Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Proceedings of the Seminars Future Internet (FI) and Innovative Internet Technologies and Mobile Communication (IITM)

Summer Semester 2018

Munich, Germany

Editors

Georg Carle, Daniel Raumer, Stephan Günther, Benedikt Jaeger

Publisher

Chair of Network Architectures and Services

Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München

Proceedings zu den Seminaren Future Internet (FI) und Innovative Internet-Technologien und Mobilkommunikation (IITM)

Sommersemester 2018

München, 26. 2. 2017 – 19. 8. 2018

Editoren: Georg Carle, Daniel Raumer, Stephan Günther, Benedikt Jaeger

Network Architectures
and Services
NET 2018-11-1

Proceedings of the Seminars
Future Internet (FI) and Innovative Internet Technologies and Mobile Communication (IITM)
Summer Semester 2018

Editors:

Georg Carle
Lehrstuhl für Netzarchitekturen und Netzdienste (I8)
Technische Universität München
85748 Garching b. München, Germany
E-mail: carle@net.in.tum.de
Internet: `https://net.in.tum.de/~carle/`

Daniel Raumer
Lehrstuhl für Netzarchitekturen und Netzdienste (I8)
E-mail: raumer@net.in.tum.de
Internet: `https://net.in.tum.de/~raumer/`

Stephan Günther
Lehrstuhl für Netzarchitekturen und Netzdienste (I8)
E-mail: guenther@net.in.tum.de
Internet: `https://net.in.tum.de/~guenther/`

Benedikt Jaeger
Lehrstuhl für Netzarchitekturen und Netzdienste (I8)
E-mail: jaeger@net.in.tum.de
Internet: `https://net.in.tum.de/~jaeger/`

Cataloging-in-Publication Data

Seminars FI & IITM SS 18
Proceedings zu den Seminaren „Future Internet“ (FI) und „Innovative Internet-Technologien und Mobilkommunikation“ (IITM)
München, Germany, 26. 2. 2017 – 19. 8. 2018
ISBN: 978-3-937201-63-4

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2018-11-1
Lehrstuhl für Netzarchitekturen und Netzdienste (I8) NET 2018-11-1
Series Editor: Georg Carle, Technische Universität München, Germany

Vorwort

Vor Ihnen liegen die Proceedings der beiden Seminare „Future Internet“ (FI) und „Innovative Internet-Technologien und Mobilkommunikation“ (IITM), welche die finalen Ausarbeitungen unserer Studierenden enthalten. Die beiden Seminare wurden am Lehrstuhl für Netzarchitekturen und Netzdienste an der Fakultät für Informatik der Technischen Universität München im Sommersemester 2018 durchgeführt. Allen Teilnehmerinnen und Teilnehmern stand es wie in der Vergangenheit frei, die Ausarbeitung und den Vortrag in englischer oder in deutscher Sprache zu verfassen. Dementsprechend finden sich sowohl englische als auch deutsche Beiträge in diesen Proceedings.

Einige der Vorträge wurden aufgezeichnet und sind auf unserem Medienportal unter `https://media.net.in.tum.de` abrufbar.

Wir hoffen, dass Sie den Beiträgen dieser Seminare wertvolle Anregungen entnehmen können. Falls Sie weiteres Interesse an unseren Arbeiten haben, so finden Sie weitere Informationen auf unserer Homepage `https://net.in.tum.de`.

München, September 2018

Georg Carle

Daniel Raumer

Stephan Günther

Benedikt Jaeger

Preface

We are pleased to present you the proceedings of our seminars on "Future Internet" (FI) and "Innovative Internet Technologies and Mobile Communication" (IITM) which took place in the Summer Semester 2018. In all seminar courses organized by our research group, the authors were free to write their paper and give their talk in English or German.

Some of the talks were recorded and published on our media portal `https://media.net.in.tum.de`.

We hope that you appreciate the contributions of these seminars. If you are interested in further information about our work, please visit our homepage `https://net.in.tum.de`.

Munich, September 2018

Seminarveranstalter

Lehrstuhlinhaber

Georg Carle, Technische Universität München, Germany

Seminarleitung

Daniel Raumer, Technische Universität München, Germany
Stephan Günther, Technische Universität München, Germany
Benedikt Jaeger, Technische Universität München, Germany

Betreuer

Paul Emmerich (emmericp@net.in.tum.de)
Technische Universität München

Stefan Liebald (liebald@net.in.tum.de)
Technische Universität München

Holger Kinkelin (kinkelin@net.in.tum.de)
Technische Universität München

Marcel von Maltitz (maltitz@net.in.tum.de)
Technische Universität München

Kajó Márton (kajo@in.tum.de)
Technische Universität München

Heiko Niedermayer (niedermayer@net.in.tum.de)
Technische Universität München

Quirin Scheitle (scheitle@net.in.tum.de)
Technische Universität München

Dominik Scholz (scholz@net.in.tum.de)
Technische Universität München

Jan Seeger (seeger@net.in.tum.de)
Technische Universität München

Fabien Geyer (geyer@net.in.tum.de)
Technische Universität München

Simon Bauer (bauersi@net.in.tum.de)
Technische Universität München

Seminarhomepage

`https://net.in.tum.de/teaching/ss18/seminars/`

Inhaltsverzeichnis

Future Internet

Innovative Internet-Technologien und Mobilkommunikation

Key Performance Indicators of TCP Flows

Patryk Brzoza
Advisor: M.Sc. Simon Bauer
Seminar Future Internet SS2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: brzoza@in.tum.de

ABSTRACT

As most of today's Internet traffic is provided by TCP, this also means that the overall performance is heavily dependent on the quality of these TCP flows. To make it possible to evaluate the state of connections or networks for further potential problem assessment, it is necessary to introduce metrics that act as key performance indicators. This paper introduces multiple of these performance metrics and classifies them into latency, packet loss, throughput and other indicators. As a next step, various methods to conduct measurements and data processing approaches to extract valuable information are presented and discussed. Finally, this allows to draw conclusions about a flow's or network's quality and state.

Keywords

TCP Flows, Performance Metrics, KPIs, Network Monitoring, Intrusion Detection

1. INTRODUCTION

Due to its broad range of features and its numerous advantages, such as providing reliable and ordered communication, TCP has become one of the most important backbones of today's Internet. As most commonly used protocols (including HTTP, SMTP or FTP) are relying on it, even up to 95% of the total traffic volume is provided by TCP [1, 2]. Corresponding to this development, this also implies that the performance of TCP flows is playing a crucial role for the overall performance of applications that depend on the Internet. This implies that ISPs and network administrators should be interested in analyzing and optimizing it, which can be achieved through making use of various numerous network monitoring tools. By collecting and processing flow information, it is then possible to extract certain metrics that are essential for evaluation of the overall quality and performance of a network and its flows. This paper's goal is thus to find such suitable metrics to assess the performance of TCP connections and to determine how to obtain them effectively.

After discussing related work in Section 2 and revising the backgrounds of the TCP protocol in Section 3, this paper presents and categorizes multiple of these *key performance indicators* (KPIs) in Section 4. To acquire different KPIs for performance evaluation, it is however necessary to conclude certain measurement techniques first, which is further discussed in Section 5. As a next step, the collected raw data must then be - depending on its size - processed to reduce it to a necessary amount of information and to extract various KPIs from it. A range of different approaches from various tools is evaluated in Section 6. Finally, Section 7 shows which assumptions and conclusions about different network states and events can be made by inspecting the priorly collected information.

2. RELATED WORK

While this paper's focus is set on defining and evaluating KPIs of TCP flows, the performance of TCP has long been subject of previous research. A thorough analysis of the role of TCP/IP in high performance networks has been concluded by Hassan and Jain in [3]. Based on the fundamentals of the protocol, it defines how to measure the efficiency and create simulations and suitable models for wired and wireless networks.

Additionally, the passive extraction of valuable information like the round trip time has been thematized by Shakkottai et al. in [4] and forms a base for parts of this paper.

Finally, there is a broad range of tools that rely on monitoring network performance to conclude events like intrusion attacks. Examples for this research topic are the tool FlowScan by Plonka in [5] and FIRE by Dickerson in [6].

3. PROTOCOL BACKGROUND

As this paper evaluates the performance metrics of TCP flows, it is necessary to take a closer look at this protocol first. In 1974 the *Transport Control Protocol (TCP)* was first introduced by Cerf and Kahn in [7] and since then has been gradually improved. Compared to other packet-oriented protocols like UDP, which may deliver packets out of order or even lose them during the transmission, TCP allows for a *reliable*, *connection-oriented* and *ordered* communication between two nodes. The three terms imply that upon establishing a transport layer connection between two exchange partners, the transmitted data segments are guaranteed to arrive at their right destination in the correct order until the connection is terminated [8]. In [4], Shakkottai et al. describe a *flow* as a "transfer of packets between two ports of a source-destination pair using a particular protocol". For this reason, such a bidirectional TCP connection consists of two *flows* in each direction [4].

Apart from addressing the partners by their respective IP addresses, the TCP header - visible in Figure 1 - also includes a port number for each flow respectively to allow

doi: 10.2313/NET-2018-11-1_01

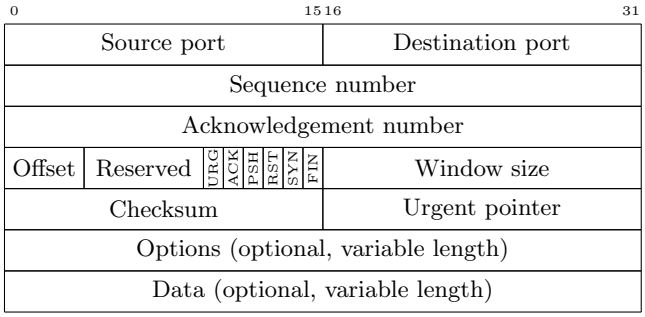


Figure 1: TCP Header [9]

inter-process communication. Additionally there are several other important fields, for instance sequence and acknowledgement numbers to keep the segments in order and retransmit lost packets, or flag bits to exchange control information that is necessary to conclude a three way handshake. This process is necessary to establish a new TCP connection between nodes and will be presented in Section 4. Finally, the congestion window defines how much data a node may send out to its partner before waiting for an acknowledgement. Sophisticated flow and congestion control algorithms are used to avoid congesting the network or the recipient, which may be caused because of sending out too many segments at the same time [9, 8].

4. KPI OVERVIEW

By further inspecting the features which were presented in the last section, it can be noted that many of these properties may constrain the performance of TCP flows. For instance, while retransmissions allow for a reliable communication, they also generate an additional overhead which impacts the ongoing communication by causing delays. This knowledge allows to derive certain metrics which act as key performance indicators and can be subdivided among *latency*, *packet loss*, *throughput* or *other indicators*, which is done in this section [10].

4.1 Latency Indicators

A major subclass of KPIs used to evaluate the connection quality and responsiveness of a TCP flow consists of temporal metrics measuring its latency, as high values can indicate severe delays and thus heavily affect the connection's performance.

Round Trip Time

TCP flows can be modeled in terms of *rounds*, which start with the sender beginning to transmit a window of segments and eventually finish upon receiving back acknowledgements for one or more of these segments from the recipient [11]. Using this assumption, the arguably most common metric in use is the *Round Trip Time* (RTT), which is defined as the time interval between a sender transmitting a segment and the reception of its corresponding acknowledgement segment. This interval may be influenced by several factors including propagation times, the processing and queuing times, which are dependent on the TCP stack implementations, or routing impacts [4]. Obviously it is necessary to keep this value low, as higher latencies would impact the flow performance significantly. An example illustration can be seen in Figure 2, further information about how to calculate the RTT from the monitor node and other measurement techniques are discussed in Section 6.

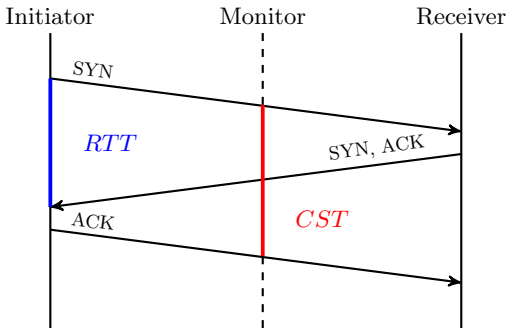


Figure 2: Initiator RTT and CST during a Three Way Handshake [10]

Connection Setup Time

Apart from the RTT, another metric to measure latency in networks is the *Connection Setup Time (CST)*, which describes the time interval that is necessary to establish a new TCP connection by performing a three way handshake. To gain the CST of a successful connection, it is necessary to inspect the time interval between the segments that are involved in the handshake process, i.e. the first SYN and the last ACK segment sent out by the initiator. The process of a full three way handshake can be seen in Figure 2, together with an illustration of the CST. Obviously this additionally implies, that this metric does include the RTT and can be influenced by different factors, for example due to retransmissions that were caused by lost acknowledgements [11]. A difference to the RTT metric measured in ongoing flows is that some implementations may prioritize handshake segments [10].

4.2 Packet Loss

As mentioned in the protocol background section, an important trait of TCP is its reliability, which is provided by detecting and retransmitting unacknowledged segments that may have been lost during their transmission. A common reason for this may be a congestion in the network. For this reason, packet loss is another key performance indicator, as every retransmission causes an additional delay and thus an increase in latency. [12]

A flow's packet loss rate can be visualized by comparing the amount of retransmitted segments r to the overall number of transmitted segments t during the inspected connection interval. Therefore, the *retransmission rate* (RR) can be calculated with the following formula:

$$RR = \frac{r}{t}$$

While retransmissions are a natural behavior in TCP, this value should nevertheless be kept low and sudden increases further investigated. [10, 13]

4.3 Throughput

A third category for evaluating flow performance is the analysis of its data throughput, which influences the responsive-

doi: 10.2313/NET-2018-11-1_01

ness of a connection. This can once again be achieved by introducing analytical throughput metrics.

The *throughput rate* describes the current amount of data that is delivered between the connection nodes and may be dependent on several factors. For instance, TCP's flow control algorithms may throttle the throughput of a flow to avoid congesting the partner node [8]. As this rate is predefined by the congestion window, which additionally is inversely proportional to the RTT [4], this value is correlated to other performance metrics. Thus, given the current window size $cnwd$ and RTT, the current throughput rate T can be estimated using this formula [9]:

$$T = \frac{cnwd}{RTT}$$

Provided that no new congestions are created, a higher throughput rate allows for better performance, while suspicious spikes should be examined in time.

Alternatively, another metric suitable for measuring throughput based on temporal units is the *data transfer time*, which is defined as the time interval beginning with the first and ending with the last data segment arriving at the client, thus describing the time to answer a request [11, 10].

4.4 Other Indicators

Apart from the aforementioned metrics, there are also several KPIs that do not fit into those categories, but may still impact a flow heavily and for this reason are presented in this subsection.

Response Time

For instance, the response time is an factor that is given by higher-layer server applications which are consuming services from the transport layer. The term describes the processing time between the last packet of a request and the first packet of a response (excluding acknowledgement segments) and gives information about the health of an application [10, 13].

Reset Rate

Finally, a way to measure a server's health is to keep track of the amount of sent RST segments over a long term. While TCP connections are torn down with FIN segments usually, RST packets can be used to immediately abort a connection instead and may indicate an error [8, 13].

Apart from the KPIs that were mentioned in this overview, a list including further metrics can be found in [10] and [13].

5. MEASUREMENT METHODS

To acquire the presented metrics for performance evaluation or problem assessment in networks, it is necessary to conduct measuring methods first. This can be achieved by applying two different approaches: *active* and *passive monitoring*. The idea of the active approach is to generate special probe segments that are then evaluated, while in passive measurement dedicated measurement devices are attached to network links in order to capture and process flow data from them passively. The concept of passive flow monitoring was also already introduced briefly in Section 4 for calculating the CST value.

While conducting active measurements gives us full control on the specific data we are currently interested in, it unfortunately also generates interferences to the original flows, as they are loaded with additional measurement data. For this reason, we choose the passive approach to be the more appropriate method to be applied on ongoing TCP flows, as we can apply it without constraining the performance unnecessarily. Apart from this, it first has to be decided, at which place in the network the monitoring points should be placed. An important task is then to adjust the locations and the amount of those measuring devices, so that the percentage of the monitored traffic is maximized, as unfortunate positioning of monitors could deliver poor results. A description of such optimization algorithms has been presented by Chaudet et al. in [14]. In [15] an environment for conducting passive measurements anywhere between the sending and receiving node is described, which is also similar to the description defined in [4]. Here, two unidirectional optical fibers - marked with 0 and 1 in each direction - connect the networks X and Y. These two networks contain nodes that communicate with each other. We tap both fibers - one fiber for each flow in the bidirectional connection - to siphon a part of the signal to the monitoring device, allowing it to collect segments from the connection that may be processed later on. This setup can is illustrated in Figure 3.

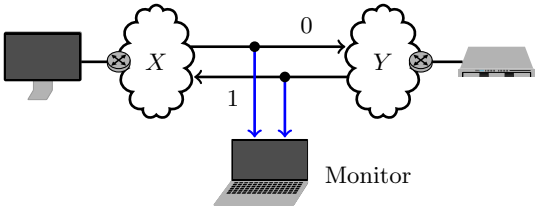


Figure 3: Example of a passive measurement setup used in [4]

Depending on the applicability, it makes sense to choose between acquiring data from one or both flows. Additionally, it is also necessary to detect which connections between which hosts and ports have been established to further concentrate on a concrete flow. Both of these measuring approaches together with possible solutions for the connectivity problem will be discussed in the following two subsections.

5.1 Bidirectional Approach

The *bidirectional approach* assumes, that the monitor has access to segments traversing both direction 0 and 1, meaning that it may access segments going from network X to Y and vice versa. This means that first of all, it has to be determined which hosts are communicating with each other on which ports to focus on a particular connection. One example on how to achieve this is to introduce and make use of a specialized metric that makes it possible to detect connectivity between two hosts [16].

A implementation of such a metric has been presented by Mahdavi and Paxson in [16] and is based on the Framework for IP Performance Metrics (IPPM) specified in [17]: we differentiate between a *instantaneous* connection, which took

doi: 10.2313/NET-2018-11-1_01

place at one certain moment in time, and between a *temporal* connectivity, which notes that a connection took place over a specified time interval. As a next step we can then define our metrics, i.e. either an instantaneous or temporal two-way connectivity consisting of two separate one-way connectivities in each flow direction. When we detect an SYN/ACK segment that acknowledges a handshake process in a flow, we may immediately return `true`. Another indicator for a temporal connectivity may be a RST segment with correct port numbers.

The KPIs presented in the overview can then be calculated easily by correlating relevant segments from a connection's flows with each other. For instance, the general approach for estimating the round trip time RTT, which is an essential performance metric, is capturing segments from one node to another together with its corresponding acknowledgement at the measurement point (Figure 2). By subtracting the timestamp values of the two segments, one can calculate an estimation of this value [15]. This value however depends on the location of the monitor and can easily become underestimated, which is why an alternative method is presented in Section 6 about data processing.

5.2 Unidirectional Approach

Unfortunately, it is not always possible to conduct bidirectional measurements. There are many reasons for this, for example the route between the two connection nodes could have different paths due to being asymmetric or we could not have access to such a measuring point, meaning that we can only capture a flow in one direction [18, 4]. For this reason, *unidirectional approaches* have become a new trend in monitoring research.

The first step is to decide, which flow we have access to and to identify which role the transmitting nodes are playing. By looking at the segments that are transmitted, it is possible to subdivide TCP flows into three categories, that are visualized in Figure 4 [4]:

- **Download Flows:** These flows carry large amounts of data. Capturing a SYN/ACK segment at the fiber going in direction 0, which is followed by data packets, may indicate that the receiver node is located in network X.
- **Feedback Flows:** Observing an initiating SYN segment that is followed by an ACK segment as an answer to the receiver may indicate that the initiator node is located in network X and the receiver node in network Y.
- **Unknown Flows:** Flows which have begun outside of the interval cannot be classified as their initiating segments were never captured.

To detect connectivity in unidirectional flows, we can then once again apply the metrics presented in the subsection before [16].

6. DATA PROCESSING

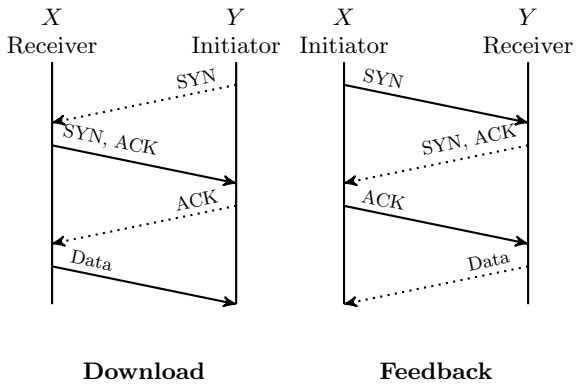


Figure 4: Examples of two possible unidirectional flow types [4]

After having collected enough measurement data, the next step usually is to process it in such a way, so that valuable information can be extracted efficiently and accurately. To achieve this, this section introduces various data processing approaches and evaluate their applicability.

6.1 Data Reduction

Unfortunately, monitoring and capturing packets from heavily loaded flows can easily generate vast amounts of raw data that are hard to process and manage. On the other side, due to collecting too little data one may lose valuable information, which is why it is important to find a good compromise between those two extremes [5]. For this reason, it makes sense to reduce the quantity of input to such a level, so that only necessary information is available. Such reduction techniques can be classified in three different categories [14]:

- **Filtering:** Only packets, that match certain networking criteria (e.g. port numbers or control flags) are captured, otherwise they are excluded.
- **Classification:** Instead of evaluating the whole data set, it is possible to subdivide it into multiple classes and then apply our processing techniques only on necessary classes.
- **Sampling:** Only capture packets randomly. This can be done by either capturing a packet every predefined time interval, every n packets, with a probability of $1/n$ or by combining the aforementioned criteria.

Which technique to use depends on the output we are interested in. For instance, sampling only requires a small amount of calculation and can be set up easily, but does not deliver as specific results as the other two techniques do [14].

Recently newer *data mining* approaches, which are defined as "the process of looking for features and relationships in data" [6], have gained in popularity. One example of such an approach is the implementation of the intrusion detection system FIRE presented in [6], which first extracts multiple fields from the TCP header and then creates an aggregate key defined as *sdp* consisting of the packet's source IP s,

doi: 10.2313/NET-2018-11-1_01

destination IP d and destination port number p. Using this key represents a flow's existence and is necessary for the data mining phase: a network data processor (NDP) maintains a table of selected *sdp*'s and can be used to prepare statistics for them.

6.2 KPI Extraction

Eventually, after having processed the measurement data in an appropriate way, it is now possible to extract valuable KPIs from it. As a main example we have chosen to show how to derive and accurately estimate a flow's round trip time RTT, as it is probably one of the most important KPIs listed above, nevertheless other KPIs can be extracted in a similar way. Which approach to choose heavily depends on the measurement method we chose before.

For example, for bidirectional measurement data, this estimation depends on the location of the monitoring node and can so easily become underestimated. This can be seen in Figure 5, where the original RTT estimation $d1$ captured at the monitor turns out to be smaller than the expected value. One example on how to prevent this behavior is introduced with the RTT estimation method presented in [15], where the inferred estimation was split up into two parts, $d1$, which is the initiator RTT value we calculated before, and $d2$, where we calculate another receiver RTT value based on the previous acknowledgement segment coming from the receiver and a data segment coming from the initiator that was triggered by this acknowledgement. This process can also be seen in Figure 5. By inspecting the value $d = d1 + d2$. we see that we now finally derived a value that matches the real RTT as observed from the initiator more accurately.

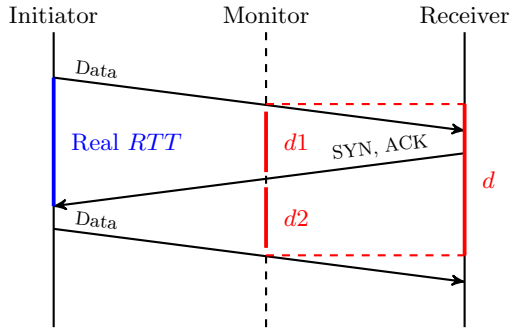


Figure 5: Passive RTT estimation method presented in [15]

This procedure is similar for the other KPIs, for which either the time intervals between segment samples can be calculated using the descriptions from the KPI overview in Section 4 or counted using a counter which increments itself upon e.g. the detection of RST segments to calculate the reset rate or a retransmission to calculate the retransmission rate. A more detailed graphic for this process that includes multiple TCP connection types has been described by Rogier in [10].

On the other side, using unidirectional measurement data for performance metric extraction turns out to be more complicated than the method described before, as we now have to take different flow types into consideration from which information may not instantly be obvious. In [4], three methods that can be used to calculate RTT are introduced:

- **SYN-based Method:** Both download and feedback flows are detected in direction XY. For download flows, this means that we are interested in the RTT for the path XYX, which can be done by calculating the time interval between the SYN/ACK and the data segment. Vice versa, for a feedback flow the path of interest is YXY, meaning that we are interested in the time interval between the initiating SYN and the following ACK segment. Both calculations may differ due to serialization delays.

