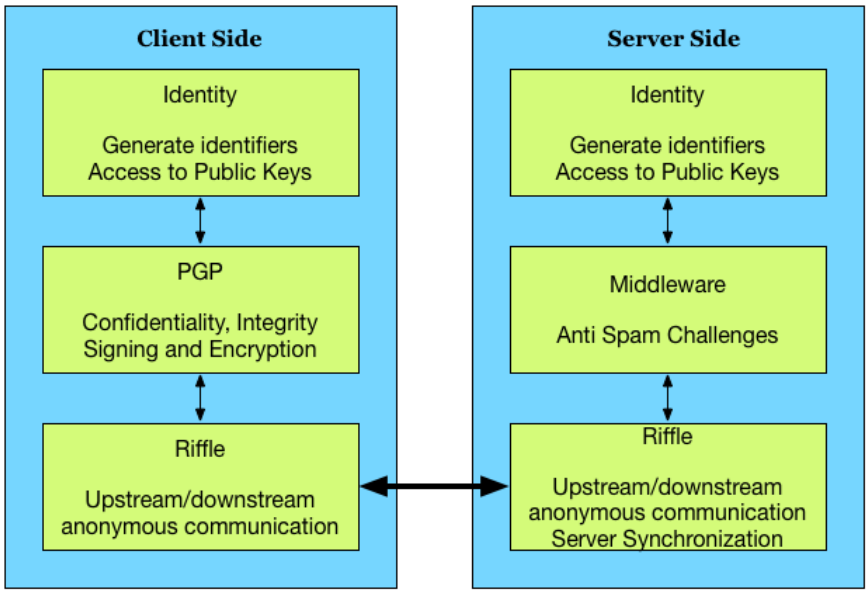

Figure 22: Solution technology stack.

Riffle is providing anonymous transport for all communication. On the server side, a middleware layer allows for modules to be added, e.g. a challenge system as discussed in section 4.2 to mitigate spam. The server side is not concerned with the payload, and thus need not implement any PGP functionality.

5 Conclusion

5.1 Answer to Research Question

Re-stating the research question:

> ***How can existing communication technologies be combined and enhanced into a generative technology, supporting freedom of expression?***

I have demonstrated a proposed solution, built on the same fundamental principle as the Internet itself: the *end-to-end principle* (see section 1.1).

By applying this principle, I arrived at a *generative technology*, that can *support freedom of expression* (see chapter 4), by giving the user as much control as possible (see section 4.3).

By not relying on an *authoritative trust model*, and using *state of the art anonymization technology*[23], the technology can protect users from very powerful adversaries, such as governmental surveillance programs (see section 2.1.1).

5.2 Contribution to Research Domain

The main contribution to the research domain from this project is the suggestion of a *full stack, generative solution*. It is made clear, that such a solution is not only plausible, but indeed realizable with the technology at hand.

The realization of the suggested technology and a global implementation would help support freedom of expression in the world at large.

5.3 Recommendations for Future Work

The most significant limitation of the suggested technology is the groups of servers as defined by Riffle[23].

While groups can support 100,000s of users, this is not enough on a global scale. I would suggest research into bridging groups of servers, enabling users to message any user in any other group.

Careful research must obviously be done into the impact on the security objectives of the technology if such a solution was to be implemented.

6 References

[1]D. Clark, "The design philosophy of the DARPA internet protocols," *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4, pp. 106–114, 1988.

[2]R. T. Griffiths, "History of the Internet, Internet for Historians," *Universiteit leiden homepage*. 2002 [Online]. Available: http://www.let.leidenuniv.nl/history/ivh/chap2.htm. [Accessed: 22-Jun-2016]

[3]J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems*, vol. 2, no. 4, pp. 277–288, 1984.

[4]J. Zittrain, *The future of the Internet and how to stop it*, vol. 321. 2008, p. 342 [Online]. Available: http://ssrn.com/paper=1125949

[5]T. J. Berners-Lee, "The world-wide web," *Computer networks and ISDN systems*, vol. 25, no. 4, pp. 454–459, 1992.