- **Flight Method:** Due to TCP's congestion control, flows have a specific structure defined as *flight behavior*, whereas flights are defined as "consecutive packets with nearly identical inter-arrival times" [4]. This behavior implies that a flow may consist of groups of flights that are separated by gaps. It is then possible to derive an estimation of RTT by calculating and averaging the intervals between the first packet of each of those flight groups.

- **Rate Change Method:** For this method it is necessary to look back at the flow and congestion control algorithms implemented in TCP and described in the protocol background section. This leads to TCP having two different operation modes: the slow start mode, in which the congestion window increases by one maximum segment size (MSS) for every acknowledgement received, and the congestion avoidance phase, in which the MSS is incremented every RTT, while the congestion window size is halved for every segment that is lost. This special property can be exploited to determine an estimation for the RTT: assuming that x is the number of bits transferred in the time interval $[t_0, t]$, the instantaneous rate is defined as $\frac{dx}{dt}$ and that the interval $[t_0, t]$ is placed in the congestion avoidance phase, does not include packet drops and has a persistent RTT, then the overall RTT can be then calculated using this generic formula:

$$RTT = \sqrt{\frac{MTU}{\frac{d^2x}{dt^2}}}$$

  A detailed proof for this formula with more background information can be found in [4].

Out of the three methods, the SYN-based approach seems to be the most useful one as it delivers estimates for more sets of data than the other two methods do [4]. Its complexity is also significantly lower than for the other methods as it does not require as many calculations. Additionally, this approach can also be readapted to extract other KPIs in a similar way as before.

7. NETWORK EVENT DERIVATION

Now that we have successfully derived several key performance metrics from our monitored TCP flows, we may now use them to gain further information about the state of connections or networks. To keep track of sudden changes or

doi: 10.2313/NET-2018-11-1_01

suspicious anomalies in those extracted KPIs, various network visualization tools that create graphical statistics can be used. One example for such a tool is FlowScan, which was specified by Plonka in [5] and provides a powerful option to generate such visualizations even for longer measurement intervals. The following subsections will thematize KPI applications in two different fields, from which assumptions by inspecting certain metrics can be made.

7.1 Intrusion Detection

Due to the persistent danger of becoming a victim of a cyber attack, one of the most important tasks of a network administrator is to keep the network safe from possible attacks from the outside and to detect them in time before it becomes compromised. The process of identifying this kind of activity in a network is defined as *network intrusion detection* [6]. For example, a first step of conducting an attack that exploits certain TCP functionalities may be a port scan, that iteratively sends SYN segments from the attacker node to a list of ports on the target to detect whether they are open. If that is not the case, a closed port would normally send back a RST segment to abort the connection process. Thus, a significantly higher reset rate may already be an indicator of such an attack type. Another type of network abuse attacks that heavily affects the performance of networks are denial-of-service (DoS) floods, which aim to make a target unavailable. A common method to accomplish this is to flood the victim with a mass of ACK segments, which have to be processed and may receive RST segments as a response. As before, this would bring a sudden increase of the reset rate, but other KPIs including the current throughput, latency or response time may also be affected by the mass of requests the server has to process [5]. An example visualization of a DoS flood captured in the tool FlowScan can be seen in Figure 6.

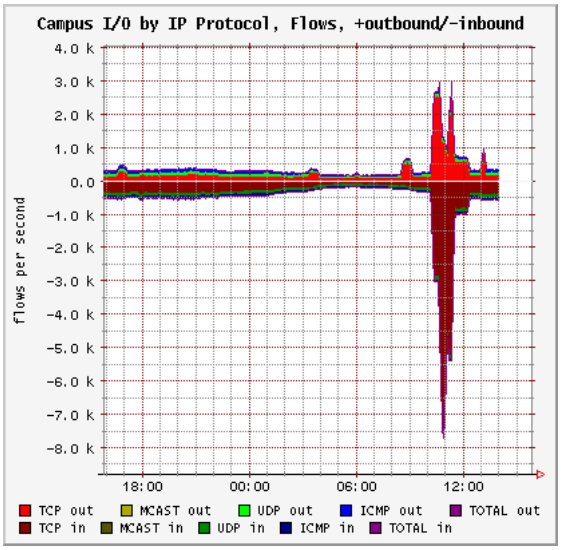


Figure 6: Example of a DoS flood attack visualized using FlowScan [5]

7.2 Network Health

Finally, a significant part of the overall performance is influenced by the network itself. A survey on speed limiting factors of TCP flows was conducted by Timmer et al. in [19] revealed and that approximately a quarter of the flow performance depends on the network itself. This can mean, that e.g. a persistently low throughput can be a sign of a bottleneck located in the network and should be investigated. An additional impact are the applications in use: a slow sender, that is using multiple applications that rely on TCP at once or that is running out of resources, can for instance have a high response time.

8. CONCLUSION

In this paper, we gave a brief summary of the TCP protocol together with its numerous features and the definition of the flow term. Due to the implementation of these features, the performance of TCP flows is constrained by multiple factors. We introduce metrics acting as key performance indicators for connections between end points. These KPIs were subdivided into four categories: latency, packet loss, throughput and other indicators. To be able to determine these values, it is necessary to conduct means of measurement first. We have differentiated between active and passive measurement methods and chose the latter option to be the more appropriate approach for ongoing flows. Depending on the accessibility and other factors, it makes sense to either monitor data from only one or both flows of a connection. Bidirectional approaches can easily detect ongoing connections using specialized metrics, while unidirectional approaches have to determine the flow type before proceeding. To extract specific information from these measurements, it is often necessary to reduce the amount of collected data first. We presented three data reduction methods together with newer data mining approaches to accomplish this task. We evaluated different KPI extraction methods by using round trip time estimation as an example: as for bidirectional data, it can be derived fairly easy by correlating associated segments with each other, while for unidirectional data the SYN-based method delivered the most reliable outputs. Finally, we showed the applicability of KPIs by applying it on intrusion detection and network error assessment.

9. REFERENCES

[1] M. Mellia and H. Zhang. Tcp model for short lived flows. *IEEE Communications Letters*, 6(2):85–87, Feb 2002.

[2] Anja Feldmann, Jennifer Rexford, and Ramón Cáceres. Efficient policies for carrying web traffic over flow-switched networks. *IEEE/ACM transactions on Networking*, 6(6):673–685, 1998.

[3] Mahbub Hassan and Raj Jain. *High Performance TCP/IP Networking*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

[4] S. Shakkottai, N. Brownlee, A. Broido, and k. claffy. The RTT distribution of TCP flows on the Internet and its impact on TCP based flow control. Technical report, Cooperative Association for Internet Data Analysis (CAIDA), Mar 2004.

[5] Dave Plonka. Flowscan: A network traffic flow reporting and visualization tool. In *Proceedings of the 14th USENIX Conference on System Administration*, LISA '00, pages 305–318, Berkeley, CA, USA, 2000. USENIX Association.

doi: 10.2313/NET-2018-11-1_01

[6] J. E. Dickerson and J. A. Dickerson. Fuzzy network profiling for intrusion detection. In *PeachFuzz 2000. 19th International Conference of the North American Fuzzy Information Processing Society - NAFIPS (Cat. No.00TH8500)*, pages 301–306, 2000.

[7] V. Cerf and R. Kahn. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, 22(5):637–648, May 1974.

[8] W. Richard Stevens. *TCP/IP Illustrated (Vol. 1): The Protocols*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.

[9] *Transmission Control Protocol Specification*. RFC 793, September 1981.

[10] B. Rogier. How to measure network performance through passive traffic analysis. `http://blog.performancevision.com/eng/earl/how-to-measure-network-performance-through-passive-traffic-analysis`. Last accessed on 2018/03/23.

[11] N. Cardwell, S. Savage, and T. Anderson. Modeling tcp latency. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, volume 3, pages 1742–1751 vol.3, Mar 2000.

[12] P. Benko and A. Veres. A passive method for estimating end-to-end tcp packet loss. In *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, volume 3, pages 2609–2613 vol.3, Nov 2002.

[13] D. Shanahan. 5 Key TCP Metrics for Performance Monitoring. `https://www.linkedin.com/pulse/5-key-tcp-metrics-performance-monitoring-daniel-shanahan`. Last accessed on 2018/03/23.

[14] Claude Chaudet, Eric Fleury, Isabelle Guérin Lassous, Hervé Rivano, and Marie-Emilie Voge. Optimal positioning of active and passive monitoring devices. In *Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology*, CoNEXT '05, pages 71–82, New York, NY, USA, 2005. ACM.

[15] Sharad Jaiswal, G Iannaccone, C Diot, J Kurose, and D Towsley. Inferring tcp connection characteristics through passive measurements, 04 2004.

[16] *IPPM Metrics for Measuring Connectivity*. RFC 2498, January 1999.

[17] *Framework for IP Performance Metrics*. RFC 2330, May 1998.

[18] Hao Jiang and Constantinos Dovrolis. Passive estimation of tcp round-trip times. *SIGCOMM Comput. Commun. Rev.*, 32(3):75–88, July 2002.

[19] M. Timmer, P. T. de Boer, and A. Pras. How to identify the speed limiting factor of a tcp flow. In *2006 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, pages 17–24, April 2006.

doi: 10.2313/NET-2018-11-1_01

Use case study on machine learning for network anomaly detection

Edin Citaku
Advisor: Simon Bauer
Seminar Future Internet SS2018
Seminar Innovative Internet Technologies and Mobile Communications
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: citaku@in.tum.de

ABSTRACT

Technological advancement has reached the point where the vast majority of businesses rely on computer networks. Thus businesses are exposed to a manifold of network security threats such as network intrusions which can pose a serious threat to them. Network intrusions can be very hard to detect and stop since attackers are steadily developing new ways to intrude into a system. Anomaly detection is a promising approach in solving this problem. Any deviations from the normal state of the system are interpreted as harmful. Thus even novel forms of attacks that alter the system in an unusual way will be detected. This assumption can also backfire and lead to a high rate of false positives since networks often show a lot of variation in their traffic. We will discuss possible ways to circumvent this downside of our approach. Machine learning has shown to be an useful tool for generating an effective notion of what normal or abnormal behaviour in a network system is. Since using machine learning for network anomaly detecting is a hot topic and new research has been accumulating steadily we have decided to review the most common applications of machine learning used for anomaly detection and explain how they were implemented in recent research. In the end we discuss which challenges still remain and propose possible ways to overcome them.

Keywords

machine learning, anomaly detection, intrusion detection

1. INTRODUCTION

Network Intrusion Detection(NID) is an issue that has huge concern in network security. Victims of such intrusions can range from small businesses to military facilities. These attacks can cause tremendous costs and can even be a danger to national security thus new development in this area is of great interest. Network intrusions are unauthorized activities on a computer network that attempt to compromise the integrity, confidentiality or availability of resources on said network. Network Intrusion Detection Systems(NIDS) are programs which actively try to detect those intrusions. Network Intrusion detection can be divided into two categories *misuse detection* and *anomaly detection*. In the former, abnormal behaviour is defined first and then all other behaviour is defined as normal. This approach works well in recognizing already known attack patterns but it is not suitable to detect novel forms of attack. Since NIDS and attackers are in a constant arms race and new forms of attacks are developed steadily, *anomaly detection* shows to be a promising approach. Here normal behaviour is defined first and behaviour deviating from this norm is interpreted as harmful. The primary objective of this technique is to find a suitable way to define what normal behaviour exactly is.

Machine Learning(ML) has helped to advance many different areas of research in the past decades thus using it for anomaly detection does seem like a suitable approach. Like many other areas of research Network Anomaly Detection comes with its unique properties which make it necessary to tweak the methods of ML in such a way that their use for this particular problem becomes practical. ML is a type of algorithm, where the solution for a problem is not explicitly programmed but the algorithm learns its task with the use of training data and progressively improves its performance. In this paper we will look at the different use cases of ML in anomaly detection. For this we will first explain the different types of ML and then for each method of ML we will show a use case, that was developed in current research and explain their unique properties. Later on we will discuss the current state of ML in anomaly detection and explain the different hurdles that still remain and propose ways to overcome them.

The remainder of the paper will be structured as follows. In Section 1 we will explain different background studies relevant to ML and anomaly detection. Later in Section 2 we will talk about work already published related to this paper. Then in Section 3 we will explain different types of ML-techniques that are relevant to network anomaly detection. In Section 4 we analyse implementations of A-NIDS. Finally in section 5 we will discuss the obstacles that still remain in NID using ML and possible ways to overcome them.

2. RELATED WORK

In the recent past numerous researches have published reviews about the current state of ML in anomaly detection. Chih-Fong Tsai et al.[28] wrote such a review where the focussed on the time span between 2000 and 2007. They compared the different studies by their designs, datastes and other experimental setups. A major focus of their paper was to observe how the focus of the research community has evolved in those years. Animesh Patcha et al.[21] worked on a very thorough review of the uses cases of anomaly detec-

doi: 10.2313/NET-2018-11-1_02

tion research where they also looked at ML approaches. A major difference this paper shows compared to the related work listed is the thorough discussion of the challenges of ML in anomaly detection. Major conclusions from this discussion come from the work of Robin Sommer et al. [26]

3. BACKGROUND STUDIES

3.1 Types of classification

When classifying data traffic as either normal or abnormal there are two types learning we can use. These are namely supervised and unsupervised learning. In the former the training data given to the algorithm is already labelled, while in the unsupervised approach it is not labelled. A label in this context is some kind of additional information that indicates to what class a specific dataset belongs. Unsupervised learning is common in the use case of anomaly detection since actual data of an attack is hard to come by.[19]

3.2 Commonly used Datasets

The most commonly used datasets in network anomaly detection research are the DARPA Intrusion Detection Data Sets [17], that were generated by simulating a Airforce network, and the KDD Cup 1999 Data[5], which is data derived from the DARPA Data sets. These datasets are supposed to measure how well an algorithm performs under real life circumstances. Furthermore, it is possible to compare different algorithms if they have been evaluated on the same datasets.In the recent past a lot of criticism about these datasets has been voiced since they are not gathered from a real networks, but from a simulation and also are already considerably old. Despite this criticism, most of current researchers still uses these datasets to evaluate their approaches.

3.3 Measures of Accuracy

When evaluating an algorithm for anomaly detection, the most important metric is its accuracy. We distinguish between two different types of errors, false positives and false negatives. False negatives occur when an intrusion is taking place but is not detected. A false positive on the other hand means, that an anomaly is detected, but no attack is happening. One could assume, that diminishing false negatives is of primary concern to researchers, since an undetected attack is a serious danger for the integrity of a network. While this is certainly the case, it must be noted, that high rates of false positives are a major problem in the field of anomaly detection and diminishing it, is of high priority for researchers.[7]

4. MACHINE LEARNING TECHNIQUES

There are two different approaches on using ML for network anomaly detection. With single classifiers only one kind of ML is, while for hybrid classifiers multiple tools of ML are used in conjunction.

4.1 Single classifiers

We will take a further look at the following ML approaches for network anomaly detection:

- Decision Tree Learning

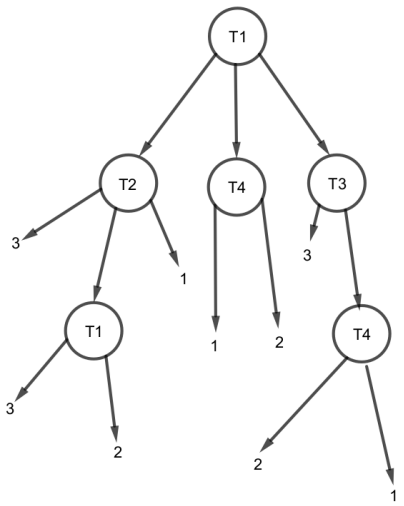


Figure 1: A simple decision tree[13]

- Bayesian Networks
- Genetic Algorithms
- K-nearest neighbor
- Support Vector Machines
- Artifical Neural Networks

4.1.1 Decision Tree Learning

A decision tree is a tool commonly used for classification, where a tree-like graph models different decisions and their consequences. In this graph the attributes of the target are assigned to each node, while the leaf represents a possible classification of the object. In anomaly detection each leaf would be either labelled as ”normal” or as ”abnormal”, although it would be possible to add classes, that specify the type of anomaly. The edges in the graph correspond to different attribute values. To classify an object, you simply traverse the tree, picking the edge, that corresponds to its attribute value. The leaf you reach in the end decides the class your object belongs to. An example for a decision tree can be found in Figure 1. The decision tree is built according to the training data, where different algorithms are possible to use, the so called ID3-algorithm being the most commonly used. Joong-Hee Lee [13] built a model based on decision trees, and showed their performance on the KDD 99 Dataset. The data from the KDD 99 Dataset was first preprocessed, by selecting specific features, that suitably characterize the data. The researchers decided on selecting features, that are already selected in other IDS systems like snort [1], to detect intrusions. To built the tree the researches choose the commonly used ID3 algorithm.

4.1.2 Bayesian Networks

A Bayesian Network[9] is a graphical model representing probabilistic relationships among variables via directed acyclic graphs (DAG). In the DAG the vertices represent events and the edges represent relationships between these events. A numerical component is added with links in the DAG, that represent conditional probabilities of each node in context

of its neighbors. Bayesian networks both have probabilistic as well as causal components making them suitable for classification, even if data is missing.
Figure 2 shows such a bayes network. Rain has an influence on whether the sprinkler is activated. Both the rain and the sprinkler influence whether the grass is wet.
To build a suitable Bayesian network, prior knowledge about the dependencies in the problem are required. It has to be noted, that the prior knowledge the researcher has to bring can show to be a downside since wrong assumptions can decrease the performance and accuracy of the system. A major upside of Bayesian Networks is that they show significantly less computational time in construction and classification compared to other methods of ML. Nahla Ben Amor et al.[3] compared Naive Bayesian Networks to decision trees. They showed, that both had comparable accuracy, while the Bayesian network was computationally much cheaper than the decision tree. Valdes et al. proposed[2] an NADS using naive Bayesian networks. Their model, that is implemented in EMERALD[22] has the capability to detect distributed attacks, where individual attacks are not suspicious enough to generate an alert. One down side of Bayesian networks is that they still show similiar accuracy to simple threshold based systems while still being computationally much more demanding as pointed out by Kruegel et al. [12]. Another problem Kruegel et. al [12] identified is the high percentage of false positives which make the use of traditional Bayesian networks in real life situations impractical. They propose a solution to this problem where they aggregate the outputs of different IDS sensors to produce one single alarm. They assume that one singular sensor is not enough to effectively detect intrusions.

4.1.3 Genetic Algorithms

Genetic algorithms(GA) [30]are a heuristic inspired by the process of natural selection. They are commonly used for optimisation and search problems. GA use techniques inspired by biology, such as mutations, crossover and natural selection. Thus with GA it is possible to derive classification rules or select appropriate features or optimal parameters for the detection process. In recent research Wei Li[15] showed advancements in using GA for intrusion detection. In his research he introduced an algorithm that constructed rules in the form:
if {*condition*} *then*{*act*}
These rules are represented as genes in a chromosome, where up to fifty-seven genes make up a chromosome. In the first generation of the algorithm multiple individuals with a set number of chromosomes and random genes are generated. By only letting individuals with the best fitting rule set, represented by their genes, move forward to the next generation the accuracy of the detection gets gradually better. After each generation the individuals reproduce generating offspring with new sets of genes where the chromosomes of its parents are mixed together. When mixing chromosome sets so called cross-overs can occur with a certain probability where parts of chromosomes can be exchanged between each other. On top of that there is a chance for random mutations introducing singular changes in genes. With all these mechanics it is ensured that the algorithm gradually finds the optimal rule set for the given problem.

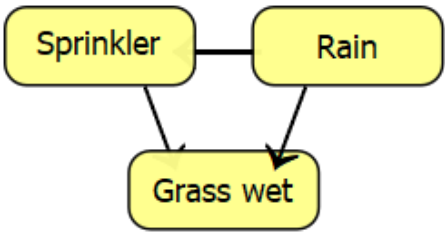


Figure 2: Simple example of a bayesian network[4]

4.1.4 K-nearest neighbor

k-nearest neighbor[16] (k-NN) is an algorithm used for classification and regression. It computes the distance between different points in the input vectors and assigns the point to a cluster of its k-nearest neighbors. k is an important parameter which can have a huge impact on the performance of the algorithm. Typically k is chosen through empirical measures, e.g. different instances of k are tried out, and the instance which yields the best results is chosen. To measure the distance between two data points typically euclidean distance is used, but metrics, such as the Hamming distance can be suitable to. k-NN is called instance based learning or lazy learning which means that new problems are directly compared to the data seen in training without any steps of explicit generalization. Once the algorithm has been trained, the classification of new data is simple: A majority vote is done where the new data point is assigned the class chosen by the majority of its k-nearest neigbors. One use case is the research of Yihua Liao and V. Rao Vemuri [16]. The Algorithm used system calls in the network as input data and generated individual program profiles. The researchers achieved a false positive rate as low as 0.44%, while detecting all intrusions. For training they used the 1998 DARPA BSM audit data. Another interesting approach using k-NN was done by Sutharshan Rajasegarar et al. [23]. They used k-NN to detect anomalies in wireless sensor networks and applied clustering before the actual anomaly detection to minimize computational overhead.

4.1.5 Support Vector Machines

SVM are supervised learning tools used for classification. Objects are represented as hyper dimensional vectors. The SVM maps the training vectors into a hyper dimensional space and then derives an optimal hyper plane dividing the different classes. The decision boundaries are obtained by the support vectors representing the hyper planes rather than the whole training samples. Thus it is robust to outliers. The SVM is mainly designed as a tool for binary classification where training data for both classes is present. Since it is very hard to obtain network data representing an attack different approaches have been made to modify SVMs in such a way that only the data for one class is needed. This type of SVM is called One-class SVM[24]. Such a SVM is able to detect whether an input vector is similar to the dataset it was trained on. Kun-Lun Li et al.[14] developed an approach of ML for anomaly detection where they improved the traditional one-class SVM. In regular one-class SVMs the origin of the hyperplane is defined as the second classification type. The researches modified this method in such a way, that also points ”close enough” to the origin, are detected as anomalies, as seen in Figure 3.

doi: 10.2313/NET-2018-11-1_02

4.1.6 *Artificial Neural Networks*

Artificial Neural networks(ANN) are information processing units inspired by biological neural networks in animal brains. These ANN constitute of connected nodes called artificial neurons each having the ability to process an input signal and then send it to other neurons connected to it. Multilayer Perceptron(MLP) is a architecture of ANN where the neurons are ordered in layers. It consists of input nodes, called sensory nodes more hidden computation nodes and an output layer of computation nodes. Each connection between the nodes is associated to a scalar weight which is initially assigned at random. During the training stage those scalar weights are adjusted usually with the so called back propagation algorithm. An example for an MLP can be seen in Figure 4. Another common architecture of ANN is the so called self-organizing feature map(SOFM)[11]. In this ANN two dimensional discretized representations of the input space of the training data, called *maps*, are generated. Chandrika Palagiri[20] proposed an NIDS using artificial neural networks. They both implemented MLP and SOFM. The results were as expected all anomalies were being detected with the main concern of a high false positive rate. 76% of detected anomalies were false positives. The researcher suggested, that the high rate of false positives stems from the lack of training data for each individual type of attack. The classification times of the program were fast making them suitable for real time classification.

4.2 Hybrid Classifiers

An approach to network anomaly detection gaining rapid popularity is the so called hybrid classification where multible tools of ML are used in different steps of the algorithm. Typically raw data is being preprocessed to some kind of intermediate result. Those results will be taken as input for the second ML tool to produce the final classification. It is possible for example to simply use the step of preprocessing to eliminate unrepresentative training data to then do the actual learning in the second step. You can also use hybrid classification to integrate two different techniques in which the first one aims to optimize the learning performance(e.g. parameter tuning) of the second model of prediction.[29]

4.2.1 *Hybrid classification with enhanced SVM*

One example of such a hybrid classifier is found in the work of Taeshik Shon et al. [25]. The researchers first use a SOFM to create a profile of normal traffic packets for the

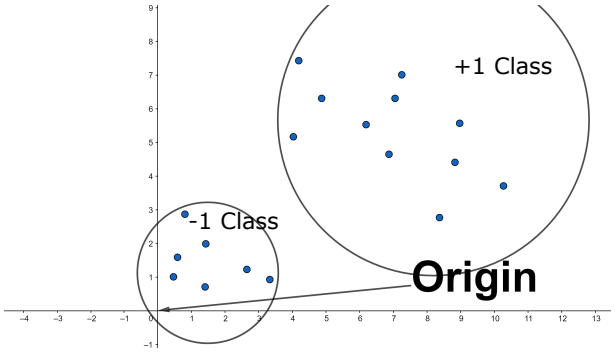


Figure 3: Improved one-class SVM[14]

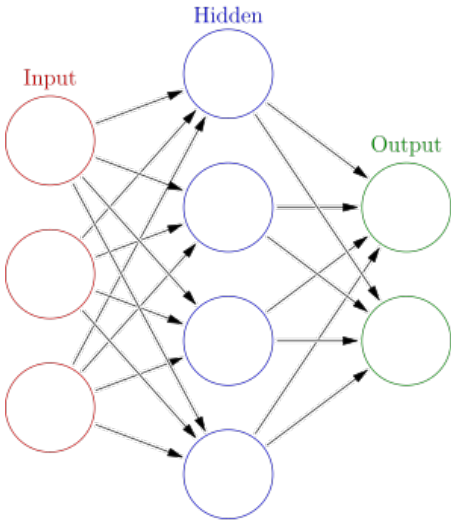


Figure 4: Example of an ANN[8]

SVM. After using a packet filtering scheme to reject incomplete network traffic they apply feature selection using GA on this data to extract optimized data from the raw internet traffic. Once all of this preprocessing is done they use an enhanced SVM to classify the data. The enhanced SVM is a derivative of the traditional SVM which is used for unsupervised learning. The researchers deem this enhancement as necessary because in the traditional SVM approach both normal and abnormal data is needed. Since the goal is to detect *novel* attacks such data is not easily available, and the traditional approach must be modified.