[6]R. Bhatia, "The inside story of Facebook's biggest setback," *The guardian*. 2016 [Online]. Available: https://www.theguardian.com/technology/2016/may/12/facebook-free-basics-india-zuckerberg?utm_source=pocket&utm_medium=email&utm_campaign=pockethits. [Accessed: 20-May-2016]

[7]E. Ofek and M. Richardson, "Dotcom mania: The rise and fall of internet stock prices," *The Journal of Finance*, vol. 58, no. 3, pp. 1113–1137, 2003.

[8]O'reillyT., "What is Web 2.0: Design patterns and business models for the next generation of software," *Communications \& strategies*, no. 1, p. 17, 2007.

[9]"GOOGLE ACCUSED OF 'ABUSIVE' CONDUCT IN PRIVACY APP CASE," *The intercept*. 2015 [Online]. Available: https://theintercept.com/2015/06/23/google-disconnect-antitrust-privacy-app-europe/. [Accessed: 20-Jul-2016]

[10]T. Berners-Lee, J. Hendler, O. Lassila, and Others, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[11]"A second spring of cleaning," *Google official blog*. 2013 [Online]. Available: http://googleblog.blogspot.dk/2013/03/a-second-spring-of-cleaning.html. [Accessed: 01-Sep-2016]

[12]"Why did Google kill Google Reader?" *Quora*. 2013 [Online]. Available: https://www.quora.com/Why-did-Google-kill-Google-Reader?share=1

[13]E. Pariser, *The filter bubble: What the Internet is hiding from you*. Penguin UK, 2011.

[14]O. B. B. Orwell, "George. 1984." Secker; Warburg London, 1949.

[15]G. Greenwald, *No Place To Hide*. 2014, p. 259.

[16]"Mission & Strategy - NSA.gov," *NSA homepage*. 2016 [Online]. Available: https://www.nsa.gov/about/mission-strategy/. [Accessed: 20-Jun-2016]

[17]"UK security agencies unlawfully collected data for 17 years, court rules," *The guardian*. 2016 [Online]. Available: https://www.theguardian.com/world/2016/oct/17/uk-security-agencies-unlawfully-collected-data-for-decade. [Accessed: 16-Oct-2016]

[18]E. Stoycheff, "Under Surveillance : Examining Facebook's Spiral of Silence Effects in the Wake of NSA Internet Monitoring," *Journalism & Mass Communication Quarterly*, vol. 1, no. 1, pp. 1–16, 2016 [Online]. Available: http://jmq.sagepub.com/content/early/2016/02/25/1077699016630255.full.pdf

[19]R. McKie, "Shamed Nobel laureate Tim Hunt 'ruined by rush to judgment after stupid remarks'," *The guardian*. 2015 [Online]. Available: https://www.theguardian.com/science/2015/jun/13/tim-hunt-forced-to-resign?CMP=share_btn_tw. [Accessed: 13-Jun-2015]

[20]"Twitter Trust and Safety Council." 2016 [Online]. Available: https://about.twitter.com/safety/council. [Accessed: 17-May-2016]

[21]"Fair Use - DMCA.com," *DMCA.com*. 2016.

[22]A. Stellman, "Requirements 101: User Stories vs. Use Cases." 2009 [Online]. Available: http://www.stellman-greene.com/2009/05/03/requirements-101-user-stories-vs-use-cases/. [Accessed: 20-Jun-2016]

[23]A. Kwon, D. Lazar, S. Devadas, and B. Ford, "Riffle: An efficient communication system with strong anonymity," in *Proceedings on privacy enhancing technologies*, 2016, vol. 2016, pp. 115–134.

[24]"Compromise needed on smartphone encryption," *The washington post*. 2014 [Online]. Available: https://www.washingtonpost.com/opinions/compromise-needed-on-smartphone-encryption/2014/10/03/96680bf8-4a77-11e4-891d-713f052086a0_story.html?utm_term=.80a6c8a8a3f4. [Accessed: 03-Oct-2014]

[25]C. Coyne, "The Horror of a 'Secure Golden Key'," *Keybase blog*. 2014 [Online]. Available: https://keybase.io/blog/2014-10-08/the-horror-of-a-secure-golden-key. [Accessed: 08-Oct-2014]

[26]D. Goodin, "Cisco confirms NSA-linked zeroday targeted its firewalls for years," *Ars technica*. 2016 [Online]. Available: http://arstechnica.com/security/2016/08/cisco-confirms-nsa-linked-zeroday-targeted-its-firewalls-for-years/. [Accessed: 18-Aug-2016]