4.2.2 *Decision Tree Classifier with GA-based Feature Selection*

Gary Stein et al.[27] proposed a NIDS based on decision tree classification where the data is preprocessed with GA before the decision tree is generated. The preprocessing step has the purpose of feature selection and is based on the wrapper model.[10] Between each generation of the GA the optimal features, represented by individuals with specific genes are selected for the next generation. This selection is done by constructing a decision tree based on the feature selection each individual represents. These decision trees are then evaluated by their performance. The individuals, whose corresponding decision tree yields the best results then reproduce offspring carrying their genes to the next generation. The feature selection is coded into the genes with binary values where 0 means that a feature is not selected and 1 means that the feature is selected. The researchers showed, that this GA based feature selection improved the accuracy of the final decision tree, compared to a decision tree, where no feature selection was done.

5. CHALLENGES OF MACHINE LEARNING IN NIDS

Despite numerous advancements in the past decades ML for NIDS is still purely a topic of interest for researchers. No NIDS using ML is widely used in industrial settings. This hints that there are still challenges to overcome before

doi: 10.2313/NET-2018-11-1_02

Table 1: Types of ML and their characteristics

Type of ML	Generation	Classification
Decision Tree	Treelike Structure, generated on the basis of training data	The tree is traversed, according to attribute values
Bayesian Network	Directed Acyclic Graph built according to the input data and with additional knowledge by researcher	Decision is done according to the graph
Genetic Algorithms	Features of the classification are first picked at random, then they gradually improve with mechanisms inspired by evolutionary biology, like selection, mutations and crossover.	Rules generated by algorithm are followed.
k-Nearest Neighbor	Input vectors are assigned to a cluster of its k-nearest neighbors	Classification is done, by majority vote. A new object belongs to the same class as the majority of its k-nearest neighbors.
Support Vector Machine	Assigns training vectors a to hyperspace, and computes an hyperplane that divides points into two classes	Classification is simply done by checking what side of the hyper plane the observed vector belongs too.
Artificial Neural Networks	Artificial Neurons, inspired by their biological counter parts, are simulated. These neurons adapt to training data .	Neurons communicate their classification decision with output nodes .

widespread use is viable. In the following we want to discuss these hurdles and what steps have to be taken to overcome them.[26]

5.1 Outlier Detection and Closed World assumption

Machine learning tools are best at finding similarities between data sets. They generally do not exceed at finding outliers in a given dataset. Finding outliers is what network anomaly detection at its core is. We try to find patterns in network flow that do not match the normal state of the network. One could argue that NIDS are doing simple classification because they classify their input data into two categories namely "normal" and "abnormal". This comparison still does not hold, since in traditional ML approaches training data of *all* classes in large quantities are required. This requirement is impossible to fulfil for network anomaly detection because we specifically try to find *novel* attacks. In the approach of anomaly detection we deem all abnormal behaviour as potentially harmful. This assumption does not hold true in real life networks. As quoted by Witten et al.[31]:" *The idea of specifying only positive examples and adopting a standing assumption that the rest are negative is called the closed world assumption. . . . [The assumption] is not of much practical use in real- life problems because they rarely involve "closed" worlds in which you can be certain that all cases are covered.*" Specifically this flawed assumption is the reason for the biggest problem in Network anomaly detection: The high rate of false positives. This make them impossible to use in the real world which leads us to the next problem.

5.2 High cost of errors

Generally an error in intrusion detection is associated with a much higher cost compared to other applications of ML. Thus even if the anomaly detection has the same accuracy, or even better than applications of ML in other domains, it is not necessary that such a method is of any real world value. A false positive, meaning detecting an anomaly that is not there costs extensive time of analysis just to determine it was innocent behaviour after all. Even a small rate of false positives can render a NIDS unusable.[6]. High false positive rates are a major problem of anomaly detection. False negatives on the other hand, have potential of causing serious damage: Even a single compromised system can seriously undermine the integrity of the IT structure. Thus researches have to juggle between false positives and false negatives. To reduce the amount of false positives the algorithm at hand must become *more* sensitive to abnormal behaviour thus raising the rate of false positive at the same time. According to Robin Sommer et al.[26] the number one priority should be to reduce the rate of false positives since they are a big reason why business are not willing to use ML driven anomaly detection. A possible way to reduce the rate of false positives as suggest by Robin Sommer et al.[26] is to reduce the scope of the algorithm so that the algorithm only detects a specific kind of anomaly instead of anomalies in general. This approach also circumvents the problem of juggling between false positives and false negatives. By reducing the scope the relations between the data get much tighter making it easier to differentiate between normal and abnormal behaviour.

5.3 Diversity of Network Traffic

Another problem is that network traffic is more diverse than one would intuitively expect. The networks basic characteristics like bandwidth or duration of connections can vary greatly throughout the day. A solution to this problem is to simply apply aggregation to the input data. This makes it easier for the tool at work to find a notion of normal behaviour and does not waste any valuable information since network data is most of the time noisy anyway. For example in a business where bandwidth usage around the time of lunch is heavily reduced from other times in the day a spike in network flow could be of concern and should be detected as an anomaly. But in a lot of other cases, such anomalies are not a concern of network security. This Network diversity in fact makes it really hard for systems to find a suitable notion of normal.

doi: 10.2313/NET-2018-11-1_02

5.4 High computational demand

A major issue of ML tools is the high computational demand they take while not providing any major performance improvements compared to much simpler methods. In fact in a lot of cases simple threshold based systems provide solutions with comparable detection accuracy compared to their ML counterparts. This can mean higher cost for a business since more computational time is needed to set up the system. Not only is setting up the system by applying the training data computationally demanding also the actual classification can take a considerable amount of time. Time that could be crucial in stopping an attack. On top of that in a threshold based systems detected anomalies are easy to understand and deciding what further actions have to be taken is simple. On the other hand with ML based systems the sheer complexity makes understanding why an anomaly was detected very hard and thus makes it harder to react accordingly. Thus the accuracy of ML tools has to increase in order to justify their high complexity and computational demand.

5.5 Difficulties of Evaluation

One big concern in anomaly detection is that the datasets most commonly used for evaluation are not representative to current real world networks. The two most common used datasets are the DARPA/Lincoln Lab packet traces and the KDD Cup dataset derived from them which are both publicly available. Both datasets are almost two decades old making studies of current forms of attack impossible. Furthermore the DARPA dataset was artificially generated, by simulating an Air Force network. Whether one can derive conclusions for real world networks by researching synthetic datasets is highly doubtful. The KDD Cup datasets which are directly derived from the DARPA datasets show the same problems. On top of that, both datasets show simulation artefacts.[18]. These artefacts can heavily bias the evaluation, since it is possible that the ML tool at hand learns through these artefacts. In that case any deduction to the real world is impossible. A commonly used example that illustrates the issue pretty well is a story where US researchers were trying to detect tanks in images using neural networks.[32] The mistake they made was that all the pictures of tanks were made on a cloudy day, while the pictures of images without tanks were made on a sunny day. What happened was that the neural network was trained on detecting the weather instead of detecting the tanks. In our case the equivalent would be that our algorithm is not trained on detecting anomalies but on detecting artefacts in the dataset. Thus it is necessary to find another source of data to evaluate IDS. Some researchers choose to use data gathered from their own networks. This solution is not satisfactory since big networks, commonly used by businesses, are fundamentally different than smaller scale networks in research facilities. [26] A way to gather suitable data for evaluation would be to directly gather it from businesses. The problem is that most businesses are not willing to share data regarding their networks, since a lot of sensitive information can be deduced from it which in turn can hurt these businesses. Thus the data has to be anonymized, preferably in such a way that no information relevant to anomaly detection is lost.

6. CONCLUSION

Network anomaly detection is of high importance to businesses since a network intrusion can have catastrophic consequences. Machine Learning shows to be a promising approach because novel forms of attack can be effectively detected. Through Machine Learning the system learns what normal and abnormal behaviour in the network means. Since most attacks alter the system in an unusual way they will be detected. This paper briefly describes the foundations of different machine learning approaches, such as support vector machines or artificial neural networks and explains how they can be used to detect network anomalies. To do this we present recent research for each approach and explain the most important details of it. Later we discuss the hurdles that still remain in network anomaly detection and how to overcome them. It is clear that the whole research community will benefit from more reliable datasets. More work has to be done in this area. With new approaches of ML being developed steadily and computers getting faster there is still hope that Machine Learning in anomaly detection will be widely used in the near future.

7. REFERENCES

[1] https://www.snort.org/[Online, accessed 01-April-2018].

[2] V. A. and S. K. Adaptive, model-based monitoring for cyber attack detection. In *Recent Advances in Intrusion Detection. RAID 2000. Lecture Notes in Computer Science*. 2000.

[3] N. B. Amor, S. Benferhat, and Z. Elouedi. Naive bayes vs decision trees in intrusion detection systems. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 420–424. ACM, 2004.

[4] AnAj. Simplebayesnetnodes.svg. https://commons.wikimedia.org/wiki/File:SimpleBayesNetNodes.svg, 2013. [Online, accessed 01-April-2018].

[5] T. U. K. Archive. Kdd cup 1999 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

[6] S. Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 1–7. ACM, 1999.

[7] D. Colquhoun. The reproducibility of research and the misinterpretation of p-values. *Royal Society Open Science*, 4(12), 2017.

[8] Glossar.ca. Colored neural network.svg. https://commons.wikimedia.org/wiki/File:Colored_neural_network.sv, 2013. [Online, accessed 01-April-2018].

[9] F. V. Jensen. *Bayesian Networks and Decision Graphs*. 2001.

[10] R. Kohavi and G. H. John. The wrapper approach. In *Feature extraction, construction and selection*, pages 33–50. Springer, 1998.

[11] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[12] C. Kruegel, D. Mutz, W. Robertson, and F.Valeur. Bayesian event classification for intrusion detection. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual.*

[13] J.-H. Lee, J.-H. Lee, J.-H. R. Seon-Gyoung Sohn, and T.-M. Chung. Effective value of decision tree with kdd 99 intrusion detection datasets for intrusion detection system. In *10th International Conference on Advanced Communication Technology*, 2008.

[14] K.-L. Li, H.-K. Huang, S.-F. Tian, and Xu. Improving one-class svm for anomaly detection. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, volume 5, pages 3077–3081 Vol.5, Nov 2003.

[15] W. Li. Using genetic algorithm for network intrusion detection. *Proceedings of the United States Department of Energy Cyber Security Group*, 1:1–8, 2004.

[16] Y. Liao and V. Vemuri. Use of k-nearest neighbor classifier for intrusion detection11an earlier version of this paper is to appear in the proceedings of the 11th usenix security symposium, san francisco, ca, august 2002. *Computers & Security*, 21(5):439 – 448, 2002.

[17] R. Lippmann, R. K. Cunningham, D. J. Fried, I. Graf, K. R. Kendall, S. E. Webster, and M. A. Zissman. Results of the 1998 darpa offline intrusion detection evaluation. *Proc. Recent Advances in Intrusion Detection*, 1999.

[18] M. V. Mahoney and P. K. Chan. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In *International Workshop on Recent Advances in Intrusion Detection*, pages 220–237. Springer, 2003.

[19] N. M. Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[20] C. Palagiri. Network-based intrusion detection using neural networks. *Department of Computer Science Rensselaer Polytechnic Institute Troy, New York*, pages 12180–3590, 2002.

[21] A. Patcha and J.-M. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.

[22] P. A. Porras and P. G. Neumann. Emerald: Event monitoring enabling responses to anomalous live disturbances. In *20th NIST-NCSC National Information Systems Security Conference (1997)*.

[23] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek. Distributed anomaly detection in wireless sensor networks. pages 1 – 5, 11 2006.

[24] B. Schoelkop, J. C. Platt, J. Shawe-Taylor, and A. J. Smola. Estimating the support of a high-dimensional distribution. pages 1443–1471, 2001.

[25] T. Shon and J. Moon. A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177(18):3799–3821, 2007.

[26] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 305–316. IEEE, 2010.

[27] G. Stein, B. Chen, A. S. Wu, and K. A. Hua. Decision tree classifier for network intrusion detection with ga-based feature selection. In *Proceedings of the 43rd annual Southeast regional conference-Volume 2*, pages 136–141. ACM, 2005.

[28] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000, 2009.

[29] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000, 2009.

[30] D. Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[31] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[32] E. Yudkowsky. Artificial intelligence as a positive and negative factor in global risk. *Global catastrophic risks*, 1(303):184, 2008.

Spieltheoretische Analyse von Blockchains

Schahed Hadawal
Betreuer: Heiko Niedermayer
Seminar: Future Internet SS18
Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: hadawal@in.tum.de

KURZFASSUNG

Die digitale Währung Bitcoin, welche auf Blockchain basiert, hängt bezüglich ihrer Korrektheit und Stabilität von einer Kombination aus Kryptographie, verteilten Algorithmen und anreizgesteuertem Verhalten ab. Dieses Paper untersucht Kryptowährungen bzw. das Minen von Kryptowährungen als ein Konsens-Spiel und stellt fest, dass das Minen (Generieren) von Bitcoins oder deren Handeln in volatilen Devisenmärkten nach der Spieltheorie eine Art von Spiel darstellt. Beim Mining-Mechanismus erhhält jeder Teilnehmer (Spieler bzw. Miner) Belohnungen in Form von Bitcoin oder Transaktionsgebühren. Dabei muss der Miner entscheiden, ob er sein Block an die längste Transaktionskette anhängt oder an einen anderen Zweig der Transaktionskette. In Anlehnung der Spieltheorie lässt sich dieses "Spiel" - wie jedes andere auch - in einer Entscheidungsmatrix (Bimatrix) runterbrechen. Analogien zwischen diesem Spiel und der Hirschjagd, Taube-Falke Spiel und das Chicken Spiel werden aufgezeigt. Des Weiteren wird begründet, weshalb sich ein Spieler an die vom *Proof-of-Work-Protokoll* festgelegten Regeln halten soll. Insgesamt geht das Paper also der Frage nach, ob sozialwissenschaftliche Probleme auf das Mining der Blockchain anwendbar ist.

Schlüsselworte

Bimatrix, Bitcoin, Blockchain, Chicken-Spiel, Dezentralisierung, Entscheidungsmatrix, Fork, Hash, Hirschjagd, Konsensbildung, Mining, Nash-Gleichgewicht, Nullsummenspiel, Proof-of-Work, Spieltheorie, Taube-Falke Spiel, Zentralisierung

1. EINLEITUNG

Bitcoin ist eine dezentrale elektronische Fiat-Währung[1], die auf Blockchain basiert und mittels Kryptographie und Peer-to-Peer-Technologie implementiert wird. Derzeit befinden sich ca. 17 Millionen Bitcoins im Umlauf[2], welche für eine Vielzahl von Waren und Dienstleistungen gehandelt werden können. Bitcoins werden auf volatilen Devisenmärkten gehandelt: Die globale Marktkapitalisierung beträgt ca. 108 Milliarden Euro (Stand: März 2018); das Handelsvolumen beträgt ca. 4 Milliarden Euro und ein Bitcoin kostet derzeit ca. 9.000 Dollar (Stand: März 2018)[3].[8]
Wie jede andere Währung auch, kann man Bitcoin für das Kaufen und Verkaufen von Waren und Dienstleistungen verwenden. Aber Bitcoin ist mehr als nur eine Währung. Bitcoin ist ein verteilter Algorithmus, der stets einwandfrei funktionieren muss, damit die Währung funktioniert, um beispielsweise einen Konsens darüber zu erzielen, wer welche Münzen besitzt. Der erfolgreiche Betrieb dieser Algorithmen beruht wiederum auf Annahmen, dass die Teilnehmer im System in bestimmter Weise kooperieren. Ob eine Zusammenarbeit sicher ist, hängt davon ab, ob die Anreize der Parteien sie zur Kooperation veranlassen.[5]

Dieses Paper untersucht, ob das Minen von Kryptowährungen mit Bitcoins als eine Art von Spiel hinsichtlich der Spieltheorie angesehen werden kann. Dabei werden in Kapitel 2 formale Definitionen über Blockchain geklärt. Des Weiteren wird u.a. die Funktionsweise von Blockchain erläutert und eine Methode zur Konsensbildung vorgestellt. Kapitel 3 befasst sich mit der Spieltheorie. In diesem Abschnitt werden grundlegende Definition der Spieltheorie erklärt und anhand von Beispielen untermauert. Außerdem wird ein Einblick in die 2x2-Bimatrix-Spiele gewährt, welche eine vereinfachte Darstellung jeglicher "Spiele" liefern kann. Diese Sektion soll zusammen mit Sektion 2 die Grundlage für die philosophische Frage klären, ob das Minen von Kryptowährungen als Spiel betrachtet werden kann. Diese Fragestellung wird in Kapitel 4 diskutiert. Im letzten Kapitel befindet sich eine kurze Zusammenfassung über dieses Paper und gibt ein kurzes Fazit zu dem Mining-Spiel ab.

2. BLOCKCHAIN

Seit Beginn der Digitalisierung von Geschäftsprozessen sind Verlässlichkeit und Vertrauen entscheidende Kernelemente der Digitalisierung. Gleichgültig, ob es sich um organisationsübergreifende Prozesse zwischen kooperierenden Geschäftspartnern in der Lieferkette oder zwischen Verkaufsportalen und Kunden handelt. Im heutigen Internet der Werte stellt sich die Frage nach dem Vertrauen in Transaktionen, die in verschiedene Formen von Werten abgebildet werden können. Traditionell setzen Datenbanken und Prozessmanagementsysteme auf einen zentralistischen Ansatz, der von einer autoritären Person mit einer zentralen Prozesssynchronisation verwaltet wird. Allerdings birgt die Zentralisierung eine beachtliche Anzahl an Risiken mit sich, die u.a. wären: "Leistungsengpässe, Ausfallsicherheit, Authentizität oder in-

[1]Als Fiat-Währung wird jedes Zahlungsmittel bezeichnet, welches keinen inneren Wert besitzt. Fiatgeld unterscheidet sich vom Warengeld wie z.B. Gold, Reis, Salz etc. - https://www.moneyland.ch/de/fiatgeld-definition
[2]https://de.statista.com/statistik/daten/studie/283301/umfrage/gesamtzahl-der-bitcoins-in-umlauf/
[3]https://coinmarketcap.com/de/currencies/bitcoin/

doi: 10.2313/NET-2018-11-1_03

terne und externe Angriffe auf die Integrität."[10]
Der Zentralisierung durch eine Autoritätsperson steht die Dezentralisierung durch die Blockchain gegenüber. Die Innovation von Kryptowährungen gewährleistet die Korrektheit von Transaktionen innerhalb eines Netzwerks und die gemeinsame Konsensfindung zwischen den Netzwerkpartnern. Die Einigkeit über die Transaktionskorrektheit findet nicht zentral, sondern durch eine Konsensfindung zwischen den Partnern statt. [10]

2.1 Definition

Die meisten Menschen kennen Blockchain als Basis von Kryptowährungen wie z.B. "Bitcoin" oder "Litecoin". Blockchain ist eine Datenbank, in der Daten angelegt werden, jedoch nicht mehr nachträglich verändert und gelöscht werden können. Dabei suggeriert der Begriff "Blockchain", dass die Datenbank aus mehreren aufeinanderfolgenden Blöcken besteht, welche Transaktions- bzw. Handänderungsdaten enthalten. Jeder dieser Blöcke wird zeitlich in eine Kette geordnet. Für jeden Block und dessen kompletten Inhalt wird ein Hash erzeugt, welcher eine eindeutige Referenz des Blocks darstellt. Der nächste Block verweist dann jeweils auf den Hash-Wert des vorgängigen Blocks und ein neuer wird jeweils immer vorne an eine bestehende Kette angehängt.[10]

2.1.1 Öffentliche und Private Blockchain

Blockchains können privat oder öffentlich zur Verfügung stehen, wobei der Unterschied darin besteht, welcher Nutzer neue Transaktionen jener Blockchain hinzufügen und validieren darf. Bei einer öffentlichen Blockchain hat jeder Nutzer das Recht, neue Informationen zur Blockchain hinzuzufügen und zu validieren. Bei privaten Blockchains ist der Nutzer auf die Erlaubnis einer Organisation oder Konsortiums angewiesen, erst dann darf er Informationen auf die Blockchain schreiben bzw. validieren. Diese Arbeit beschäftigt sich ausschließlich mit öffentlicher Blockchain, da andernfalls der Rahmen dieser Arbeit gesprengt wird.[10]

2.1.2 Was ist ein Block?

Jeder Block besitzt jeweils einen Header, der aus einer ID, einem Zeitstempel, der Difficulty, der Nonce (number used only once), dem Hash-Wert des vorgängigen Blocks und dem Hash des aktuellen Blocks besteht. Außerdem enthält ein Block ein Datenfeld, das für die Transaktionsdaten reserviert ist. Eine Blockkette wird ungültig, sobald man versucht, die Daten auf einem Block zu verändern. Denn eine Änderung des Blocks führt zur Änderung des Hash-Werts des gesamten Blocks, wodurch die Referenz zum nachfolgenden Block nicht mehr stimmt. Um die Kette wieder gültig zu machen, müsste man für jeden Block in der Kette einen neuen Hash-Wert erzeugen bzw. generieren (Mining), welches allerdings sehr zeit- und rechenintensiv ist.[10]

2.1.3 Was ist ein Hash?

Ein Hash ist eine Zahl, die einer beliebigen Zeichenkette einen eindeutigen Wert zuordnet, d.h. ein beliebiger Datenraum wird auf Daten fester Größe mittels Hashfunktion abgebildet. Wird ein Zeichen in der Zeichenfolge verändert oder hinzugefügt, sei es auch nur ein einzelnes Zeichen, so führt das zu einem neuen Hash-Wert, der durch einen beliebigen Computer sehr schnell erzeugt werden kann. Auf diese Weise kann man prüfen, ob ein Hash-Wert zu einer bestimmten Zeichenkette gehört oder nicht.[6]

2.1.4 Was bedeutet Mining?

Es sind die Miner, die entscheiden, zu welchem vorherigen Block ein neuer Block verkettet wird uns somit eine einzelne Kette oder Fork (Spaltung der Transaktionskette in mehreren Ketten) entstehen lässt. Miner sind an das *Proof-of-Work*-Protokoll (Abschnitt 2.3) gebunden und sie versuchen zu jedem Zeitpunkt, einen neuen Transaktionsblock zu generieren und zu validieren. Dieses Konzept der Validierung nennt man Mining. Derjenige Miner, der den nächsten Block vorschlagen darf, wird ausgelost, somit wird gewährleistet, dass alle Miner sich beim Validieren der Blöcke abwechseln und zu keinem Zeitpunkt ein bestimmter Miner den gesamten Verifizierungsprozess unter seine Kontrolle bringt.[4]
Welcher Anreiz wird den Minern geboten? Jeder Miner, der dabei hilft ein *Proof-of-Work*-Problem (z.B. in der Handelsfinanzierung) zu lösen, indem er erfolgreich einen Transaktionsblock generiert und validiert, wird mit neu geschaffenen Einheiten der Währung belohnt (z.B. 12.5 Bitcoin pro Block - Stand: 2018 [4]). Des Weiteren erhält jeder Miner eine *Transaction fee*, welche mit dem Block generiert wird.[4]

Im folgenden wird die Funktionsweise der Blockchain bzw. das Mining genauer betrachtet.