[27]D. Goodin, "No TitleNSA-linked Cisco exploit poses bigger threat than previously thought," *Ars technica*. 2016 [Online]. Available: http://arstechnica.com/security/2016/08/nsa-linked-cisco-exploit-poses-bigger-threat-than-previously-thought/. [Accessed: 23-Aug-2016]

[28]S. Biddle, "THE NSA LEAK IS REAL, SNOWDEN DOCUMENTS CONFIRM," *The intercept*. 2016 [Online]. Available: https://theintercept.com/2016/08/19/the-nsa-was-hacked-snowden-documents-confirm/. [Accessed: 19-Aug-2016]

[29]"NSA: Some used spying power to snoop on lovers," *CNN*. 2013 [Online]. Available: http://edition.cnn.com/2013/09/27/politics/nsa-snooping/index.html. [Accessed: 24-Apr-2016]

[30]C. B. Ravn, "Amerikansk politi under anklage for ulovligt at snage i data om partnere, journalister og naboer," *Version 2*. 2016 [Online]. Available: https://www.version2.dk/artikel/amerikansk-politi-anklaget-at-misbruge-adgang-offentlige-databaser-968179. [Accessed: 30-Sep-2016]

[31]"Cops deploy StingRay anti-terror tech against $50 chicken-wing thief." [Online]. Available: http://www.theregister.co.uk/2016/05/04/cops_deploy_antiterrorist_tech_against_50_chickenwing_thief/. [Accessed: 16-May-2016]

[32]S. Gjerding, "Scharf: Bred elektronisk overvågning giver sjældent resultater," *Information*. 2016 [Online]. Available: https://www.information.dk/indland/2016/10/scharf-bred-elektronisk-overvaagning-giver-sjaeldent-resultater?utm_source=Dagens+vigtigste+nyheder&utm_campaign=877415e512-Stop_med_at_tale_politikerleden_op_og_ve8_11_2016&utm_medium=email&utm_term=0_2ccbb7fa83. [Accessed: 12-Oct-2016]

[33]"No Place To Hide Documents." [Online]. Available: http://us.macmillan.com/static/holt/greenwald/NoPlaceToHide-Documents-Uncompressed.pdf. [Accessed: 01-Jun-2016]

[34]G. Greenwald, "No Place To Hide." [Online]. Available: http://glenngreenwald.net. [Accessed: 30-Sep-2016]

[35]"The NSA Files," *The guardian*. [Online]. Available: https://www.theguardian.com/us-news/the-nsa-files. [Accessed: 01-Jun-2016]

[36]J. Kloc, "The definitive guide to NSA spy programs," *The daily dot*. [Online]. Available: http://www.dailydot.com/layer8/nsa-spy-prgrams-prism-fairview-blarney/. [Accessed: 01-Jun-2016]

[37]L. Jensen, "Zuckerberg er ikke til bongopatter," *Information*. 2016 [Online]. Available: https://www.information.dk/debat/2016/09/zuckerberg-bongopatter?utm_source=Dagens+vigtigste+nyheder&utm_campaign=20307fd6f3-Stop_

med_at_tale_politikerleden_op_og_ve8_11_2016&utm_medium=email&utm_term=0_2ccbb7fa83-20307fd6f3-409001805. [Accessed: 14-Oct-2016]

[38]A. Ohlheiser, "Just how offensive did Milo Yiannopoulos have to be to get banned from Twitter?" *The washington post*. 2016 [Online]. Available: https://www.washingtonpost.com/news/the-intersect/wp/2016/07/21/what-it-takes-to-get-banned-from-twitter/. [Accessed: 03-Sep-2016]

[39]"Social Autopsy." [Online]. Available: http://socialautopsy.com. [Accessed: 01-Aug-2016]

[40]"Wave Goodbye to Cyberbullies and Trolls: SocialAutopsy.com," *kickstarter.com*. 2016 [Online]. Available: https://www.kickstarter.com/projects/1968200734/wave-goodbye-to-cyberbullies-and-trolls-socialauto. [Accessed: 04-Sep-2016]

[41]"Social Autopsy Introduction," *YouTube*. [Online]. Available: https://www.youtube.com/watch?v=5hbRFj93IC0. [Accessed: 01-Aug-2016]