2.2 Funktionsweise

Wie bereits oben erwähnt, gibt es zwei Kernelemente der Technologie, nämlich die Codierung von Transaktionen durch Hashing und die Konsensfindung über die Korrektheit von Transaktionen. Beim ersten Kernelement ist eine Abbildung unterschiedlicher Zeichenketten auf denselben Code ausgeschlossen (Kollisionsfreiheit). Nach einer formalen Überprüfung der Transaktion versuchen die Netzwerkpartner über die Transaktion einen Konsens zu finden. Falls die Partner für die Transaktion einen Konsens gefunden haben, so wird sie im Netz verteilt und in die globale Blockchain aufgenommen. Abbildung 1 zeigt eine vereinfachte schematische Darstellung von der Iniitierung bis zum Abschluss der Transaktion.[5]

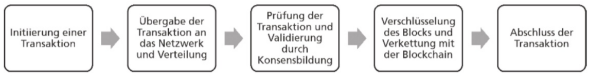


Abb.1: Funktionsweise einer Blockchain (Fraunhofer FIT)[10]

Zuerst wird jede Transaktion von einem Sender generiert und digital signiert (Initiierung). Eine Transaktion kann dabei z.B. die Überweisung einer Kryptowährung oder die Registrierung eines Dokuments sein. Anschließend wird die Transaktion an das Netzwerk übermittelt und an die beteiligten Knoten im Netz verteilt. Dabei überprüft jeder Knoten im Netzwerk die Gültigkeit der Transaktion und versucht dadurch einen Konsens zu finden. Bevor es zum Abschluss der Transaktion kommt, wird die Transaktion in einem Block

doi: 10.2313/NET-2018-11-1_03

gespeichert und durch Hashfunktionen in ein standardisiertes Format überführt, d.h. jede einzelne Transaktion wird in Hashwerte codiert und hierarchisch zu einer Blockkette verdichtet (Hash oder Merkle-Baum). Diese Codierung ist sicher gegenüber Manipulationsversuchen, da mögliche Änderungen der Transaktionen den Hashwert des Blocks ändern würden. Dadurch wäre der Hashbaum nicht mehr konsistent. Abbildung 2 verdeutlicht den vorherig genannten Ablauf der Transaktion mit Hilfe eines konkreten Beispiels, in der Partner A Geld an Partner B überweisen möchte. Mittels eines Computers überweist Partner A einen Geldbetrag an Partner B. Dabei wird die Transaktion einem Block angefügt und an beide Netzwerkteilnehmer weitergeleitet. Anschließend wird die Transaktion von beiden Netzwerkteilnehmern ge-nehmigt und validiert, so dass dieser Block an die bereits bestehende Blockkette (Blockchain) angehängt wird. Wie gewünscht wird das Geld am Ende überwiesen und die Transaktion erfolgreich abgeschlossen.[9]

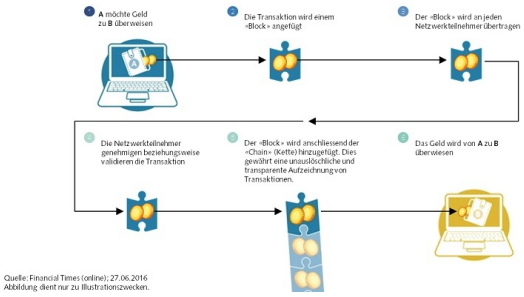


Abb.2: Beispiel für die Funktionsweise einer Blockchain

Damit ein Block in die Blockkette (Blockchain) aufgenommen werden kann, muss ein Verfahren zur Konsensbildung absolviert werden.

2.3 Methoden der Konsensbildung - Proof-of-Work

Ein wesentlicher Grundpfeiler des Blockchains-Konzepts ist die Konsensbildung. In der Konsensbildung werden Transaktionen so validiert, dass eine Übereinstimmung über die als gültig anzuerkennende Transaktion gefunden werden kann. Die Konsensbildung sorgt dafür, dass die gespeicherte Aussage anerkannt wird und zukünftig nicht mehr veränderbar ist.[9]
Das bekannteste Verfahrung zur Konsensbildung ist der Proof-of-Work der Bitcoin Blockchain. Dieses Verfahren basiert auf einen asymmetrischen Ansatz, bei dem ein Dienstnutzer Arbeit leisten muss, die von einem Dienstanbieter ohne große Mühe überprüft werden kann.[6] Im Kontext der Blockchain entsprechen die Nutzer den Minern, welche den Prof-of-Work aufwändig berechnen. Demenstprechend stellen die Anbieter alle Knoten dar, die ohne großen Aufwand prüfen, ob der Miner den Proof-of-Work ordnungsgemäß berechnet hat.[9]
Wie bereits in Kapitel 2.1.4 erwähnt, ist das Ziel des Proof-of-Work-Algorithmus, eine Zahl (Nonce = number used only once) zu finden, die in Kombination mit dem neu generierten Block, welcher an die bestehende Blockchain angehängt wird, einen Hashwert ergibt, der eine bestimmte Bedingung erfüllt. Eine solche Bedingung könnte sein, dass der zu findende Wert aus einer bestimmten Anzahl an führenden Nullen besteht (Kapitel 2.1.4). Hashfunktionen sind Einwegfunktionen, daher kann diese Zahl nur durch Brute-Force (Ausprobieren) gefunden werden.[10]
Beim Proof-of-Work-Verfahren steht die Rechenleistung der Knoten im Vordergrund, wenn es darum geht, einen passenden Nonce-Wert zu finden. Jeder Miner wird für das Finden eines solchen Wertes mit Bitcoins (Kryptowährung - virtuelle Währ-ung) belohnt, daher entsteht ein Wettbewerb, bei dem jeder Miner versucht, seine Rechenleistung zu verbessern bzw. zu erhöhen. Dies hat zur Folge, dass die Zeitdauer für das Auffinden von gültigen Nonce-Werten reduziert wird. Würden sich die Zeitintervalle verkürzen, in denen neue Blöcke erzeugt werden, dann würde sich die Geldmenge zu schnell erhöhen. Daher wird die Schwierigkeit des Auffindes von jenen Werten erhöht, wenn sich die Rechenkapazität verkürzt. Für die Miner bedeutet das: erhöhter Aufwand für geringe Erfolgsaussichten.[9]
Neben der Rechenleistung gibt es auch speicher- oder netzwerkbasierte Proof-of-Work-Verfahren. Beim speicherbasierten Proof-of-Work-Verfahren wird das Finden von Nonce-Werten durch eine entsprechende Anzahl von Speicherzugriffen gelöst [3]. Beim netzwerkbasierten Proof-of-Work-Verfahren wird das Rätsel durch Kommunikation mit anderen Netzknoten gelöst. Zum Beispiel werden Informationen gesammelt, die für das Auffinden des Wertes notwendig sind.[2]
Das Proof-of-Work verfahren eignet sich insbesondere für öffentliche Blockchain-Netzwerke, da es keiner Zugangsbeschränkung unterliegt. Für private Blockchains eignet sich das Proof-of-Stake-Verfahren, bei dem die Knoten, welche einen neuen Block validieren können, nach ihren Anteilen an der Kryptowährung oder über ein Zufallsverfahren ausgewählt werden.[1]

2.4 Was ist ein Fork?

Ein Fork in der Blockchain beschreibt ein Vorgang in der Kryptowährung, bei dem ein Projekt durch Modifikation des Quellcodes die ursprüngliche Blockchain in eine neue Blockchain abspaltet. Graphisch gesehen entwickelt sich aus der Transaktionskette ein "Transaktionsbaum". In Abb. 3 entwickeln sich ab dem Block X parallel zwei Blockchains. Die ältere Blockchain wird weiterhin von einigen Benutzern der älteren Version mit Mining versorgt. Dabei ignorieren die älteren Clients den neuen Block X. Neue User widerum versorgen die abgespaltene Blockchain X mit weiteren Blöcken, sodass sie wächst. Unabhängig davon, wie lange die einzelnen Transaktionsketten werden können die neuen User zwischen den Blöcken hin und her wechseln (minen).[4]

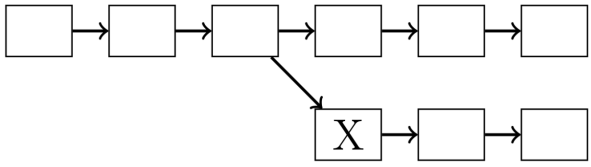


Abb.3: Fork in der Blockchain[4]

[4]https://bitcoinblog.de/2015/06/15/was-passiert-bei-einem-hard-fork/

doi: 10.2313/NET-2018-11-1_03

2.4.1 Warum treten Forks in der Blockchain auf?

Bei einem Fork treten immer wieder Änderungen in der Blockchain auf, die nicht kompatibel mit den alten Blöcken sind. Dies führt hin und wieder dazu, Erweiterungen an den Protokollen vorzunehmen. Die Änderungen können verschiedener Natur sein: Von kleinen Ergänzungen bis hin zu neuen Features, wie z.B. die Blockgrößenerweiterung. Da die Blockchain ein Open-Source Projekt ist und für jeden frei zugänglich ist, kommt es häufig zu Uneinigkeiten in der Community. Falls die Community zu keiner Einigung gelangt, so entstehen zwei Gruppen, die unterschiedliche Ziele erreichen wollen und somit zu einer Blockchainspaltung. Dies kann mitunter zur Entstehung neuer Kryptowährungen führen. Ein bekanntes Bespiel hierfür ist das Bitcoin Gold[6], welches sich von der Bitcoin Blockchain abgespalten hat, um eine eigenständige Kryptowährung zu werden.[4]

2.4.2 Welche Konsequenzen resultieren aus den Forks?

Viele Parteien sind an der Blockchainentwicklung und -benutzung beteiligt. Dies erfordert ihre Teilnahme an allen Änderungen der Blockchain.[4]

1. Szenario: die neue Software wird bei vielen Endnutzern, Börsenbetreibern und Händlern verwendet. Allerdings weigern sich die Miner die neuen und größeren Block zu veröffentlichen und verhindern die Transaktionsbestätigung.

2. Szenario: Alle Miner benutzen die neueste Version der Blockchain. Die Endnutzer, Börsenbetreiber und Händ-ler sind aber noch nicht auf die neue Version umgestiegen, nutzen noch die alte Version und ignorieren die großen Blöcke. Dies führt dazu, dass keine weiteren Transaktionen mehr bestätigt werden können, da keine neuen Blöcke mehr gefunden werden können. Für die Miner ist dieses Szenario ebenso ungünstig, da sie ihre Belohnung nicht erhalten würden. Die Miner sind dementsprechend daran interessiert, diese Situation zu vermeiden und sollten sich mit den Endnutzern abstimmen.

3. Szenario: In diesem Szenario wird die neue Version nur von wenigen Usern und Minern verwendet. Die generierten Blöcke werden zwar innerhalb dieser "Gruppe" anerkannt und validiert, aber vom restlichen Netzwerk ignoriert. Die Belohnung wird dementsprechend klein ausfallen, da es sich um kleine Gruppen handelt.

4. Szenario: Analog zum vorherigen Szenario ist die umgekehrte Variante ebenfalls möglich, dass eine große Mehrheit die neue Version benutzt. Die erste Blockchain wird von den Usern der alten Version ignoriert. Dies würde allerdings keine größeren Konsequenzen zur Folge haben, da alle Endnutzer, Händler und Börsen ebenso die neue Version nutzen würden.

5. Szenario: In diesem Szenario sind beide Blockchains nach der Abspaltung aktiv. Dies hat zur Konsequenz, dass beide Blockchains unterschiedliche Informationen enthalten, da Miner frei entscheiden können, welche Transaktionskette verlängert werden soll. Die eine Blockchain könnte das Geld für ein gekauftes Auto enthalten während in der anderen Blockchain das Geld noch in der Wallet[7] des Käufers steckt. Durch geschickte Manipulationen kann man so ein Szenario nach einem Fork auch erzwingen.

[6]https://bitcoingold.org

Diese Szenarien verdeutlichen, wie wichtig es ist, einen Konsens zu erzeugen. Wie erzeugen alle Beteiligten einen Konsens?

2.4.3 Konsensbildung

Kroll et al (2013) definiert drei Typen von Konsens:

1. Konsens über Regeln: Alle Miner müssen sich auf Kriterien einigen, um zu bestimmen, welche Transaktionen gültig sind. Nur gültige Transaktionen sollten im Bitcoin-Protokoll vermerkt werden. Jedoch erfordert dies eine Übereinstimmung darüber, wie die Gültigkeit zu bestimmen ist.

2. Konsens über den Status: Die Miner müssen sich darauf einigen, welche Transaktionen tatsächlich stattgefunden haben, d.h. sie müssen ein gemeinsames Verständnis dafür haben, wer zu welcher Zeit welche Münzen besitzt.

3. Konsens über den Bitcoinwert: Außerdem müssen alle Beteiligten akzeptieren, dass Bitcoins einen Wert besitzen, so dass Endnutzer bereit sind, Bitcoins in Zahlungen zu akzeptieren und zu verwenden.

Jeder dieser Punkte ist abhängig von den jeweils anderen. Diese Konsensprozesse sind ein sozialer Prozess in denen die Teilnehmer sich darüber einig werden müssen, was erlaubt ist und was nicht. Die Regeln müssen dann in der Software, die jeder Teilnehmer verwendet, kodiert werden können.

Bevor der Frage nachgegangen werden kann, ob es sich bei Mining von Bitcoins um ein Spiel im Sinne der Spieltheorie handelt, müssen ersteinmal die Begriffe rund um die Spieltheorie näher betrachtet werden.

3. SPIELTHEORIE

Damit der Mensch seine Handlungsoptionen und die daraus resultierenden Folgen im Voraus analysieren kann, muss er mit Hilfe eines im Geist konstruierten Abbildes der realen Umgebung arbeiten. Alle Spieltheorien basieren auf dieser Tatsache. Nur Dank dieser Planungsfähigkeit ist ein soziales Miteinander auf unterschiedlichen Ebenen (z.B. in polit. Verhandlungen oder wirtschaftlichen Systemen wie Märkte etc.) möglich. Eine weitere Grundlage dieser Theorien ist die Annahme, dass alle Teilnehmer eines Spiels ein möglichst gutes Ergebnis für sich selbst erzielen wollen, d.h. jeder Spieler (Player) ist daran interessiert, sein Gewinn zu maximieren.

Im folgenden wird die Spieltheorie im Allgemeinen definiert.

[7]Eine Wallet ist eine virtuelle Geldbörse, in der man seine Kryptowährung anlegen kann

doi: 10.2313/NET-2018-11-1_03

Anschließend werden einige Begriffe wie z.B. die Entscheidungsmatrix, Bimatrix und einige global anwendbare Spiele vorgestellt.

3.1 Definitionen

Spieltheorie ist eine Theorie sozialer Interaktion, d.h. ein Spiel ist nichts anderes als eine soziale Interaktion zwischen n Spielern (n-Personen Spiele). Im Vordergrund der Spieltheorie steht die Entscheidungssituation, bei dem das Ergebnis einer Entscheidung nicht nur von dem Entscheider (Spieler), sondern auch von dem Verhalten der anderen Entscheider (Gegenpieler) abhängt. Solche Entscheidungssituationen sind beispielsweise in Gesellschaftsspielen wieder zufinden. Allerdings hat die Spieltheorie weniger mit Gesellschaftsspielen zu tun, sondern ist lediglich im übertragenen Sinn eine Basis für jedes Spiel und relevant für Entscheidungen im täglichen Leben.[11]
In der Spieltheorie spielt man immer gegen die Natur, die in unterschiedlichen Formen z.B. konkurrente Mitspieler oder Zufall, aber auch als die Natur selbst auftreten kann. Ein Beispiel, in der die Natur als Gegenspieler im Spiel dargestellt wird, ist die landwirtschaftliche Produktion, in der das Wetter viele Umweltzustände bestimmen kann.[11]
Spieltheorien betrachten ebenfalls strategische Spiele, die wiederum auch Zufallselemente enthalten können. Dabei ist ein strategisches Spiel, auch strategische Interaktion/Konflikt genannt, eine Entscheidungssituation, in der mehrere vernunftbegabter Entscheider Einfluss auf das Resultat haben und ihre eigenen Interessen verfolgen. In erster Linie wird dabei versucht, für ein Problem (gegebenes Spiel), das als strategisches Konflikt abgebildet ist, eine Lösung zu ermitteln. Dabei ist die Lösung des Spiels ein Vorschlag bzw. Anleitung, wie das Spiel zu spielen sei. Die Lösung eines Spiels wird aus den Eigenschaften des Spiels hergeleitet und nicht aufgrund von psychologischer Überlegungen. Im Zusammenhang mit strategischen Spielen kann eine Strategie auch mit Entscheidungsalternative übersetzt werden. [7]

In der Spieltheorie lässt sich ein Spiel in *kooperative* und *nicht-kooperative* Spiele unterteilen. Diese Begrifflichkeiten werden im folgenden Abschnitt erklärt.

3.1.1 Kooperative und nicht-kooperative Spieltheorie

In der *kooperativen Spieltheorie* wird versucht, Situationen zu untersuchen, in denen sich Spieler mittels natürlicher Sprache verständigen. Bei einem Spiel hat jeder einzelne Spieler einen Einfluss auf den Spielausgang, daher verhandeln die Spieler zusammen darüber, welches Spielergebnis gemeinsam realisiert werden soll. Um das gemeinsam vereinbarte Spielergebnis zu gewährleisten, können die Spieler einen Vertrag abschließen, der jeden Spieler dazu verpflichtet, sich auch so zu entscheiden, wie abgesprochen. Die Einhaltung des Vertrages ist also im Rahmen der Spielregeln gesichert und extern vorgegeben. In dieser Form der Spieltheorie wird der Gewinn für alle beteiligten umso größer, je besser die Player zusammenarbeiten.[11]
In der *nicht-kooperativen Spieltheorie* wird versucht, Situationen zu untersuchen, in denen die Spieler vollkommen getrennt voneinander sind und nicht in natürlicher Sprache miteinander kommunizieren und bindende Verträge eingehen können. Die einzige Möglichkeit miteinander zu kommunizieren sind die Entscheidungen, die jeder Spieler während dem Spiel trifft. Dabei stellen die Züge eine Art stilisierterSprache dar. Des Weiteren hat keiner der Spieler einen Einblick über die Psychologie des Gegenspielers. Die Rationalität ist also die einzige psychologische Komponente, von der jeder Spieler ausgehen kann, dass der Gegenspieler sie besitzt. In dieser Form der Spieltheorie versucht jeder Player das bestmögliche Ergebnis für sich selbst zu erhalten, auch auf Kosten der Gegenspieler. Daneben zielen sie darauf ab, nicht nur den absoluten, sondern auch den relativen Gewinn - d.h. den Gewinn im Verhältnis zu den Ergebnissen der anderen - zu maximieren.[11]
Eine weitere Form, die aber eigentlich Teil der vorher genannten Formen bildet, ist das Nullsummenspiel. Nullsummenspiel ist ein Spiel, in dem einige der Player das gewinnen, was andere verlieren. Dabei addiert sich die Gesamtsumme aus Gewinn und Verlust aller Spieler immer zu null.[11]

Eine einfache und wohlbekannte Methode, mit dem ein Problem bzw. gegebenes Spiel strukturiert werden kann, ist die Entscheidungsmatrix, die im nächsten Abschnitt anhand eines Beispiels erläutert wird.

3.1.2 Entscheidungsmatrix

Angenommen ein Geschäftsführer (erster Spieler) eines Unternehmens muss für das folgende Geschäftsjahr festlegen, wie stark sein Unternehmen sich auf dem Markt engagieren will. Dafür muss er das Marketing- und Investitionsbudget im Vornherein festlegen. Dem Geschäftsführer stehen vier Engagement-Stufen zur Verfügung, nämlich von 1 (geringes Engagement) bis 4 (sehr starkes Engagement). Der Erfolg seines Unternehmens hängt von der Konkurrenzlage (zweiter Spieler) ab, die entweder günstig, normal oder ungünstig sein kann. Abbildung 4 zeigt die bisher aufgezählten Daten, die in der Matrix eingetragen sind. Die darin enthaltenen Zahlen stellen den Gewinn für das Unternehmen dar, sind frei erfunden und dienen zur Veranschaulichung des Beispiels. Die Zeilen der Matrix repräsentieren das Engagement des Unternehmens auf dem Markt, während die Spalten die Konkurrenzlage widerspiegelt.[11]

Konkurrenzlage: Engagement auf dem Markt:	günstig	normal	ungünstig
1 (gering)	5	3	1
2 (mittel)	14	10	0
3 (stark)	30	5	-5
4 (sehr stark)	12	9	-9

Abb.4: Darstellung einer Entscheidungsmatrix[11]

Die zentrale Frage, die sich hier der Geschäftsführer (Spieler) stellt, ist wie er sich verhalten muss, damit er seinen Gewinn maximieren kann. In der Entscheidungstheorie gibt es zahlreiche Strategien (Lösungsvorschläge), um dieses Ziel zu erreichen.
Im folgenden werden zwei davon aufgelistet. Der Geschäftsführer kann u.a. von den folgenden zwei Strategien eine auswählen:

doi: 10.2313/NET-2018-11-1_03

1. Der Unternehmer könnte die Eintrittswahrscheinlichkeiten für die verschiedenen Konkurrenzlagen schätzen und die Handlungsalternative wählen, für die der Erwartungsgewinn maximal ist.
2. Er wählt das Maximal-Prinzip, bei der der schlechteste mögliche Gewinn maximal ist.

Angenommen das Unternehmen agiert gering auf dem Markt und die Konkurrenzlage ist hoch, sprich ungünstig, dann erwirtschaftet das Unternehmen einen Gewinn von einer Einheit. Auf der anderen Seite erwirtschaftet der Geschäftsführer für sein Unternehmen einen Gewinn von 30 Einheiten, wenn er sich am Markt stark beteiligt und die Konkurrenzlage günstig steht.

Anmerkung: Logischerweise müsste der Unternehmer einen höheren Gewinn erzielen, falls er sich "sehr stark" am Markt engagiert und die Konkurrenzlage günstig steht (hier Gewinn von 12 Einheiten, Maximum liegt aber bei 30 Einheiten). Jedoch wurden hier bewusst die Zahlen so eingetragen, dass dieser "logische" Fall nicht eintritt, um zu veranschaulichen, dass ein Spiel immer noch Regeln definiert und unabhängig von Logik und Rationalität agiert.

Wie sollte sich der Geschäftsführer nun entscheiden?

Für den Spieler ist es nicht leicht, sich für eine Alternative zu entscheiden, jedoch lassen sich Alternativen leicht aussondern. Der Gewinn bei Alternative 4 (sehr starkes Engagement) ist bei jedem Umweltzustand, also bei jedem Verhalten des Gegenspielers, schlechter als der Gewinn bei Alternative 2 (mittleres Engagement). In anderen Worten: Egal was passiert, Alternative 2 schüttet - aus Sicht des Geschäftsführers - einen höheren Gewinn aus, als Alternative 4. Somit wird Alternative 4 aus der Matrix gestrichen, da es für den Geschäftsführer keinen Grund gibt, sich jemals "sehr stark" zu engagieren. Die selbe Entscheidungsregel gilt auch für Alternative 1, die von Alternative 2 überboten und somit dominiert wird. Daher wird auch Alternative 1 von der Entscheidungstheorie eliminiert.

Wie der Spieler (Geschäftsführer) sich letztendlich im weiteren Verlauf entscheidet, hängt von seiner Strategie (Lösung) bzw. Ziel und von den Regeln, die das Spiel definiert, ab. Auf die Lösung wird nicht eingegangen, da der Rahmen dieser Arbeit gesprengt wird.

3.1.3 Dominierte Strategien und Nash-Gleichgewicht

Wenn es für einen Spieler vorteilhaft ist, immer die selbe Strategie zu wählen, unabhängig davon, welche Strategie die anderen Spieler wählen, dann spricht man von einer dominanten Strategie.[11] Damit eng verknüpft ist das Nash-Gleichgewicht.
Das Nash-Gleichgewicht (NGG) beschreibt in nicht kooperativen Spielen eine Kombination von Strategien, bei denen es für keinen der Spieler sinnvoll ist, von seiner Strategie als einziger abzuweichen, um seinen Gewinn zu maximieren. D.h. in einem NGG bereut jeder Spieler auch im Nachhinhein seine Strategiewahl nicht und würde seine Entscheidung wieder genau so treffen.[11]

3.2 2 x 2 - Bimatrix Spiele

Eine Reduktion der Entscheidungsmatrix (Kapitel 3.1.2) ist die 2x2-Bimatrix, in der zwei Spieler ein Spiel spielen, wobei jeder der Player jeweils zwei Handlungsmöglichkeiten hat.[11] Wie man diese Form der Matrix liest und anwendet, wird anhand folgender Grafik (Abb. 5) welche das Beispiel verdeutlichen soll, erläutert.

		Spielerin B(erta): links	rechts
Spieler A(nton):	oben	(2, 1)	(9, 0)
	unten	(1, -1)	(9, 8)

Abb.5: Darstellung einer Bimatrix[11]

Wie bereits oben erwähnt, stellt die Bimatrix zwei Player auf, die jeweils zwei Handlungsmöglichkeiten haben. Die Player können nicht kontrollieren, welche Strategie sein Gegenüber wählt. Die Lösung eines Spiels ist dabei eine Kombination von Verhaltensweisen der Spieler, die als rational gerechtfertigt werden kann [11]. Spieler A (Anton) und Spieler B (Berta) befinden sich in einer Quizsendung und müssen unabhängig voneinander und gleichzeitig auf einen Knopf drücken. Player A hat die Möglichkeit zwischen den Knöpfen *oben* und *unten* zu drücken, während Player B zwischen den Knöpfen *links* und *rechts* drücken kann. Der jeweilige Gewinn bzw. Verlust beider Spieler ist in einem Tupel (a,b) in der Matrix eingetragen. Dabei steht a für den Gewinn/Verlust von Spieler A. Das selbe gilt analog für Spieler B. Wenn Anton beispielsweise den Knopf *unten* drückt und Berta auf *links*, dann gewinnt er in dem Spiel eine Einheit und sie verliert eine Einheit. Die Player erfahren die Knopfwahl des Gegenübers erst nachdem beide unanbhängig und gleichzeitig auf einen Knopf gedrückt haben.[11]

Für welche Alternative sollten sich die Spieler entscheiden?

In der Spieltheorie gilt generell die Annahme, dass jeder Player nur an seinem Gewinn interessiert ist [11]. Betrachtet man die Entscheidungsalternativen, die sich für Berta ergeben, so sieht man folgendes:

1. Sollte Anton *oben* wählen, so wäre es für Berta am gewinnbringendsten den *linken* Knopf zu betätigen
2. Wählt Anton *unten*, so sollte sich Berta für den *rechten* Knopf entscheiden, da dieser - hinsichtlich des Gewinns - am sinnvollsten erscheint.

Aus den zwei genannten Punkten lässt sich nur eines rückschließen: Spielerin B hat keine dominierte Strategie und muss rational handeln. Spieler A muss sich logischerweise und unabhängig von der Wahl, die Spieler B trifft, für den Knopf *oben* entscheiden, da dieser Knopf den *unteren* Knopf dominiert, d.h. der Gewinn von Knopf *oben* ist mindestens

doi: 10.2313/NET-2018-11-1_03

gleichbringend wie der Knopf *unten*. Daher kann Spieler A die zweite Zeile aus der Matrix streichen und somit seine Entscheidungsalternative festlegen. Berta handelt rational und erkennt diese Strategie, womit sie nur noch ihre Wahl auf die reduzierte Matrix anwenden muss. Folglich fällt ihre Wahl selbstverständlich auf den Knopf *links*. Damit ist die Lösung nach dem Dominanzkriterium klar, die Spieler müssen sich für die Kombination (*oben*, *links*) entscheiden. Jeweils beide Player würden einen höheren Gewinn erwirtschaften, sofern sich beide auf die Kombination (*unten*, *rechts*) einigen könnten. Jedoch treten in solchen Spielen meist immer Komplikationen auf: Die Player können nicht kontrollieren, welche Strategie sein Gegenüber wählt.

Im folgenden werden drei Spiele vorgestellt, die in der Sozialwissenschaft häufig verwendet werden. Diese Spiele werden später einen Bezug zu Kapitel 4 darstellen und dort wieder aufgegriffen.

3.2.1 Hirschjagd (Stag Hunt)

Die Hirschjagd geht auf J.J. Rousseau zurück und beschreibt ein Spiel, bei dem es um eine *win-win* Situation geht. In erster Linie beschreibt das Spiel einen Konflikt zwischen Sicherheit und soziale Kooperation, indem die Widersprüche des individuellen Handelns innerhalb der Gemeinschaft aufgezeigt werden, welche letztendlich zur Einrichtung von sozialen Zwangmitteln führt, die eine Kooperation der Mitglieder erfordert und sichert.[12]
Rousseau lässt zwei Spieler (Jäger 1, Jäger 2) auf die Jagd gehen. Im Jagdgebiet kann man zwei Tiere jagen: Hirsche (stag) und Hasen (rabbit). Dabei ist folgendes zu beachten:

1. Beide Spieler müssen unabhängig voneinander entscheiden, welches Tier sie jagen
2. Den Hirsch können nur beide gemeinsam erlegen bzw. jagen, einen Hasen kann jeder alleine jagen. Falls einer der beiden einen Hasen jagen sollte, so entkommt der Hirsch.
3. Beide Tiere gleichzeitig zu jagen ist unmöglich

Das zugehörige Spiel sieht in 2x2-Bimatrix-Form folgendermaßen aus:

		Jägerin 2:	
		Hirsch	Hase
	Hirsch	(5, 5)	(0, 1)
Jäger 1:	Hase	(1, 0)	(1, 1)

Abb.6: Gewinne und Verluste im Hirschjagdspiel in 2x2-Bimatrix-Form[9]

[9]http://www.spieltheorie.de/spieltheorie-grundlagen/win-win-hirschjagd/)

Wie man an Abbildung 6 sehen kann, ergibt sich ein Nullsummenspiel, wenn einer der beiden einen Hasen und der andere jeweils einen Hirsch jagt. Auf der anderen Seite entsteht ein Nash-Gleichgewicht (Kap. **??**) wenn beide Spieler den Hirsch jagen, als auch wenn beide einen Hasen jagen. Allerdings gewinnen beide Spieler beim Hirsch eine Auszahlung von 5 Einheiten, beim Hasen dagegen nur eine Auszahlung von einer Einheit. Die Zusammenarbeit lohnt sich also für beide und es gibt eigentlich gar keinen Grund für einen Interessenkonflikt.
Jedoch gibt es einen Haken: jeder Spieler versucht in erster Linie seinen Gewinn zu maximieren. Allerdings achtet der Spieler auch darauf, sein Minimum zu maximieren. In diesem Spiel beträgt das Minimum null, falls man einen Hasen jagt und eins, falls man einen Hirsch jagt, vorausgesetzt, der Mitspieler jagt das jeweils andere Tier. Rational empfiehlt es sich hier, den Hasen zu jagen, weil man da gegen leere Taschen abgesichert ist. Diese Strategie ist aber nur sinnvoll, wenn es sich um ein Nullsummenspiel handeln würde. In diesem Spiel ist aber die Zusammenarbeit und Kommunikation möglich, daher sollten beide Parteien den Hirsch jagen.[12]

3.2.2 Taube-Falke Spiel (Hawk Dove Game)

In diesem Spiel gelangen zwei Tiere gleichzeitig zu einer Ressource, z.B. ein günstiges Gebiet für ein Revier. Dieses Revier kann allerdings nur von einem der beiden genutzt werden. Hier repräsentieren die Tiere die Spieler A und B und haben zwei Auswahlmöglichkeiten: Flucht (Taube) oder Kampf (Falke). Jedes Tier nimmt zu Beginn eine Drohposition ein, damit der Mitspieler nicht sofort erkennt, ob sein Gegenüber kämpfen oder flüchten will. Wenn ein Tier sich entscheidet, die Strategie Taube (dove) zu wählen, so flüchtet es sofort. Entscheidet das Tier sich für den Falken (hawk), so nimmt es nicht nur die Drohposition ein, sondern kämpft - falls das andere Tier sich auch für den Falken entscheidet - bis aufs Blut.[11]
Die Tiere müssen sich wegen der Eingangssituation (Drohposition) sofort und unabhängig von der Strategie des Gegenspielers entscheiden. Dadurch kann sich keiner der Player auf das Verhalten des anderen einstellen.[11]

Das zugehörige Spiel sieht in 2x2-Bimatrix-Form folgendermaßen aus:

		Tier B:	
		Taube	Falke
Tier A:	Taube	(1,1)	**(0,2)**
	Falke	**(2,0)**	(-10,-10)

Abb.7: Gewinne und Verluste im Taube-Falke-Spiel in 2x2-Bimatrix-Form[11]

Treffen zwei Tauben aufeinander, so erhält jeder der beiden Tauben das Revier mit einer Wahrscheinlichkeit von $\frac{1}{2}$:

doi: 10.2313/NET-2018-11-1_03

damit beträgt die erwartete Auszahlung: $\frac{1}{2} * 2 = 1$. Treffen zwei Falken aufeinander, so erleiden beide einen Verlust von -10 Einheiten und sterben. Diese zwei Fälle sind dem Nash-Gleichgewicht zugeordnet.[11]
Stößt eine Taube auf einen Falken, so geht die Taube leer aus (0 Einheiten) und der Falke enthält 2 Einheiten (Nullsummenspiel).
Wie löst ein Spieler dieses Spiel zu seinen Gunsten? In diesem Spiel wäre es ratsam nach dem Prinzip *"der Klügere gibt nach"* zu handeln. Man geht zwar nach dem Nullsummenspiel eventuell das Risiko ein, leer auszugehen, jedoch hat man vielleicht Glück und der Gegenspieler wählt ebenfalls die selbe Strategie, sodass man immerhin noch eine Einheit erhält. Andernfalls kommt es zu einem bitteren Kampf, in dem im schlimmsten Fall beide -10 Einheiten verlieren.[11]
Aus diesem Modell geht hervor, dass Tiere derselben Art sich nicht zerfleischen. Dies liegt aber nicht an der inneren Harmonie der Natur, sondern aus Angst, dass der Gegner auch *Falke* spielen könnte und es somit zu einem Zweikampf kommt, der einen immensen Schaden verursacht. In der realen Welt könnte dieses Modell vielleicht erklären, warum der Kalte Krieg nicht in einen Atomkrieg geendet hat.[11]

3.2.3 *Chicken Spiel (Game of Chicken)*

Bei einer Mutprobe fahren Spieler A und Spieler B mit ihren Autos direkt aufeinander zu. Dabei hat jeder der beiden Spieler zwei Optionen zur Auswahl: Kurs halten oder ausweichen. Derjenige Fahrer, der zuerst ausweicht, ist das "Chicken", bleibt aber am Leben. Der andere Spieler gilt dann als "Held".[11]

Das Spiel sieht in 2x2-Bimatrix-Form folgendermaßen aus:

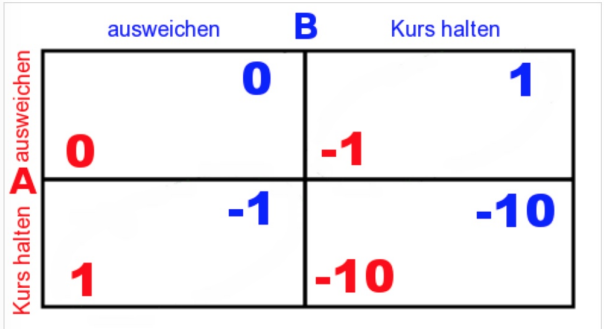


Abb.8: Gewinne und Verluste im Chicken-Spiel in 2x2-Bimatrix-Form[13]

Erweisen sich beide Player als irrational stur, so kommt es zu einer Kollision und beide sterben (-10 Einheiten). Dies steht nicht im Interesse beider Spieler. Wenn beide Fahrer entweder Kurs halten oder ausweichen, so entsteht hier wieder ein Nash-Gleichgewicht (Kap. **??**). Falls einer der Fahrer kurs hält und der andere ausweicht, so verliert der eine einen Minuspunkt (Chicken), der andere gewinnt einen Punkt (Held) und beide überleben.[11]
Hier existiert keine dominante Strategie, da Weiterfahren zum bestmöglichen, aber auch zum allerschlechtesten Ergebnis führen kann. Wiederum besteht Unsicherheit über das Verhalten des anderen, weswegen auch Wahrscheinlichkeitsüberlegungen in die Entscheidung einfließen können. Für beide Spieler ist klar, dass sie es doch bevorzugen als Feigling dazustehen. Weichen beide gleichzeitig aus, gibt es weder Gewinner noch Verlierer.[11]

Im nächsten Kapitel werden die im Kapitel 3 gennanten Spiele hinsichtlich der Blockchain aufgearbeitet, gedeutet und versucht einen Zusammenhang bzw. Parallele zwischen der Blockchain und der Spieltheorie herzustellen. Dabei wird zuerst ein Spiel beschrieben und anschließend die Relation zur Spieltheorie herangezogen.

4. INTERPRETATION

Angenommen es gibt einen Spieler A, der sich überlegt in das Mininggeschäft einzusteigen. Der Spieler kann helfen, jedes Projekt, das auf Blockchain basiert, zu entwickeln, indem er einen Block generiert und validiert (Mining). Für jeden Miningvorgang erhält der Spieler im Gegenzug ein Honorar in Form von Bitcoin oder Transaktionsgebühr. Für das Minen muss der Spieler Ressourcen (z.B. technische Ausrüstung und Strom) zu einem bestimmten Preis investieren. Es wird angenommen, dass jeder Spieler dieselben technischen Voraussetzungen besitzt.
Im Mining gibt es unterschiedliche Möglichkeiten für einen Miner zu minen. Damit dieses Spiel nicht in der Komplexität ausartet, gehen wir jedoch von dem einfachsten Fall aus: der Spieler hat zwei Alternativen, von denen er eine auswählen muss. Nachdem der Spieler einen Block generiert hat, muss dieser Block an die bestehende Blockkette angehängt werden (vgl. 2.2):

1. Strategie 1 (*monotone Strategie*[14]): Spieler A könnte die längste Kette verlängern, indem er eine Belohnung von 5 Einheiten erhält. Der nächste Miner (Spieler B), der einen Block mined, erhält dann auch 5 Einheiten.

2. Strategie 2: Spieler A lässt den Transaktionsbaum spalten (Fork) bzw. hängt seinen Block irgendwo an einer anderen Verzweigung der Kette und nicht an der längsten Transaktionskette. Er erhält dafür eine Belohnung von 55 Einheiten.

Abbildung 9 zeigt das Spiel in der 2x2-Bimatrix. Dabei repräsentiert Spieler B nicht nur einen anderen Mitspieler, sondern alle anderen Miner. Die Einheiten sind willkürlich gewählt.

Die Spielregeln der Miner werden in den großen öffentlichen Blockchains durch das *Proof-of-Work*-Protokoll (vgl. Kap. 2.3) festgelegt. Diese Bitcoin-Dokumentation besagt, dass Miner die monotone Strategie wählen sollten. Jedoch ignorieren einige Spieler diese Regeln, um aus ihrem Nutzen ein Maximum zu ziehen. In diesem Spiel gehen wir genau diesem Beispiel nach und gehen davon aus, dass Spieler A genau so handeln wird, um seinen Nutzen zu maximieren, unanbhängig davon, was das *Proof-of-Work*-Protokoll vorschreibt.[8]

[10] http://scienceblogs.de/zoonpolitikon/2008/04/24/spieltheorie-einfach-erklart-ii-feiglingsspiel-chicken/

[14] Eine Strategie ist monoton, wenn der längste Block um eine neuen Block ergänzt wird.

doi: 10.2313/NET-2018-11-1_03

	Spieler B (alle anderen Miner)		
		Längste Block	Fork
Spieler A	Längste Block	(5,5)	(5,55)
	Fork	(55,5	(55,55)

Abb.9: Mining-Game in 2x2-Bimatrix-Form - Eigene Darstellung

4.1 Analogie zur Spieltheorie

Welche Option ist die bessre Alternative für Spieler A? Um dies herauszufinden, muss man überlegen, welche Strategie die anderen Miner anwenden würden. Dazu betrachtet man die Spiele, die in Kapitel 3 vorgestellt wurden.

Hirschjagd (vgl. Kap. 3.2.1)

Sowohl Spieler A, als auch alle anderen Spieler (Miner) müssen wie bei der Hirschjagd unabhängig voneinander entscheiden, welche Strategie Sie wählen. Beide Player können gemeinsam dafür sorgen, dass der längste Block noch länger wird. Somit sollte es keinen Interessenskonflikt geben. Es kann aber auch sein, dass einer der Spieler an seinen maximalen Gewinn denkt und deshalb seinen Block auf dem Fork aufbaut. Nur wenn beide Miner "kooperieren" im Sinne der Hirschjagd, können Forks verhindert werden und sichern so, dass ihre Blöcke in der Konsens-Kette landen, um dann honoriert zu werden. Da aber beim Minen keine klassische Kooperation möglich ist, bleibt nur noch die Rationalität und das *Proof-of-Work* als Wegbegleiter.[8]

Taube-Falke Spiel (vgl. Kap. 3.2.2)

Nach dem Taube-Falke Spiel "gibt der Klügere nach" und jeder Spieler sollte nach Nakamoto (2008) und dem *Proof-of-Work* die monotone Strategie wählen. Dadurch hat jeder Spieler kurzfristig gesehen einen niedrigeren Gewinn erzielt, aber langfristig gesehen landet sein Block möglicherweise in der Konsens-Kette, wodurch er einen höheren Nutzen erreicht. In Anlehnung an das Taube-Falke Spiel würden seine Blöcke somit in der Blockchainkette "überleben" und einen Gewinn honorieren.

Chicken Spiel (vgl. Kap. 3.2.3)

Ähnlich zum Taube-Falke Spiel verhält sich in diesem Fall das Chicken Spiel, bei dem derjenige Miner, der die monotone Strategie wählt, möglicherweise zwar als "Chicken" degradiert wird, dessen Block allerdings nachhaltig in der Transaktionskette landet, die am längsten ist, welche seine Honorierung absichert.

In allen drei Spielformen ist zu beobachten, dass falls Spieler A die monotone Strategie wählt, ein Nash-Gleichgewicht existiert, da alle Player ebenfalls die erste Strategie wählen. Falls Spieler A in der nächsten Transaktionsrunde zu einer anderen Strategie wechseln würde, so hätte er keinen höheren Nutzen (Gewinn) davon, da sich die Rate für die Blockerstellung nicht beschleunigen würde.[8]. Falls Spieler A einen Fork wählt und Spieler B die monotone Strategie oder umgekehrt, dann erhält man ein Nullsummenspiel.

Reale Lösung

Spieler A kann nicht kontrollieren, welche Strategie die anderen Miner wählen. Wenn alle anderen Miner der Heuristik des Mining auf dem Block folgen, der am längsten ist und wenn es im Netzwerk keine Latenz gibt, so ist Forking ineffektiv und Strategie 1 ist eindeutig überlegen. Auf der anderen Seite werden sie sich vielleicht dafür entscheiden, auf dem Fork statt auf dem längsten Block zu bauen, da Option 2 eine höhre Belohnung (55 Einheiten) ausgibt. Diese Belohnung ist jedoch nur wertvoll, wenn der neue Block in der langfristigen Konsens-Kette endet. Wenn ständig neue Forks entstehen, dann hat man langfristig gesehen keinen Nutzen aus den Blöcken, da jede Kette unterbrochen wird. Des Weiteren gelten in der Realität Forks als gefährlich für Bitcoin, da sie mehrere konkurrierende Versionen der Transaktionsgeschichte erstellen und somit Zweifel darüber aufkommen lassen, wem welche Münzen gehören.[8] Tatsächlich nimmt der Spieler rational an, dass jeder Block, der aus dem Gleichgewichtspfad gelöst wird, von anderen Minern nicht akzeptiert wird und die entsprechende Belohnung bzw. dieser Zweig wertlos sein wird. Folglich hat ein Miner, der Belohnungen gesammelt hat, indem er mehrere Blöcke auf einer gegebenen Kette gelöst hat, ein Interesse daran, dass diese Kette aktiv bleibt. Insbesondere würde der Wert dieser Belohnungen sinken, wenn er in eine andere Kette wechseln würde.[4] Jeder Miner sollte sich also an das *Proof-of-Work*-Protokoll halten.

5. ZUSAMMENFASSUNG/FAZIT

Dieses Paper hat gezeigt, dass das Minen von Kryptowährungen, als ein Konsens-Spiel im Sinne der Spieltheorie betrachtet werden kann. Beim Mining-Mechanismus erhält jeder Teilnehmer (Spieler bzw. Miner) Belohnungen in Form von Bitcoin oder Transaktionsgebühren. Im Gegenzug muss Spieler A Ressourcen - wie z.B. Strom - investieren um "Rechenpuzzle" zu lösen. In Anlehnung der Spieltheorie lässt sich dieses "Spiel" - wie jedes andere auch - in einer Entscheidungsmatrix (Bimatrix) runterbrechen. Dabei werden die Regeln von einem *Proof-of-Work*-Protokoll definiert, welche allerdings von den Spielern (Minern) möglicherweise ignoriert werden, um ihren Gewinn zu maximieren. Damit jeder Miner seine Rechenleistung honoriert bekommt, muss er sich vereinfacht für eine Strategie entscheiden. Dabei gibt es zwei Optionen zur Auswahl. Entweder der Miner mined die Transaktionsblöcke und hängt sie, damit sie validiert werden, an den längsten Block der Transaktionskette (monotone Strategie) oder er hängt seinen Transaktionsblock an einer anderen Verzweigung in der Transaktionskette (Fork). Ersteres wird vom *Proof-of-Work*-Protokoll empfohlen, da Forks mehrere konkurrierende Versionen der Transaktionsgeschichte erstellen und somit Zweifel darüber aufkommen lassen, wem welche Münzen gehören. Entscheidet sich Spieler A für die monotone Strategie, so erhält er einen bestimmten Betrag. Fällt seine Wahl allerdings auf den Fork, so erhält er eine höhere Gage. Auf dem ersten Blick liegt die Wahl, für die sich der Miner entscheiden

doi: 10.2313/NET-2018-11-1_03

sollte, auf der Hand. Doch bei genauerer Betrachtung ist ersichtlich, dass es für ihn doch nicht so einfach ist, seine Strategie zu wählen. Wie bereits erwähnt, birgen Forks einige Gefahren. Des Weiteren erhält ein Miner nur dann seinen Gewinn, wenn sein Block langfristig in der Konsens-Kette landet. Dies bedeutet intuitiv: wenn alle anderen Spieler die längste Transaktionskette anvisieren und Miner A seinen Block an den Fork anhängt, er möglicherweise seinen Gewinn nicht einstreichen kann, da sein Block langfristig aus der Konsens-Kette verschwindet. Das optimale Ergebnis nach der Hirschjagd scheint in diesem Fall ein unsicheres Ergebnis zu sein. Nach dem Taube-Falke Spiel "gibt der Klügere nach" und jeder Spieler sollte nach Nakamoto (2008) und dem *Proof-of-Work* die monotone Strategie wählen. Ähnlich zum Taube-Falke Spiel verhält sich in diesem Fall das Chicken Spiel, bei dem derjenige Miner, der die monotone Strategie wählt, möglicherweise zwar als "Chicken" degradiert wird, dessen Block allerdings nachhaltig in der Transaktionskette landet, die am längsten ist, welche seine Honorierung absichert. Folglich sollte sich also jeder Miner an das *Proof-of-Work*-Protokoll halten.

6. LITERATUR

[1] Whitepaper: Nxt: https://nxtwiki.org/wiki/whitepaper:nxt.

[2] M. Abliz and T. Znati. A guided tour puzzle for denial of service prevention. In *Proceedings of the 2009 Annual Computer Security Applications Conference*, ACSAC '09, pages 279–288, Washington, DC, USA, 2009. IEEE Computer Society.

[3] A. Back. Hashcash - a denial of service counter-measure. September 2002.

[4] B. Biais, C. Bisiere, M. Bouvard, and C. Casamatta. The blockchain folk theorem. January 2018.

[5] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 154–167, New York, NY, USA, 2016. ACM.

[6] S. King and N. S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake; https://peercoin.net/assets/paper/peercoin-paper.pdf. 2012.

[7] P. D. W. Krabs. *Spieltheorie - Dynamische Behandlung von Spielen*. G. Teubner Verlag - Springer Science+Business Media, Stuttgart, 2005.

[8] J. A. Kroll, I. C. Davey, and E. W. Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. 2013.

[9] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system; http://bitcoin.org/bitcoin.pdf. 2008.

[10] W. Prinz, T. Rose, T. Osterland, C. Putschli, T. Osterland, and C. Putschli. *Blockchain*, pages 311–319. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018.

[11] C. Rieck. *Spieltheorie - Einführung für Wirtschafts- und Sozialwissenschaftler*. Dr. Th. Gabler GmbH Verlag - Bertelsmann International, Wiesbaden, 1993.

[12] R. v. Rooij. The stag hunt and the evolution of social structure. *Studia Logica*, 85(1):133–138, Feb 2007.

Performance of Message Authentication Codes for Secure Ethernet

Philipp Hagenlocher
Advisors: Dominik Scholz & Fabien Geyer
Seminar Future Internet SS2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: philipp.hagenlocher@in.tum.de

ABSTRACT
Cyclic redundancy checks within Ethernet do not provide data integrity, which can be solved by using message authentication codes (MACs) within Ethernet frames. This paper gives theoretical background on hash functions and hash-based MACs, different schemes are compared for the usage in Ethernet frames by multiple attributes such as performance, output lengths and security. The functions Poly1305, Skein and BLAKE2 are tested for throughput against implementations of a SHA-2 HMAC and SipHash and the results are discussed. Numerical measurements and results show that Poly1305 is a suitable candidate for the usage as a per-packet MAC.

Keywords
Message Authentication Codes, Hash functions, Performance evaluation, Secure Ethernet

1. INTRODUCTION
In daily communication within LANs, MANs and WANs, computers use the Ethernet protocol and furthermore send so called Ethernet frames over the network in order to communicate. These frames possess a checksum calculated with a cyclic redundancy check (CRC), which is a code to detect errors, in the form of flipped bits, within the frame [27]. This ensures that transmission errors are detected.

Using CRC has the advantage of needing only a small amount of space (32 bits) and having a code that can be computed in short time. The problem with CRC is that it does not indicate data integrity of the frame if an attacker tries to maliciously alter it. The attacker can easily recompute the CRC since there is no secret protecting the checksum [31].