[42]"The Mother of All Demos, presented by Douglas Engelbart (1968)." 1968 [Online]. Available: https://www.youtube.com/watch?v=yJDv-zdhzMY. [Accessed: 01-Oct-2016]

[43]J. Postel, "SIMPLE MAIL TRANSFER PROTOCOL," *The internet engineering task force*. 1981 [Online]. Available: https://tools.ietf.org/html/rfc788. [Accessed: 01-Oct-2016]

[44]J. Myers, "SMTP Service Extension for Authentication," *Internet engineering task force*. 1999 [Online]. Available: https://tools.ietf.org/html/rfc2554. [Accessed: 01-Oct-2016]

[45]P. Hoffmann, "SMTP Service Extension for Secure SMTP over TLS," *Internet engineering task force*. 1999 [Online]. Available: https://tools.ietf.org/html/rfc2487. [Accessed: 12-Oct-2016]

[46]M. Rose, "Post Office Protocol - Version 3," *Internet engineering task force*. 1988 [Online]. Available: https://tools.ietf.org/html/rfc1081. [Accessed: 10-Oct-2016]

[47]M. Crispin, "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4," *Internet engineering task force*. 1994 [Online]. Available: https://tools.ietf.org/html/rfc1730. [Accessed: 10-Oct-2016]

[48]D. Crocker, "STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES," *Internet engineering task force*. 1982 [Online]. Available: https://tools.ietf.org/html/rfc822

[49]N. Freed, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies." [Online]. Available: https://tools.ietf.org/html/rfc2045.

[Accessed: 20-Jun-2016]

[50]N. Freed, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types," *Internet engineering task force*. 1996 [Online]. Available: https://tools.ietf.org/html/rfc2046

[51]K. Moore, "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text," *Internet engineering task force*. 1996 [Online]. Available: https://tools.ietf.org/html/rfc2047

[52]S. Harris, *CISSP Certification Exam Guide, 2nd edition*. Mc Graw Hill, 2003, p. 928.

[53]"TALK(1)," *FreeBSD man pages*. 2010 [Online]. Available: https://www.freebsd.org/cgi/man.cgi?query=talk&sektion=1&manpath=FreeBSD+10.3-RELEASE+and+Ports. [Accessed: 12-Oct-2016]

[54]D. Spinellis, "A repository of Unix history and evolution," *Empirical Software Engineering*, pp. 1–33, 2016.

[55]C. Ylonen, "The Secure Shell (SSH) Transport Layer Protocol," *Internet engineering task force*. 2006 [Online]. Available: https://tools.ietf.org/html/rfc4253. [Accessed: 13-Oct-2016]

[56]N. Borisov, U. C. Berkeley, I. Goldberg, and E. Brewer, "Off-the-Record Communication, or , Why Not To Use PGP," *Wpes*, pp. 77–84, 2004.

[57]"About Tor," *The tor project*. [Online]. Available: https://www.torproject.org/about/overview.html.en. [Accessed: 20-Oct-2016]

[58]A. Sanatinia and G. Noubir, "Honions: Towards detection and identification of misbehaving tor hsdirs," in *Workshop on hot topics in privacy enhancing technologies (hotPETs)*, 2016.

[59]C. P. Pfleeger and S. L. Pfleeger, *Security in computing*. Prentice Hall Professional Technical Reference, 2002.

[60]D. Wilson and G. Ateniese, "From pretty good to great: Enhancing PGP using bitcoin and the blockchain," in *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, 2015, vol. 9408, pp. 368–375.

[61]"BitTorrent." [Online]. Available: http://www.bittorrent.com. [Accessed: 20-Oct-2016]

[62]A. Hagiu, "Multi-sided platforms: From microfoundations to design and expansion strategies," *Harvard Business School Strategy Unit Working Paper*, no. 09-115, 2009.

[63]"Patreon." [Online]. Available: https://www.patreon.com

[64]"Henrik Udsen - CV - Ansatte tilknyttet CIIR - Center for informations- og innovationsret," *Copenhagen university*. [Online]. Available: http://jura.ku.dk/ciir/ansatte/?pure=da%2Fpersons%2Fhenrik-udsen(554cb042-e883-476b-998c-23bd8d153fa4)%2Fcv.html. [Accessed: 30-Jun-2016]