In order to combat the problem of data integrity *message authentication codes* (MAC) are often used in cryptographic protocols. MACs based on cryptographic hash functions can use an additional secret key in order to combine the integrity of the frame with a shared secret, thus blocking a potential attacker from altering the frame *and* computing a correct checksum. A popular usage of hash functions in MACs are *hash-based message authentication codes* defined in Section 2.3, although MACs do not have to be constructed with or from hash functions. Another application of hash functions are *TCP SYN cookies*, which are used in order to combat *TCP SYN flooding* attacks [28], where an attacker intentionally doesn't complete the 3-way-handshake of TCP connection establishment in order to waste resources on the server until a denial of service is reached. To combat this attack scheme, a hash function is used to compute a hash combining the IP address of the client, port numbers and a timestamp. This hash is then used as the sequence number for the packet. This way the client is identified by this hash, thus the server can free old resources if a new SYN is sent by the client. Another important usage for hash functions are lookup tables, where hash functions are used to group values into segments that can be retrieved in faster time than a simple lookup in a set of elements.

1.1 Outline
This paper is structured as follows: At first it will explain the theory behind hash functions and hash-based MACs in Section 2. Section 3 will give an overview over possible candidates for the usage as a MAC scheme within Ethernet and Section 4 will explain the testing environment and discuss test results. The conclusion to the test results and further discussion is given in Section 5.

2. HASH FUNCTIONS
A hash function h is defined as a *one-way function*

$$h : \Sigma^* \rightarrow \Sigma^n$$

where $\Sigma = \{0, 1\}$, Σ^n denotes all bitstrings of length n and Σ^* denotes all bitstrings of arbitrary length [6, p. 177]. In order for this function to be cryptographically secure the function needs to have weak and strong collision resistance. Weak collision resistance is defined as the absence of an efficient algorithm that, for a given m, can find a m' such that $h(m) = h(m')$. Strong collision resistance is defined as the absence of an efficient algorithm that can find any m and m' such that $h(m) = h(m')$. These properties are important for the use of these hash functions in a MAC scheme since we don't want an attacker to be able to guess what possible alterations can be made to the content of the message or, even worse, find out what the shared secret key is. Hash functions that possess these properties are known as *cryptographic hash functions*.

2.1 Security of hash functions
When talking about cryptographic strength of a hash function, what is really talked about is the computational complexity of finding a collision. Let h be a perfect hash function with perfect weak and strong collision resistance and

doi: 10.2313/NET-2018-11-1_04

an output length of n. In order to find a collision, an attacker would need to randomly try out different m and m' and check if $h(m) = h(m')$. Since hash functions have a fixed output length the number of possible hash values is limited. Due to the birthday paradox [6, p. 175-180] the complexity of randomly trying out different input values is $2^{\frac{n}{2}}$. Generally, a hash function is considered unsafe once this complexity can be reduced drastically and thus making it practically possible to brute-force collisions or even compute them outright.

This example is known as a *birthday attack*. This attack cannot be improved if the hash function has perfect strong collision resistance. Another attack is the *pre-image attack*, which describes finding an appropriate x to a given y such that $h(x) = y$. Brute-forcing this x has the complexity of 2^n where n is the output length of h. This attack cannot be improved if the hash function has perfect weak or strong collision resistance.

2.2 Construction

Constructing hash functions has many strategies. One of the most popular schemes is the *Merkle-Damgård construction* described by Merkle [24, p. 13-15]. This construction splits the message into blocks and applies a function $f(k, m)$ to these blocks:

$$f(...f(f(s, b_1), b_2)..., ...b_n)$$

where s denotes a fixed starting value and b_n denotes the nth message block. There can be a finalization function applied on the result of this computation. f could be an encryption function that takes a key k and a message m. Popular hash function families like MD, SHA-1 and SHA-2 all use this kind of construction.

Of course there are many ways of constructing hash functions such as using permutation networks, S-Boxes and other parts of block ciphers in order to build a one-way function. JH [15, p. 26], BLAKE [18] and Keccak [17] are examples of that.

The aforementioned *substitution boxes (S-Boxes)* are used by many hash functions in their construction. They describe a mapping from bitstrings of fixed size to other bitstrings of another fixed size. For example, the S-Boxes within the block cipher DES can be represented as a table of values used for substitution, where the choice of columns and rows is done by looking at the outer and inner bits of the bitstring that needs to be substituted [6, p. 76-77].

2.3 Hash-Based Message Authentication Code

A *hash-based message authentication code* (HMAC) is a MAC combining hash functions with a secret key in order to achieve integrity between parties that share said key. RFC 2104 [22] defines the way to compute an HMAC like so: Let H denote a cryptographic hash function with output length of n and internal block size of B, K a secret key, MSG the message to compute an HMAC for, $\|$ the concatenation of byte-strings, $\bigoplus$ the exclusive-or (XOR) operation, *ipad* and *opad* byte-strings consisting of 0x36 and 0x5C repeated B times respectively. Then the HMAC is defined as:

$$H((K \bigoplus opad) \parallel H((K \bigoplus ipad) \parallel MSG))$$

It is easy to see that it is absolutely possible to switch out H with any cryptographic hash function and that a *cryptographic* hash function is needed in order for the HMAC to be secure and that the computational speed of the algorithm is heavily influenced by the speed of the underlying hash function.

3. SURVEY

This section describes and compares suitable candidates for our use-case. Surveying the finalists of the hash function competition by the US National Institute of Standards and Technology (NIST) is a good choice to find hash functions for an HMAC construction since these functions have been preselected due to their performance and security. As well as the finalists of this competition more MAC schemes will be taken into consideration, that have recently found usage in real world applications.

Since performance is an important measure to go by it is advantageous for a candidate to be usable as a standalone MAC scheme. Otherwise an HMAC has to be created out of the function in question, which uses two distinct calls to the function thus slowing down the resulting MAC scheme.

3.1 Use-case

The use case is a per packet message authentication code. Obviously this MAC needs to be secure, but has to have a high performance in order to not slow down traffic too much, since bandwidths of 10 Gbit/s can be achieved and should not be bottlenecked by the CPU and hash computation. Initial tests with an HMAC using SHA-256 and SHA-512 showed bad performance, which can be seen in figure 3 and 4. This performance was increased by switching to an 32-bit variant of SipHash (explained in Section 3.3.7).

3.2 Unsafe functions

Several functions will not be taken into consideration since attacks against these functions have been demonstrated or are theoretically possible. These functions are

- MD4 [36]
- MD5 [32]
- SHA1 [30]
- GOST [11]
- HAVAL-128 [34]
- RIPEMD [35]

3.3 Candidates

In this section possible candidates for the described use-case are described and important features are highlighted.

3.3.1 SHA-2

The SHA-2 family uses the aforementioned *Merkle-Damgård construction* described in section 2.2. The performance was retested in the hash function competition which resulted in ∼11-14 cycles per byte [15, p. 43-44] for SHA-512, the variant of SHA-2 with an output length of 512 bits that generally performs the best. The SHA-2 family was developed by the NSA and standardized by NIST [16]. The family is well known, widely used and extensively studied, thus providing a popular choice for usage as an HMAC.

doi: 10.2313/NET-2018-11-1_04

3.3.2 Keccak

Keccak uses a sponge function combined with permutations and 24 rounds. It is expected to perform well when implemented in hardware [15, p. 6], which holds true when testing the algorithm in an FPGA implementation [12]. One attack found against this function had the complexity of $2^{511.5}$ on a version of Keccak with 8 rounds [15, p. 30]. It won the competition and is now known as the SHA-3 standard [17].

3.3.3 BLAKE

BLAKE is based on the *ChaCha* stream cipher and uses 14 rounds for 224 and 256 bits and 16 rounds for 358 and 512 bits of output [15, p. 17]. It was analyzed heavily during the competition. On performance it was noted that BLAKE performs well in software [15, p. 6]. A collision attack on a version of BLAKE-512 with 2.5 rounds has the complexity of 2^{224} [15, p. 22].

3.3.4 JH

JH uses permutations in combination with XOR operations and S-Boxes with 42 rounds for all output lenghts [15, p. 26]. JH is considered slower than BLAKE, Skein and Keccak [15, p. 6]. An attack was found with a time complexity of 2^{304} [15, p. 28].

3.3.5 Grstl

Grøstl is a combination of a *Merkle-Damgård construction* combined with parts of the block cipher *AES* [15, p. 23]. It uses 10 rounds for an output length of up to 256 bits and 14 rounds for an output length of up to 512 bits. It is considered slower than BLAKE, Skein and Keccak [15, p. 6] and has drastically different performance for 224/256 and 384/512 bits output length [13, p. 23]. The only collision attack on Grøstl with an output length of 512 bits had 3 rounds and a complexity of 2^{192} [15, p. 26].

3.3.6 Skein

Skein uses the block cipher *Threefish* which uses 72 rounds of a substitution-permutation network. Similar to BLAKE, Skein is expected to perform well when implemented in software[15, p. 6]. A lot of the cryptographic analysis went into the block cipher. Almost all of the found attacks on versions of Skein with reduced rounds are impractical. One of the collision attacks on Skein with the output length of 512 bits and 14 rounds has the complexity of $2^{254.5}$[15, p. 33]. Another important feature of Skein is a special MAC mode called Skein-MAC[25, p. 7-8].

3.3.7 SipHash

While not being a hash function SipHash is a keyed *pseudo random function (PRF)*, thus not necessarily having collision resistance but still making it infeasible to compute the hash value if the key is not known. This function was specifically designed to be used as a MAC for short input values and was inspired by BLAKE, Skein and JH [20]. Its performance is a key feature, since it was designed to be resistant against denial of service attacks in schemes such as TCP SYN cookies where the hash function can take up so much computational time that the server is unable to handle other workload. A cryptographic analysis of SipHash concluded that finding a collision has the probability of $2^{-236.3}$ [9].

3.3.8 Poly1305-AES

Poly1305-AES is another MAC scheme that is not a hash function. The author claims high performance, with linearly growing cycles per bytes, and guaranteed security if AES is secure [7]. Another important feature is the output size of 128 bits, which is quite small. It has since been standardized in RFC 7539 [26], has found usage by Google and was incorporated into TLS1.3. Also important to note is that AES can be replaced by any other cipher such as ChaCha20. There has been research on the security of this specific Poly1305 variant [14].

3.3.9 BLAKE2

The successor to BLAKE has been standardized in RFC 7693 [29]. The internals are comparable to BLAKE. The authors claim better performance than MD5, SHA2 and SHA3 [19]. It features special versions for 64- and 32-bit platforms and has a special prefix-MAC mode. It's security has been extensively studied, since most of the theoretical work done on BLAKE is also applicable for its successor [19, p. 4].

3.4 Comparison

In this section the aforementioned hash functions and MAC schemes are compared with the already tested algorithms and their suitability for the use case is rated. Performance, output length and security are the key factors that will be compared.

	Performance (approx. cycles/byte)	Output length (bits)	Stand-alone MAC
Keccak	6.7-19	224-512	No
BLAKE	8.4	224-512	No
JH	15-20	224-512	No
Grøstl	13-92 / 18-126	224-512	No
Skein	6.1	Any	Yes
Poly1305	4-15	128	Yes
BLAKE2	3-12	8-2048	Yes
SHA-2	11-14	224-512	No
SipHash	4-10	64	Yes

Table 1: Summary of candidates

3.4.1 Performance

There have been a lot of performance tests done for the NIST hash function competition finalists and their performances have been compared to SHA-256 and SHA-512. In order to have a comprehensive comparison, the functions were tested in three different processors. JH and Grøstl generally perform worse than functions of the SHA-2 family and the other finalists [15, p. 43-44]. The performance of Keccak depends greatly on the output length. For sizes of 224 or 256 bits it is described as *"reasonably fast"* [15, p. 46], but for the output size of 512 bits it is rather slow. Skein and BLAKE are generally faster than SHA-2, while BLAKE seems to be the fastest function. Skein is the only function that has the same performance for all output lengths [15, p. 43-46]. The performance of SipHash is comparable with SHA-512 (∼10

doi: 10.2313/NET-2018-11-1_04

cycles/byte) [15, p. 43-44] and for a message length of 64 bytes it beats out BLAKE with 4 cycles/byte [20, p. 11]. A comparable performance is achieved by BLAKE2 [19, p. 14]. Poly1305 is claimed to have $3.1n + 780$ cycles for a n byte message. For a 64 byte message this would result in ∼15 cycles/byte and for a 1024 byte message this would result in ∼4 cycles/byte. So its performance can be compared to SipHash. The main difference is mainly that Poly1305 profits from long message lengths due to the constant overhead.

3.4.2 *Output lengths*

All NIST hash function competition finalists have output lengths of 224, 256, 384 and 512 bits. Skein is very flexible by having an arbitrary output length and three different internal block sizes (256, 512 and 1024 bits) [25, p. 1]. BLAKE2 features digest sizes from 1 to 256 bytes [19, p. 6]. SipHash computes a MAC of fixed length (64 bit) [20, p. 6]. Poly1305 doubles this length for its output of 128 bits [7, p. 1].

3.4.3 *Security*

None of the functions taken into consideration are considered broken and all attacks known against them are impractical. Still, it is important to note that some functions have not been studied as rigorously as others. BLAKE, Skein and Grøstl have had extensive analysis done on them in the hash function competition, while Keccak and JH had less work done on them [15, p. 33]. Also, in real world applications Skein, JH and Grøstl are not getting as much usage as the other functions. BLAKE2 is implemented in many cryptographic libraries such as libsodium [2] and has found usage in the password hashing scheme Argon2 [8]. Poly1305 has been adopted by Google as an RC4 replacement and Poly1305 in combination with the ChaCha20 block cipher was incorporated into OpenSSH [5]. It is important to note that this implementation of Poly1305 is susceptible to side channel attacks [21]. SipHash is the hash function for the hash table implementations within Python and Rust and is the *"shorthash"* function in libsodium [2].

3.4.4 *Final Choice*

The most suitable MACs should be generally better suited than SHA-2 and SipHash. Looking at Table 1 one can see that JH and Grøstl are generally to slow to be suitable choices. Keccak and BLAKE are fast, but suffer from poor flexibility in output lengths. Since all the algorithms can be considered secure the three final candidates are Skein (as Skein-MAC), BLAKE2 (with its MAC-mode) and Poly1305. Not only do Skein and BLAKE2 have good performance, but they also have designated MAC schemes, that are designed to be faster than traditional HMACs. It is important to note that hardware performance did not factor into the selection process, because it cannot be tested in the used testenvironment, which is why Keccak will not be tested even though showing outstanding performance.

4. EVALUATION

This section describes the testing environment and testing procedure. It then discusses the results from the testing.

```c
#include "blake2.h"
#include "skein.h"
#include "poly1305-donna.h"

static inline uint32_t stream2int(const uint8_t
↪  *stream) {
    return (((uint32_t) stream[0]) << 24 |
        ((uint32_t) stream[1]) << 16 |
        ((uint32_t) stream[2]) <<  8 |
        ((uint32_t) stream[3]) <<  0);
}

uint32_t calculate_blake2b(const blake2b_state
↪  *ctx, const void *msg, uint16_t length) {
        uint8_t mac[4];
        blake2b_update(ctx, msg, length);
        blake2b_final(ctx, mac, 4);
        return stream2int(mac);
}

uint32_t calculate_blake2s(const blake2s_state
↪  *ctx, const void *msg, uint16_t length) {
        uint8_t mac[4];
        blake2s_update(ctx, msg, length);
        blake2s_final(ctx, mac, 4);
        return stream2int(mac);
}

uint32_t calculate_skein(const Skein_256_Ctxt_t
↪  *ctx, const void *msg, uint16_t length) {
        uint8_t mac[4];
        Skein_256_Update(ctx, msg, length);
        Skein_256_Final(ctx, mac);
        return stream2int(mac);
}

uint32_t calculate_poly1305(uint8_t *key, const
↪  void *msg, uint16_t length) {
        /* key must be 32 bytes and UNIQUE */
        uint8_t mac[16];
        poly1305_auth(mac, msg, length, key);
        /* only using first 32 bit */
        return stream2int(mac);
}
```

Figure 1: Implementation of the tested MACs

4.1 Setup

Skein, BLAKE2 and Poly1305 are tested in an artificial environment with a P4 switch. A load generator generates a stream of Ethernet frames with increasing bandwidth. Both the switch and load generator are equipped with a Xeon E3-1230, 15.6 GB of memory and run the Linux 4.9.0-5 kernel. For P4 compilation the environment uses *t4p4s*, a fork of *p4c*[23]. A configuration for the compiler defines which algorithm is used for which test case, even though the internal controller and remaining configuration stays the same in order to keep testing neutral. The load generator uses MoonGen [10] in order to create packets at high rates. The switch is tasked with computing the MAC for each frame and will respond with the finished packet. In order to keep the workload realistic frames are sent in lengths from 64 to 1500 bytes with rates between 50 and 10000 Mbit/s.

doi: 10.2313/NET-2018-11-1_04

TX	Poly1305	Skein	BLAKE2s	BLAKE2b
50	48.19	48.19	48.19	48.19
100	99.98	99.77	99.80	99.80
150	144.29	144.33	144.33	144.63
200	200.04	199.93	199.54	199.57
250	240.49	240.43	240.00	240.46
500	498.84	498.81	500.04	498.81
1000	961.49	529.52	961.64	963.24
2000	1977.02	561.63	1588.61	1849.23
3000	1774.09	529.58	1410.94	1662.78
4000	1973.52	561.35	1572.47	1849.95
5000	1768.91	529.56	1420.38	1657.34
7500	1974.78	562.86	1585.03	1850.57
10000	1783.94	529.47	1420.20	1666.51

(a) Bandwidths for 64 bytes per packet (in Mbit/s)

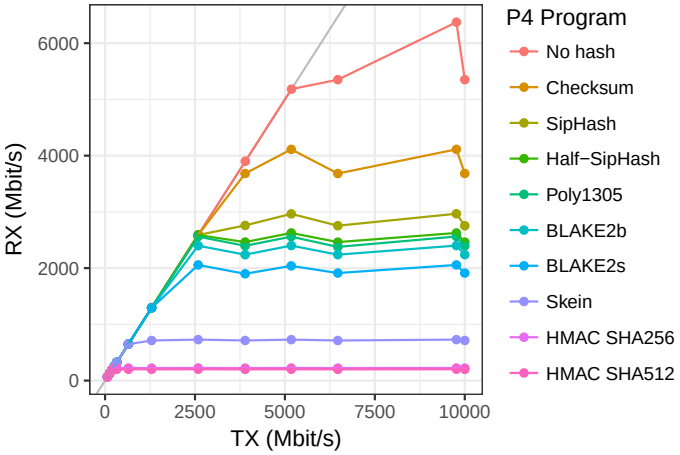
TX	Poly1305	Skein	BLAKE2s	BLAKE2b
50	50.03	50.03	50.03	49.94
100	96.55	96.55	96.36	96.39
150	150.04	149.77	150.04	150.04
200	192.90	192.91	192.47	192.56
250	249.53	249.98	249.03	250.04
500	480.97	481.00	480.98	480.86
1000	999.62	997.67	999.63	997.86
2000	1923.14	1282.15	1926.51	1923.32
3000	3001.89	1333.83	2394.66	3001.83
4000	3852.96	1282.29	2298.26	3425.08
5000	4986.97	1334.10	2399.72	3578.64
7500	6025.09	1284.78	2312.45	3417.14
10000	6428.82	1334.35	2394.06	3579.34

(b) Bandwidths for 512 bytes per packet (in Mbit/s)

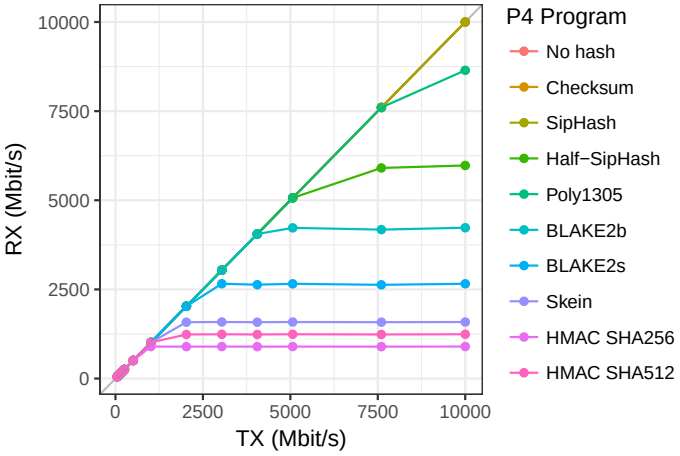
TX	Poly1305	Skein	BLAKE2s	BLAKE2b
50	50.03	50.03	50.03	50.03
100	96.55	96.57	96.59	96.57
150	150.04	149.92	150.04	150.04
200	192.74	192.85	192.77	192.98
250	250.04	249.64	250.04	249.62
500	481.14	481.23	482.12	481.28
1000	997.96	997.59	999.73	1000.07
2000	1923.15	1499.35	1922.85	1923.34
3000	2999.31	1560.45	2622.18	2993.16
4000	3845.81	1502.14	2502.63	3846.05
5000	4996.43	1562.82	2615.59	4160.39
7500	7224.33	1500.01	2498.69	3963.35
10000	8527.97	1560.32	2616.91	4164.24

(c) Bandwidths for 1500 bytes per packet (in Mbit/s)

Figure 2: Bandwidth measurements



(a) Bandwidths for 64 bytes per packet



(b) Bandwidths for 1500 bytes per packet

Figure 3: Throughput comparison

4.2 Implementation

In order to keep testing neutral, there were no optimized libraries used since the test concerns the raw performance of the algorithms, not their optimization. For Skein the reference implementation [4] is used as well as for BLAKE2 [1]. For the first the internal size is set to 256 bit and for the later there are two testing candidates namely *BLAKE2b* and *BLAKE2s* the implementations meant for 64 bit systems and 32 bit systems respectively. The Poly1305 implementation [3] is a portable implementation that is kept close to the reference implementation. For every MAC the same key is used, which should be avoided in real world usage of Poly1305, since it needs a new key for every new message. This test does not factor in the overhead that computing new keys would create even though for every MAC the needed initialization function is called every time. Figure 1 shows the relevant source code for the tested functions. The context for BLAKE2 and Skein are initialized beforehand.

doi: 10.2313/NET-2018-11-1_04

4.3 Results

The test results for packet sizes of 64 and 1500 bytes are shown in Figure 3. Figure 3a shows similar performance for Poly1305 and BLAKE2, while Skein is performing worse. For increasing size, Poly1305s performance is rapidly increasing which could be due to the performance of $3.1n+780$ cycles per n bytes, trivially converging to 3.1 cycles per byte for steadily increasing message sizes. The gap between BLAKE2b and BLAKE2s is also increasing for larger packet sizes, while Skein is hitting a bottleneck for a bandwidth of around 1500 Mbit/s. BLAKE2b has a similar bottleneck at 4000 Mbit/s. For small packet sizes, Poly1305 is also hitting a bottleneck, even though it occurs for higher bandwidths as the other candidates as can be seen in the tables of Figure 2. Comparing the three candidates to HMACs built from SHA-2 and SipHash reveals that all of them outperform the HMACs, but non of them outperform SipHash. The most important finding is that Poly1305 increases in performance for packet sizes bigger than 1000 bytes while all other performances decrease as can be seen in Figure 4. This could indicate that Poly1305 will eventually outperform SipHash at a certain packet size, but further testing would need to be done in order to prove this conjecture.

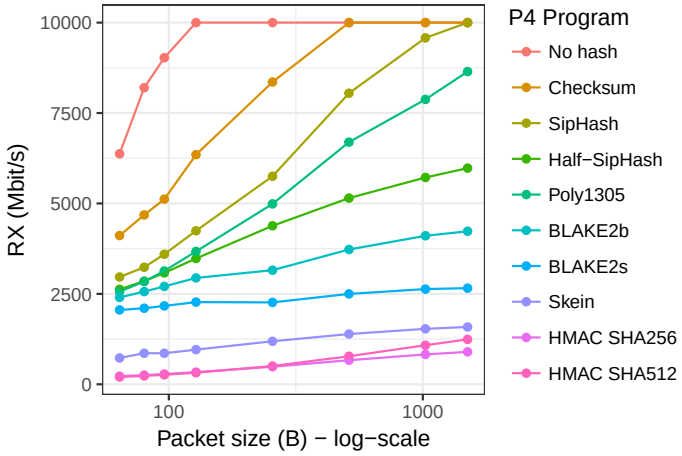


Figure 4: Throughput comparison by packet size

5. CONCLUSION

The paper has shown the motivation behind the usage of hash functions in Ethernet frames and has explained the theory behind hash functions. Multiple functions and MAC schemes have been compared by their qualities for this use case and the three candidates (Skein, BLAKE2 and Poly1305) have been compared in a synthetic test environment.

The test indicates that Poly1305 is a suitable candidate when it comes to throughput due to its great performance for bigger packet lengths. While throughput is important, Poly1305's implementation faces problems that cannot be ignored. It needs a *new* key for every message that has to be 32 bits large. This problem is fixed in an alternative proposal called DPoly1305, which does not require a new key for every message [33], but has not been standardized or thoroughly analysed yet. Another problem that the implementation faces is a fixed output length of 128 bit. The hardware performance of Poly1305 was not tested and for implementation in hardware, an algorithm like Keccak could be far more suitable, but further testing would need to be done before a conclusive statement can be made.

6. REFERENCES

[1] Blake2 github repository. `https://github.com/BLAKE2/BLAKE2`. Accessed: 2018-04-16.

[2] libsodium github repository. `https://github.com/jedisct1/libsodium`. Accessed: 2018-04-16.

[3] Poly1305-donna github repository. `https://github.com/floodyberry/poly1305-donna`. Accessed: 2018-04-16.

[4] Skein github repository. `https://github.com/meteficha/skein/tree/master/c_impl/reference`. Accessed: 2018-04-16.

[5] Super user's bsd cross reference: Protocol.chacha20poly1305. `http://bxr.su/OpenBSD/usr.bin/ssh/PROTOCOL.chacha20poly1305`. Accessed: 2018-04-16.

[6] Albrecht Beutelspacher, Heike B. Neumann, Thomas Schwarzpaul. *Kryptografie in Theorie und Praxis*. Vieweg+Teubner Verlag, 2010.

[7] D. J. Bernstein. The poly1305-aes message-authentication code, 2005. `http://cr.yp.to/mac/poly1305-20050329.pdf`.

[8] A. Biryukov, D. Dinu, and D. Khovratovich. Argon2: New generation of memory-hard functions for password hashing and other applications. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 292–302, 2016.

[9] C. Dobraunig, F. Mendel, and M. Schläffer. Differential cryptanalysis of siphash. Cryptology ePrint Archive, Report 2014/722, 2014. `https://eprint.iacr.org/2014/722`.

[10] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle. MoonGen: A Scriptable High-Speed Packet Generator. In *Internet Measurement Conference 2015 (IMC'15)*, Tokyo, Japan, Oct. 2015.

[11] Florian Mendel, Norbert Pramstalle1, ,Marcin Kontak, Janusz Szmidt. Cryptanalysis of the gost hash function, 2008. `https://online.tugraz.at/tug_online/voe_main2.getvolltext`.

[12] K. Gaj, E. Homsirikamol, M. Rogawski, R. Shahid, and M. U. Sharif. Comprehensive evaluation of high-speed and medium-speed implementations of five sha-3 finalists using xilinx and altera fpgas. *IACR Cryptology EPrint Archive*, 2012:368, 2012.

[13] P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and T. S. S. Grøstl - a sha-3 candidate. In H. Handschuh, S. Lucks, B. Preneel, and P. Rogaway, editors, *Symmetric Cryptography*, number 09031 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[14] K. Imamura, K. Minematsu, and T. Iwata. Integrity analysis of authenticated encryption based on stream ciphers. *International Journal of Information Security*,

Jun 2017.

[15] Information Technology Laboratory National Institute of Standards and Technology. Nistir 7896 (third-round report of the sha-3 cryptographic hash algorithm competition), 2012. https://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7896.pdf.

[16] Information Technology Laboratory National Institute of Standards and Technology. Fips pub 180-4 (secure hash standard (shs)), 2015. http://dx.doi.org/10.6028/NIST.FIPS.180-4.

[17] Information Technology Laboratory National Institute of Standards and Technology. Fips pub 202 (sha-3 standard: Permutation-based hash and extendable-output functions), 2015. http://dx.doi.org/10.6028/NIST.FIPS.202.

[18] W. M.-R. C.-W. P. Jean-Philippe Aumasson, Luca Henzen. Sha-3 proposal blake, 2010. https://131002.net/blake/blake.pdf.

[19] Z. W.-O. C. W. Jean-Philippe Aumasson, Samuel Neves. Blake2: simpler, smaller, fast as md5, 2013. https://blake2.net/blake2.pdf.

[20] Jean-Philippe Aumasson, Daniel J. Bernstein. Siphash: a fast short-input prf, 2010. https://131002.net/siphash/siphash.pdf.

[21] B. Jungk and S. Bhasin. Don't fall into a trap: Physical side-channel analysis of chacha20-poly1305. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 1110–1115, March 2017.

[22] D. H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, Feb. 1997.

[23] S. Laki, D. Horpácsi, P. Vörös, R. Kitlei, D. Leskó, and M. Tejfel. High speed packet forwarding compiled from protocol independent data plane specifications. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 629–630, New York, NY, USA, 2016. ACM.

[24] R. C. Merkle, R. C. erkle, R. C. Yerkle, A. Students, S. Pohlig, R. Kahn, and D. Andleman. Secrecy, authentication, and public key systems. Technical report, 1979.

[25] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, Jesse Walker. The skein hash function family, 2010. http://www.skein-hash.info/sites/default/files/skein1.3.pdf.

[26] Y. Nir and A. Langley. ChaCha20 and Poly1305 for IETF Protocols. RFC 7539, May 2015.

[27] Philip Koopman. 32-bit cyclic redundancy codes for internet applications, 2002. http://users.ece.cmu.edu/~koopman/networks/dsn02/dsn02_koopman.pdf.

[28] L. Ricciulli, P. Lincoln, and P. Kakkar. Tcp syn flooding defense. CNDS, 1999.

[29] M.-J. O. Saarinen and J.-P. Aumasson. The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC). RFC 7693, Nov. 2015.

[30] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov. The first collision for full sha-1. In *Annual International Cryptology Conference*, pages 570–596. Springer, 2017.

[31] M. Stigge, H. Plötz, W. Müller, and J.-P. Redlich. Reversing crc–theory and practice. 2006.

[32] Tao Xie and Fanbao Liu and Dengguo Feng. Fast collision attack on md5. Cryptology ePrint Archive, Report 2013/170, 2013. https://eprint.iacr.org/2013/170.

[33] D. Wang, D. Lin, and W. Wu. A variant of poly1305 mac and its security proof. In Y. Hao, J. Liu, Y.-P. Wang, Y.-m. Cheung, H. Yin, L. Jiao, J. Ma, and Y.-C. Jiao, editors, *Computational Intelligence and Security*, pages 375–380, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[34] X. Wang, D. Feng, X. Lai, and H. Yu. Collisions for hash functions md4, md5, haval-128 and ripemd. Cryptology ePrint Archive, Report 2004/199, 2004. https://eprint.iacr.org/2004/199.

[35] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu. Cryptanalysis of the hash functions md4 and ripemd. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pages 1–18, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[36] Yu Sasaki, Lei Wang, Kazuo Ohta and Noboru Kunihiro. New message difference for md4, 2007. https://www.iacr.org/archive/fse2007/45930331/45930331.pdf.

Fog Computing for Smart Environments

Ulrich Huber, B.Sc.
Advisor: Jan Seeger, M.Sc.
Seminar Future Internet SS2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: ulrich.huber@tum.de

ABSTRACT

Fog Computing or edge computing is a new approach to perform services and applications at the edge of the network, which keeps latency and bandwidth usage to a minimum. Therefore fog computing is a way to enable fast responding applications in smart buildings. This paper evaluates the suitability of multiple frameworks for fog computing and from different usage areas, for use in building automation tasks. Especially compatibility with low-powered hardware, strategies for failures of components or infrastructure as well as soft-realtime capability are necessities to such frameworks. We use these three formulated requirements, to evaluate the basic architecture of six frameworks, which are from different areas of research and usage. Throughout the paper we find that there is currently no suitable candidate for deployment in smart environments. But the results of the evaluation can be used to develop a suitable framework.

Keywords

IoT, smart building, fog computing, edge computing, frameworks, EHOPES, FRODO, Gabriel, MACaaS, NetFATE, ParaDrop

1. INTRODUCTION

With the introduction of the Internet of Things (IoT) we find ourselves confronted with not only new possibilities, but also new problems to overcome. As the number of sensors in our environment rises and therefore the amount of data that needs to be transmitted, analyzed and stored, the currently used paradigm of cloud computing is unsuited for the development of the full potential of the IoT [13]. Especially if we do not only want to gain information by analyzing the data, but want to act on it in a timely manner, the round-trip-time is too high for many scenarios like health care or smart home/city. For example when sensor data represents user-interactions, which need to be analyzed to influence actuators near the vicinity of the user, we have very strict delay requirements [5]. Additionally conglomerating all sensor data to centralized data centers is infeasible, since the network bandwidth would saturate and not be scalable. Finally moving processing of gathered data to the cloud raises security concerns, since there is no direct influence on how privacy is handled by the provider of the cloud [3].
To circumvent these restrictions, the new approach of fog computing was introduced [13]. Thus fog computing enables low bandwidth and latency by using devices (fog nodes) in the vicinity of the user to store and analyze data. These fog nodes can be any type of device in the network, including devices that already have their own purposes, such as routers. To enable a multitude of services on devices with very different hardware specifications, fog computing relies very heavily on virtualization.

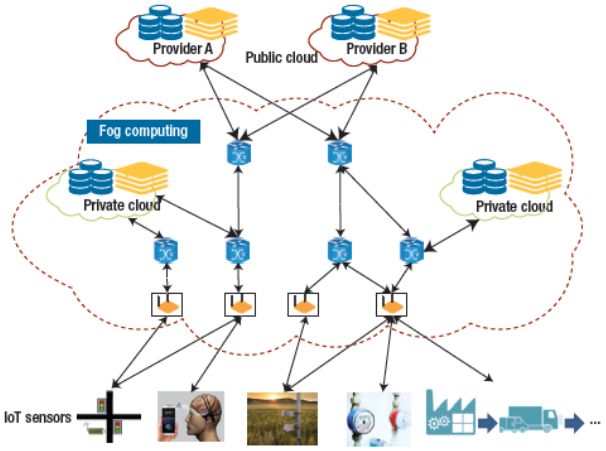


Figure 1: Distributed data processing in a fog-computing environment. Collected sensor data is dynamically procesed by nearby fog devices, including gateways and private clouds. Taken from [3].

By using fog nodes, which physically reside near the smart objects as can be seen in Figure 1, it is possible to keep latency to a minimum and perform most analysis of the sensor data without reaching out to the public cloud and thereby saving bandwidth and keeping data secure.
The previous advantages of fog computing are very important for the IoT, but most applications still require globalization as provided by the cloud. Therefore fog computing is no replacement but an expansion to the cloud, where services can be transparently moved from a centralized cloud to the very edge of the network. With the localization of services some difficulties are introduced, like the need to move the services on a fog node along with the user while he moves and the need for interoperability between different nodes, which has to be solved by frameworks using fog computing [1].
In the context of smart buildings most applications and services are latency-sensitive, for example light switches and blinds. Fog computing can be used to act upon user interac-

doi: 10.2313/NET-2018-11-1_05

tions or events within a timely manner. Since fog computing uses a distributed approach, even with increasing amounts of sensors and therefore generated data, existing infrastructure can be used to enable smart buildings, since bandwidth requirements remain stable.

In this paper we evaluate a variety of frameworks, which rely on fog computing, for their suitability for use in building automation tasks. Reliability is an important requirement for this task, since especially in cases of emergency the system needs to function to a certain degree. The easiest way to accomplish this is by building the framework with no hard dependencies on connectivity or distant resources, like the cloud. To keep deployment in buildings inexpensive and make use of "leftover" computation power of smart objects, such frameworks should have low hardware requirements. Finally many tasks, like switching lights on and off or streaming content, have soft-realtime character, which frameworks need to support. Thus strategies to instruct powerful enough devices to handle such tasks, or handle them in between multiple devices in cooperation are required. The evaluation will cover these three requirements, namely reliability, hardware requirements and soft-realtime capability.

In Section 2 we give some background on fog computing and describe the most common components of frameworks incorporating this approach. The frameworks to be evaluated, are introduced in Section 3. In Section 4 we formulate multiple evaluation criteria based on the basic requirements to frameworks using fog computation within smart buildings, and evaluate the previously introduced frameworks with them. Finally a conclusion is drawn, which approaches are especially noteworthy and should be included in frameworks for smart buildings.

2. FOG COMPUTING

In scientific literature the terms of fog computing and edge computing are interchanged quite often, ignoring the differences in meaning of the two terms. While edge computing targets the locality where services are instantiated, the edge of the network, fog computing implies the distribution of computing power, communication and data storage to devices close to the users requiring them [11]. Therefore the terms of edge computing and fog computing have a certain overlap, but while hardware for edge computing is bound to a specified service, fog computing is about creating the possibility to run any service at the edge of the network.
Thus hardware for fog computing has to provide a common interface that is suitable for many different tasks, which is accomplished by virtualization of the hardware. This has the added advantage, that there are no restrictions to the devices, which are called fog nodes, that are used in fog computing. That means, that fog computing can make use of any device from a smart light-bulb or a router with "leftover" computing power, to whole clusters of computers or even data centers, as long as a device can run the agent software needed for virtualization and communication between the fog nodes.
Not specified in the definition of fog computing but used by many frameworks that embody fog computing, is the so-called fog server. While fog nodes are the workers of such a framework, the fog server is the boss and janitor. It takes care of organizational tasks, like instantiation, deployment, migration and termination of tasks, which are then executed on fog nodes, chosen by the fog server. Additionally the fog server handles maintenance tasks, which include logging of events and redistributing work, when nodes or parts of the infrastructure fail.

3. FRAMEWORKS

In the recent years multiple groups of scientists incorporated fog computing into their projects, thus creating a huge variety of frameworks with very diverse approaches to the topic. Most frameworks are tailored to the requirements of their projects and are therefore not necessarily suited to other uses. Still there are not yet enough research projects into fog computing for smart buildings, to enable an evaluation solely using such frameworks. Therefore the frameworks, to be introduced, are not necessarily developed for the purpose of enabling smart buildings. They were selected for their unique approaches to one or more of the requirements listed in Section 1, not their compatibility with the actual automation of building tasks.

3.1 EHOPES

EHOPES is designed for the topic of smart living, which is separated into multiple subtopics for modularization. The subtopics, further on called applications, are Smart Energy, Smart Healthcare, Smart Office, Smart Protection, Smart Entertainment and Smart Surroundings. Each application brings up its own sub-network of smart objects, which provide the necessary data and actions to perform its work. EHOPES has the goal of minimizing transmission latency that smart living frameworks utilizing cloud computing show and is developed by Li et al.. The framework utilizes a star topology for its framework, where a fog server is the central node and fog nodes are the outer nodes. Additionally fog nodes can be connected to other fog nodes, to establish direct communication between them, but each fog node has to be connected to the fog server with a one-hop distance. Unfortunately Li et al. do not mention any reason for this restriction [6].

Fog Node A fog node (FN) in EHOPES is adjacent to smart objects and provides processing power as well as storage and communication abilities. As long as the agent software of EHOPES, can run on a device, it can be used as a FN. Additionally they have various interfaces to connect to smart objects, like Wi-Fi or Bluetooth. Their main purpose is to filter repetitive data from smart objects and make decisions, based on which they can take actions. FNs support collaboration between themselves and other FN and can have different capabilities depending on the requirements in a network. Last but not least, FNs are self-configuring and provide routing, security and QoS.

Fog Server The Fog Server (FS) of EHOPES is the bridge between FNs and the cloud. It hosts , not further described applications, and stores data acquired by the FNs. The applications and data are used to support FNs, additional to leasing processing power on request. With an on-demand connection to the cloud, the FS

doi: 10.2313/NET-2018-11-1_05

can work jointly and independent with/from it. Between the Cloud and the FS exists a high speed internet connection, to provide fast access to data stored in the cloud. EHOPES supports multiple FSs within a single fog network, and one physical device can take the roles of FN and FS simultaneously.
The tasks of a FS include advanced routing and switching between FNs, storing of data and applications, configuration of FNs, QoS, security and remote access for management by a maintainer. The remote access includes application deployment, data offloading, network configuration, optimization, billing and other services. How this remote access is designed, is not described by Li et al.

3.2 FRODO

Developed by Seitz et al., FRODO [10] is a fog computing extension to MIBO [9]. MIBO is a framework for multimodal intuitive building controls intended for smart buildings, that utilizes the cloud approach for a centralized decision-making and action-taking core. FRODO moves part of the tasks of the core to the edge of the network, by providing all the functionalities of the FS at each fog node (FN). Thus the FS does only handle decision-making and action-taking on the inter-FN level, while a FN can handle everything on intra-FN level.

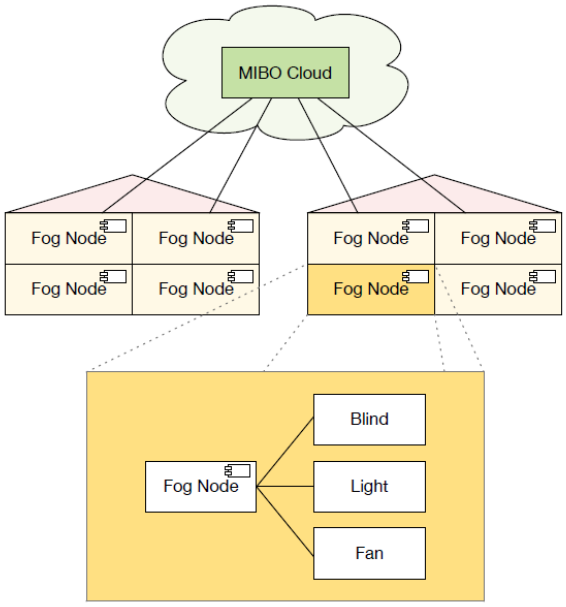


Figure 2: Architecture of FRODO, with FNs distributed on a per-room basis, taken from [10]

FRODO makes decisions based on many rules and definitions, which are used to infer the actions to take. The rules and definitions are stored in such a way, that both FS and FN can access those that are required by them, while ignoring all others. This means that a FN is a tailored clone of the FS with partial knowledge of all rules and definitions.
Rules in MIBO and FRODO are created by privileged users, for example by requesting that "All blinds are closed after 6 pm". Definitions can be created by users, for example by stating: "Open the blinds". If the previous definition is stated after 6 pm, it will contradict the previously specified rule, thus creating a conflict.
MIBO is able to negotiate such conflicts and infer the action that should be taken. FRODO moves this process to the FNs, which has the additional advantage of introducing the context of the location. Thereby decisions can also be easily tailored to specific peculiarities of the location of deployment, which is a huge improvement compared to MIBO.
Additionally FRODO is robust to network outages, since each FN can work independently from the FS. Seitz et al. propose the distribution of FNs based on rooms, as shown in Figure 2, to make rules and decisions as independent and straight forward as possible. The fog node in a room then handles all smart devices within this room. The resulting network between the FS and the FNs resembles a star topology with the FS in the center.

3.3 Gabriel

Gabriel [5] is developed by Ha et al., as an assistive system for wearable devices like Google Glass. It uses fog computing to provide crisp low-latency interaction, by offloading resource intensive processing tasks to nearby Cloudlets (fog nodes). To support mobility of the user, Gabriel supports hand-offs between Cloudlets and has multiple offload strategies implemented. Those strategies include the use of **a)** a nearby Cloudlet **b)** the distant Cloud and **c)** on-body devices like laptops or smartphones. If the Wearable device loses connection to a nearby Cloudlet it queries a Cloud-hosted list of Cloudlets for a new one nearby, that can be used to offload processing tasks. If no one is found, it offloads tasks to the Cloud and if that is also impossible, for example when no connection to the internet can be established, to the on-body device. Gabriel is additionally built to take the different latencies of the offload strategies into account and adjusts the user interface accordingly.

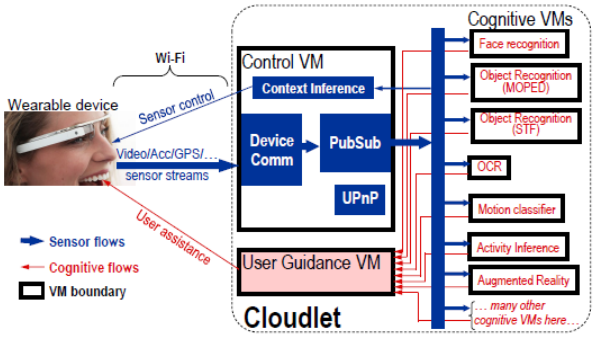


Figure 3: Back-end Processing on Cloudlet (fog node) of Gabriel, taken from [5]

As seen in Figure 3 Gabriel uses independent virtual machines (VM), with specialized cognitive engines running on them, in its Cloudlets, to provide different aspects of image processing. The cognitive engines are thinly wrapped by Gabriel to access the required sensor streams and to publish their results in a defined way. Additional to the virtual machines holding the cognitive engines (Cognitive VMs), a Cloudlet consists of two other VMs.

doi: 10.2313/NET-2018-11-1_05

The Control VM takes care of the the transportation of all required sensor data to the Cognitive VMs via an UPnP server and a publish-subscribe service. The incoming sensor data is additionally preprocessed by the Control VM, to reduce redundant tasks within the Cognitive VMs, like decoding.
The outputs of the Cognitive VMs are handled by the User Guidance VM, which integrates all of them and performs higher level cognitive processing to generate the actual user assistance that is sent to the Google Glass device, which can vary from visual aid to speech guidance.
Gabriel is a highly power saving approach, that goes as far as to switch of sensors when they are not needed and keeps processing on the wearable device very small, by offloading as much work as possible.

3.4 MACaaS

Monitoring, Analyzing and Controlling as a Serivce, in short MACaaS, developed by Byeon et al., is a platform utilizing fog computing to provide the aforementioned services for IoT devices [2]. In contrast to all other introduced frameworks, MACaaS is data-driven and utilizes a learning engine to infer new rules. Since MACaas does not include communication between fog nodes, it has no need for a fog server.

The IoT devices that can be connected to a fog node of MACaaS are separated into two groups, depending on their capabilities. If an IoT device is intelligent enough, to directly connect and communicate with MACaaS, it is classified as High-end IoT Device. If it has not the capabilities to do so (e.g. Low-end IoT Device), it will be first connected to an Edge Gateway, which is connected with the fog node instead of the device itself. The Edge Gateway can be understood as a wrapper for the IoT device, which handles communication with the framework.

A fog node of MACaaS provides all services of the platform namely Monitoring (MaaS), Analyzing (AaaS) and Controlling (CaaS). MaaS handles authentication, saving of generated data, classification of said data and monitoring of IoT device nodes and their status. AaaS tracks changes in users, IoT devices and environmental data and extracts important features of the tracked data through a learning engine, to generate new knowledge. CaaS updates event triggers and modules with the analyzed data and controls the IoT device nodes. For data backup and restoration a fog node can have more than one fog storage.

3.5 NetFATE

A solution tailored to providers of telecommunication services (TELCO) is NetFATE, which is developed by Lombardo et al. [7]. NetFATE incorporates the paradigms of Software Defined Networks (SDN) and Network Functions Virtualization (NFV), additional to fog computing to bring services to the edge of a TELCO network. Noteworthy is that NetFATE only relies on infrastructure components that are already existent in the current infrastructure of a TELCO network. Thus it is easy to deploy, as it does not need any changes to the network.
The aim of NetFATE is to provide the three service delivery models **a)** Infrastructure as a Service, **b)** Platform as a Service and **c)** Network as a Service.

NetFATE separates all its hardware components into three types, that take care of different aspects of the framework. The three logical components required for NetFATE are:

CPE/PE Node The CPE nodes and PE nodes are the fog nodes of NetFATE, since they are equipped with a virtualization framework and provide resources for the execution of VMs with network functions (NF). Physically there are two different types of nodes, the Consumer Premises Equipment (CPE) node, that resides at the location of the consumer and acts as gateway to the TELCO network. Similar to the current setups, each customer has one such CPE node, which is used exclusively by him.
The second type of fog nodes is the Premises Equipment (PE) node, which resides within the TELCO core network and has the task of aggregating many CPE nodes and is therefore used by many customers.
The fog nodes of NetFATE are general purpose computers with CentOS as a base operating system. The framework uses para-virtualization with NFs encapsulated within light network oriented OSs which can be executed on CPE/PE nodes. The nodes use the Xen Hypervisor for para-virtualization and OpenvSwitch to handle the internal network traffic between the VMs and the base OS. Deployed VMs can be migrated to other nodes on request, to support mobility of customers.

Data Centers While data centers are listed as required components, neither in the paper of Lombardo et al. nor in its cited sources exists a definite description for what they are used for.
The according ETSI Group Specification [4] lists them as environment for application virtualization and support for other nodes. Therefore it can be assumed, that the data centers support CPE/PE nodes on request and provide additional resources for processing and storage of data, as well as acting as fog nodes themselves within NetFATE. While this somehow describes their purpose, it does not describe why they are listed as required components, as they do no work that could not be done by other components.

Orchestrator The orchestrator is fog server of the framework. It handles the tasks, mentioned in Section 2 and controls the SDN by realizing routes and configuring the network interfaces of the nodes on the route. The orchestrator is a dedicated server within the telco core network and communicates with the VM hypervisor of the fog nodes via the IP network of the TELCO.
The orchestrator can be separated in three separate entities, **a)** the SDN Controller realized with POX, **b)** the NFV Coordinator, handling all tasks concerning VMs and **c)** the Orchestration Engine, which gathers information on the network and decides how the resources of the devices should be managed.
The information gathered by the Orchestration Engine includes among other things the topology of the network, the number of connected clients and the required network services. The policies used for the management of the resources can range from energy consumption to the number of hops on a route in the SDN.

doi: 10.2313/NET-2018-11-1_05

Framework	supports application migration	can handle loss of			
		cloud connection	device failure	fog node failure	fog server failure
EHOPES	✓	✓	✗	partial	✓
FRODO	✗	✓	✗	✗	partial
Gabriel	✓	✓	not applicable	✓	not applicable
MACaaS	✗	not applicable	✓	✗	not applicable
NetFATE	✓	✗	not applicable	partial	✗
ParaDrop	✓	partial	✗	✗	✗

Table 1: Results of the evaluation of reliability

Since NetFATE is intended for big TELCO networks, it is understandably oversized for smart buildings. While this is the case, the infrastructure does suite buildings very well. When looking at the framework as a hierarchy, the orchestrator with the data center stands at the top, followed by the PE nodes which finally aggregate the CPE nodes.
Comparable to the hierarchy of NetFATE, buildings can also be separated in the same way. As CPE nodes are the outermost nodes of the network in NetFATE, it is intelligent to distribute them room-wise in a building, as services will most likely be used on a per-room basis as well. The next bigger node in the hierarchy of a building would be the floor, which is reflected by PE nodes in NetFATE. Like each floor contains multiple rooms, each PE node aggregates multiple CPE nodes. Additionally when users move from room to room, a service has to be migrated to the PE node only. And only when a user changes floors it has to be migrated to the datacenter and back to the PE node of the new floor. Finally the orchestrator and data center of NetFATE can be placed anywhere in the building, while a uniform distance to all PE nodes is best for latency. As one can see, NetFATE reflects the hierarchy of buildings and thus suits the topic of smart buildings even if it is intended for distinctly bigger use-cases.

3.6 ParaDrop

ParaDrop [12] makes use of the gateway that exists in every home with internet access. Developed by Willis et al. from the University of Wisconsin, ParaDrop uses the gateways as fog nodes and positions the orchestrator in the cloud. The orchestrator of ParaDrop manages deployment and migration of VMs on fog nodes and provides APIs for companies to enable communication between their applications on the fog nodes and their services in the cloud. ParaDrop uses the widely acknowledged RESTful paradigm and a JSON-backend, for the communication between the single components, thereby simplifying development of chutes.
The virtualization in ParaDrop is realized with Linux Containers (LXC), which provides fully, virtually isolated compute containers, called chutes. In comparison to OS-level-virtualization (e.g. Lguest) LXC has the advantage of better performance and less overhead in network operations as well as fairer distribution of resources. As Willis et al. have described in their paper, LXC should be favored for I/O intensive applications. While the advantages of LXC are very interesting for devices with limited resources, the disadvantage is that applications need to be runnable on the base OS, since deployment of fully private OSs is not supported. In essence this means, that the application must be supported by the kernel used by the base OS of the fog node. The developers still claim, that porting of applications to ParaDrop containers is very easy and goes often without the need to rewrite code.
ParaDrop uses OpenFlow for effective networking between the gateway and the chutes, to distribute network resources fairly. Additionally the framework enforces very strict resource policies on CPU, RAM and the network, which can be manually edited via a management interface provided by the orchestrator. Unfortunately Willis et al. present no details on restrictions to the usage of storage space, which is most times just as limited.
Like Gabriel and NetFATE, ParaDrop supports migration of containers from one fog node to another, thus removing restrictions to the mobility of users. Even when being migrated, containers retain the user state and can resume processing of data seamlessly.

4. EVALUATION OF FRAMEWORKS

In this section we evaluate the Frameworks for their suitability for smart buildings. Since the previously described frameworks have mostly other purposes than enabling smart environments, we restrict this evaluation to the basic underlying architecture of them. To enable the evaluation of the frameworks, we use three major criteria, which are described below. The findings for each criteria are presented in a condensed form in Tables 1, 2 and 3.

4.1 Reliability

Power outage, network faults, sensor faults and loss of internet connection are only some of the problems that need to be gracefully handled by frameworks for smart buildings. In the best case, the maintainer is notified instantly and the problem is circumnavigated internally without the normal user being any the wiser, until it is solved, by switching resources or using other algorithms. But in some cases it is impossible to remain in full operational mode, when components fail. Like loss of internet connection, when a user demands a resource that is not locally available. Here frameworks should notify the user but remain operative as far as possible. Lastly smart buildings have to take the safety of inhabitants into account. For example switching on lights in case of a fire, so people can leave safely. All this needs intelligent fallback strategies that imbue all aspects of a framework.

Of the introduced frameworks, Gabriel provides the best strategies for loss of connection to certain components. It gracefully falls back to the next best offload device reliantly, and returns to better options when they are available again. Additionally it can handle faults of Cognitive VMs since it

doi: 10.2313/NET-2018-11-1_05

Framework	hardware independence	virtualization technique	requirements to fog node	requirements to fog server	requirements to infrastructure
EHOPES	✓	unknown	claimed low	unknown	high
FRODO	unknown	not applicable	unknown	unknown	unknown
Gabriel	✓	XEN	medium	not applicable	low
MACaaS	not applicable	not applicable	medium	not applicable	not applicable
NetFATE	✓	XEN	medium	high	low
ParaDrop	✓	LXC	low	not applicable	none

Table 2: Results of the evaluation of hardware requirements

gathers all data within the User Guidance VM which integrates them to the user interface [5]. FRODO has another interesting approach, in deploying nodes that work independently where possible and only rely on other resources when required. Thus providing most functionality even when there are sever problems like network faults in critical connections. The main problem of FRODO is the single fog server that has no backup available [10]. If this server fails, no connection between FENs is possible. EHOPES handles this problem by allowing more than one fog server which can work completely independent from the Cloud if necessary. Unfortunately the paper does not go into detail, how the migration from one server to the other works [6]. MACaaS is a special case when looking at the fog server. While fog nodes contain a subcomponent that is called fog server, one must not confuse this with the similar named physical component in other frameworks. MACaaS is a framework that relies on one single fog node, that is directly connected to IoT devices, with no communication between fog nodes. Thus it has the fog node as single point of failure. MACaaS can handle failing IoT devices very gracefully, with logging, notifying the maintainer and vendor and even updating drivers transparently [2]. Which stands in contrast to all other frameworks, which have absolutely no strategy to handle misbehaving IoT devices.
While most of the frameworks can handle outage of the fog server more or less gracefully, many frameworks have problems when fog nodes fail. The exceptions being Gabriel, EHOPES which can at least redistribute applications to other nodes and only loose the connection to the IoT devices connected to the failing node [7] [6]. NetFATE can redistribute work when a PE node fails, but failing CPE nodes can not be compensated, since no other connection to the customer exists.
The framework with the worst reliability is ParaDrop, since the orchestrator is positioned in the cloud. Therefore it requires internet connection for installing and migrating applications and maintanance of nodes by the network maintainer, and fails at doing these tasks when the node looses internet connection. Fortunately ParaDrop can handle loss of connection to the cloud when the previously mentioned tasks are not required by the fog node [12].

4.2 Hardware Requirements

For the practical use of a framework in a smart building not only the features it provides are of interest, but also the requirements in hardware and infrastructure. For example deploying expensive specialized hardware as fog nodes is not viable for big buildings, since the cost would be immense. Thus frameworks with low hardware requirements, that can use cheap nodes like a Raspberry Pi and are platform independent, will be favored against ones with high restrictions to hardware. Additionally frameworks with low hardware requirements make it possible to use "leftover" computational power of smart devices, that are deployed in a building and are not powerful enough to support demanding frameworks. Finally the hardware should also be power efficient to minimize maintenance cost.

With the exception of EHOPES and FRODO all frameworks provide at least some data, with which it is possible to infer their actual hardware requirements. ParaDrop requires the least amount of resources, with the orchestrator being maintained by the developers of ParaDrop itself. Thus it does not count to the actual nodes required for deployment. Additionally the fog node runs on consumer routers, comparable to ones currently distributed by TELCOs and is available as VM that can be deployed to a virtualisation environment [8]. In the paper of Willis et al. they are able to provide two additional applications to the standard router and gateway functionality on a router with an AMD Geode 500MHz CPU and 256MB of RAM as the fog node. This is possible because ParaDrop uses LXC for virtualization and not Lguest or a comparable method, which is more resource intensive [12].
While ParaDrop shines in terms of hardware requirements, MACaaS is able to work with little more resources. Byeon et al. provide no direct data on the hardware of the fog node used in MACaaS, but the applications installed on it, being Nginx, OpenMediaVault and MySQL running on Linux, point to very low requirements. To simulate the IoT device nodes, Byeon et al. use smartphones in the mid-range price segment and Raspberry Pi 2[2].
Gabriel uses for its Cloudlets clusters of four desktop machines with an Intel® Core™ i7-3770 and 32GB of RAM, calling these machines modest [5]. While this amount of computing power and RAM is partially required for the resource intensive computations the Cognitive VMs need to handle, this is also owed to the use of para-virtualization in contrast to application virtualization, since it has higher overhead, as tested by Willis et al. in their paper [12]. Therefore it can be assumed, that fog nodes of Gabriel have higher resource demands compared to ParaDrop.
Still Gabriel supports fallback strategies to less powerful devices, like smart phones, and should therefore have scalable hardware requirements. But the paper does not include if the decrease in functionality, when migrating to a fog node with less power, is only in terms of Cognitive VMs or the overall system.
In contrast to the prior frameworks NetFATE has signifi-

Framework	soft-realtime capable	chosen approach
EHOPES	✓	cooperation of fog nodes
FRODO	✓	reduction of latency by local decision-making
Gabriel	✓	choosing of most powerful fog node with lowest latency
MACaaS	unknown	
NetFATE	✓	choosing of capable node
ParaDrop	✓	use of low latency virtualization method

Table 3: Results of the evaluation of soft-realtime capability

cantly higher requirements. Intended for huge networks of TELCOs, it is understandably completely oversized when looking at smart buildings. Mainly the orchestration engine on each fog node, building on the Xen Hypervisor for para-virtualization of the NFs, requires at least a general-purpose computer, which makes deployment in smart buildings costly and possibly not viable. It is up for debate, how far NetFATE could be trimmed down to be in a competitive position to ParaDrop or MACaaS.
While EHOPES has no data on the actual hardware requirements, the paper of Li et al. mentions independence from hardware provided by Foglet and claims low requirements to the fog edge nodes. The main disadvantage of EHOPES is the restriction that fog server and fog nodes must be in one-hop proximity. Whether this refers to the term "hop" used in networking, thus disallowing switches or routers inbetween nodes, or means that no FENs are stringed together is not described in the paper [6]. If it is meant in the former we postulate that deployment in larger buildings will be very hard to realize with this restriction. Thus making EHOPES only viable for smaller buildings like small homes. If Li et al. mean stringing together FENs, it has no detrimental impact on deployment in large buildings.

4.3 Soft-Realtime Capability

The inhabitants of a smart building will evaluate the framework on reliability, evaluated in Section 4.1, and latency, since those are the two aspects which influence the user interface the most. In other terms the soft-realtime capability of a framework is as important as reliability. For example switching the light on and the framework actually switching the intelligent light bulbs on, should be two events with at the most a few seconds in between. While this scenario is not very resource intensive, there are other scenarios like streaming digital content on-demand that are harder to accomplish in nearly realtime. Therefore it is important that frameworks can handle tasks in soft-realtime, no matter how resource intensive they are.

The most obvious approach to support soft-realtime, ist to minimize latency wherever possible. FRODO does this by making decisions, concerning IoT-devices connected to a single fog node, internally in this node and thereby saving the hop to the fog server [10]. Willis et al. decided against OS-level-virtualization and for containerization to minimize latency and overhead and distribute resources more fairly in fog nodes of ParaDrop [12]. Gabriel implements the elaborate fallback strategies, additional to the restriction of communication between Cloudlet and cloud to non-time-critical parts of the code [5]. Finally Li et al. do also claim that EHOPES has very little overhead, but do not go into further detail [6].
While all those approaches are essential to enable realtime capability, they assume that one fog node will be able to handle the task on its own. This may be true for most tasks, but sometimes the physical capabilities of a fog node are inadequate for a certain task. Only NetFATE and EHOPES have implemented additional measures, to support these cases. EHOPES enables fog edge nodes to collaborate via Foglet and share resources, thereby working jointly to handle the workload. NetFATE does not allow such cooperation, but the orchestrator selects a node that is capable of handling the task on its own [7]. While this still assumes that there is a node within the network that can handle the task, it does not require every single fog node to provide the resources needed to handle every possible task. Thus allowing lightweight CPE/PE nodes and a heavy-duty node in the form of a data center. Of course this comes with additional latency, since the data center might not be in the vicinity of the user.

5. CONCLUSION

Fog computing plays a central part in enabling IoT. Especially for latency sensitive applications, like smart buildings, it can be of great advantage. None of the introduced frameworks include all necessary parts to be a perfect fit for smart buildings. Still each framework has its unique approach, providing different interesting advantages.
EHOPES brings the best reliability concerning fails of major components, with the possibility to duplicate the fog server. Additionally it handles resource intensive tasks gracefully by collaboration between multiple FENs. This approach is in some way mirrored by FRODO, where fog nodes are able to make decisions independently when they only concern smart devices attached to themselves, which ensures valuable functionality in emergency situations. MACaaS is the framework of choice, if many devices should be connected that are not able to communicate with the IoT on their own. But with the missing cooperation between multiple fog nodes it is not viable for large buildings. Finally ParaDrop shows an efficient framework for small lightweight nodes, that is easily distributed in buildings, but does not have the necessary reliability for smart buildings.
As described in Section 3.5, NetFATE is in terms of infrastructure a perfect fit for large smart buildings. Therefore it is a great start to create a framework for such buildings. To be really suitable for this scenario, it needs down-scaling of especially the CPE nodes, which can be done by utilizing the approach of ParaDrop for example. To circumvent the single point of failure the orchestrator presents, EHOPES and FRODO show great potential. When inhabitants of the building are on the move, Gabriel presents a nice strategy,

which nodes should be used as the current offload device. And finally MACaaS can be incorporated, to support even devices that existed prior to deploying the framework in the building.
Thus a suitable framework for smart buildings is a conglomeration of all introduced frameworks.

6. REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[2] G. Byeon, M. Shon, H. Lee, and J. Hong. Macaas platform for fog computing. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, RACS '17, pages 287–292, New York, NY, USA, 2017. ACM.

[3] A. V. Dastjerdi and R. Buyya. Fog computing: Helping the internet of things realize its potential. *Computer*, 49(8):112–116, Aug 2016.

[4] ETSI GS NFV 002. Standard V1.1.1, ETSI, 650 Route des Lucioles, F-06921 Sopia Antipolis Cedex, FR, 10 2013.

[5] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan. Towards wearable cognitive assistance. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages 68–81, New York, NY, USA, 2014. ACM.

[6] J. Li, J. Jin, D. Yuan, M. Palaniswami, and K. Moessner. Ehopes: Data-centered fog platform for smart living. In *Proceedings of the 2015 International Telecommunication Networks and Applications Conference (ITNAC)*, ITNAC '15, pages 308–313, Washington, DC, USA, 2015. IEEE Computer Society.

[7] A. Lombardo, A. Manzalini, G. Schembra, G. Faraci, C. Rametta, and V. Riccobene. An open framework to enable netfate (network functions at the edge). In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–6, April 2015.

[8] ParaDrop. Paradrop v0.10.3 documentation, 2017. Retrieved March 31, 2018 from http://paradrop.readthedocs.io/en/latest/device/.

[9] S. M. Peters. *MIBO - A Framework for the Integration of Multimodal Intuitive Controls in Smart Buildings*. PhD thesis, Technische Universität München, 2016.

[10] A. Seitz, J. O. Johanssen, B. Bruegge, V. Loftness, V. Hartkopf, and M. Sturm. A fog architecture for decentralized decision making in smart buildings. In *Proceedings of the 2Nd International Workshop on Science of Smart City Operations and Platforms Engineering*, SCOPE '17, pages 34–39, New York, NY, USA, 2017. ACM.

[11] P. O. Östberg, J. Byrne, P. Casari, P. Eardley, A. F. Anta, J. Forsman, J. Kennedy, T. L. Duc, M. N. Mariño, R. Loomba, M. . L. Peña, J. L. Veiga, T. Lynn, V. Mancuso, S. Svorobej, A. Torneus, S. Wesner, P. Willis, and J. Domaschka. Reliable capacity provisioning for distributed cloud/edge/fog computing applications. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–6, June 2017.

[12] D. Willis, A. Dasgupta, and S. Banerjee. Paradrop: A multi-tenant platform to dynamically install third party services on wireless gateways. In *Proceedings of the 9th ACM Workshop on Mobility in the Evolving Internet Architecture*, MobiArch '14, pages 43–48, New York, NY, USA, 2014. ACM.

[13] S. Yi, C. Li, and Q. Li. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, Mobidata '15, pages 37–42, New York, NY, USA, 2015. ACM.

Self-Driven Computer Networks

Simon Klimaschka
Advisor: Fabien Geyer
Seminar Future Internet SS2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: simon.klimaschka@tum.de

ABSTRACT

This paper provides an overview about the different kinds of machine learning algorithms that can be used for detecting anomalies in computer networks regarding how the different algorithms work and how they are beneficial for anomaly detection in computer networks. It also provides an overview about which paper uses which machine learning technique and how those techniques worked out. Furthermore, it gives an introduction to anomaly detection with graphs and what advantages graph-based anomaly detection has compared to machine learning based approaches. In the end, the paper presents the implementation of a supervised and an unsupervised machine learning algorithm for anomaly detection from the scikit module and evaluates the used algorithms in terms of overall detection performance and computing time.

Keywords

anomaly detection, machine learning, graph, computer, network, scikit

1. INTRODUCTION

One of the inventions, that brought humanity the most advantages is undoubtedly the internet. While in the beginning very few people used this new achievement, the e-mail traffic grew very fast in the first years. The web pages in these days were very lightweight and hacker attacks not very frequent. But in the past years, more and more services use the internet. Nowadays, nearly everyone owns a smartphone, uploads their data to the cloud, streams movies from services like Netflix or Amazon Prime or can look up nearly every information they want to. Companies also use the internet to be more connected and use it to connect their facilities. The Internet of Things (IoT) will enable devices like light bulbs, refrigerators or a door bell to be connected via the internet. This will result in billions of new devices in our networks. As can be see, the poor security in those devices can result in big attacks like the mirai attack [1]. With so many different devices and very diverse usages, it gets more attractive for hackers to intrude networks and devices. Possible attack scenarios are stealing classified data from company networks and blackmailing them, infiltrating military networks to gather data about upcoming missions, hacking the networks of banks and payment service providers to make money or even distributing malicious software through the whole internet like in the WannaCry attack [2] which affected railway services as well as hospitals. All these attack ideas underline, that we need systems, which can detect anomalies in networks, which often are attacks. To get an idea about how anomaly detection in computer networks works, we will observe two different ideas. In section 2, we will address the different machine learning approaches and how they can be beneficial for anomaly detection. Section 3 is dedicated to how graphs can be used to detect anomalies, as well as comparing graph-based approaches to machine learning approaches with an eye on their respective advantages. In the end, we will evaluate two implementations of a supervised and an unsupervised machine learning approach and compare them regarding their detection accuracy and computation time.

2. ANOMALY DETECTION IN COMPUTER NETWORKS USING MACHINE LEARNING

Anomaly detection plays an important role in computer networks nowadays, especially regarding security and stability. Agrawal and Agrawal [3] define "Anomaly detection [a]s the process of finding the patterns in a dataset whose behavior is not normal on(sic!) expected."
Spotting anomalies means spotting behaviours which do not occur very often and are most likely not wanted. As we want to focus on anomaly detection in computer networks in this paper, this could happen during an attack by hackers or due to malfunctioning hardware or software. As can be assumed, classifying packets is not trivial at all, many factors have to be considered.
Machine learning is very good at tasks like classifying high-dimensional data or reducing high-dimensional data to lower dimensions. To further survey machine learning for anomaly detection, we have to consider the different machine learning techniques at first, regarding how they work, for which cases they will work and if it will be beneficial for anomaly detection in computer networks:

- **Artificial Neural Networks:** Artificial Neural Networks (ANNs) can "approximate any given function" [9], which also includes anything non-mathematical, like image recognition or natural language processing. ANNs consist of so called neurons, which are connected to each other. The neuron is based on the way the neurons in the brains work, hence the similar names. In which manner they are connected depends on the style of the network.
  A feed forward network for example has an input layer, an output layer and an adjustable number of hidden layers. Every neuron in a layer is connected to the

doi: 10.2313/NET-2018-11-1_06

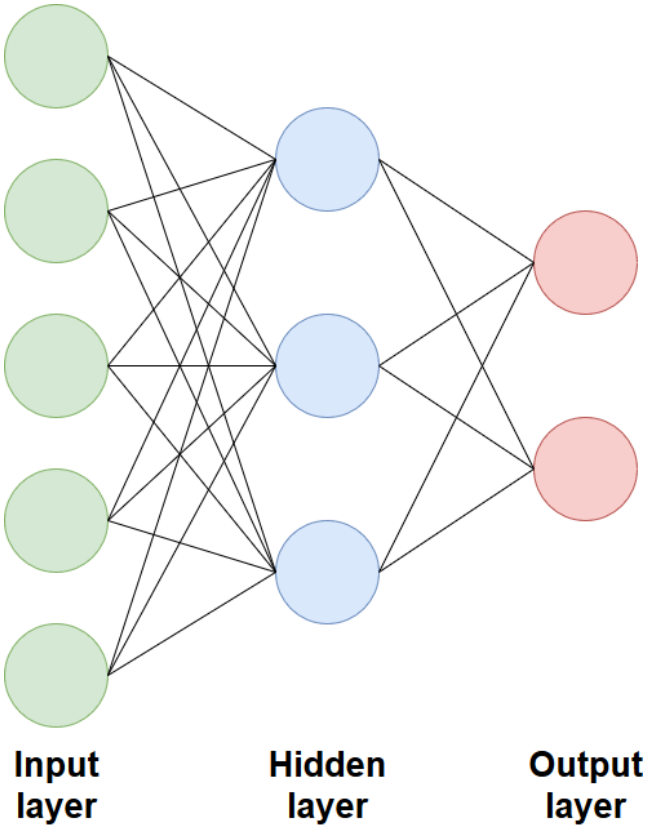


Figure 1: Neural Network

neurons of the anterior and posterior layer, as can be seen in figure 1. Every connection has a corresponding weight, which is altered depending on the usage of the network. Altering the weights of a network to a specific use-case is called training. When fitting an ANN, we use the properties of supervised training:

- In supervised training, a training data set has to consist of the input values and the expected output values. Regarding a specific input, the output of the network is calculated and compared to the expected output. Based on the rules of backpropagation the weights of the network are adapted. For anomaly detection, the ANN could be design with one output neuron and trained to output 1 for anomalies and 0 for normal packages. While the result is exactly what we want, the training process needs a lot of work done in advance for our use-case. Typical training data sets consist of hundreds of thousands of data samples which all need to be classified as normal or abnormal. This is not only a huge workload, different persons also classify the same packages differently, even the same persons tend to change their mind when classifying the same packages. [4, Section 1.2]. Furthermore, anomalies are per definition unusual and uncommon, which makes them even harder to spot. Lippmann and Cunningham [14] used a ANN with selected keywords as inputs. The ANN was able to detect 80% of all occurring anomalies with just about 1 false alarm per day. Palagiri and Chandrika [17] used a neural network trained with data from the DARPA 1999 training data set. While detecting all normal packages correctly, it could only identify 24% of all attacks, 76% were falsely classified as normal. The main problem they had was the low number of occurrences of attacks in the data set. When they trained the network for just one attack, they were able to detect all attacks with a false positive rate of 0%.
  This leads to the main problem with neural networks: They provide excellent results for very specific purposes, like a single attack, but get weaker the more attacks they should be able to recognize. They also struggle with detecting attacks they were not trained for. A solution for this could be creating an own neural network for every attack. This won't solve the new-attack problem, but will increase the rate of detected attacks, although it requires training many times over. Though the training time is relatively long, the detection itself can be done in quasi real-time.
- Unsupervised training, in contrary, doesn't need classified training data, it makes the classification internally by itself. An unsupervised learning technique is described in the subsection Clustering.

- **Fuzzy Rules:** Fuzzy sets are an extension of normal sets. In a conventional set, an object either is or isn't a member of the set. For example, the number 4 is a member of the set 'even numbers' but 5 isn't. Fuzzy logic allows objects to be a member of a set in a certain percentage. In a conventional set, 29°C will be considered as normal, but 30°C already as hot. The difference is not that big, but the outcome is very different. In fuzzy logic, 29°C can be a member of warm with 60% and a member of hot with 40%. For anomaly detection in networks, fuzzy rules have the big downside, that the training cannot be done automatically like with neural networks. Though neural networks could help with for example feature reduction, the selection of features and the definition of the rules has to be done by the system administrator with his experience. This yields the risk, that the administrator forgets certain attacks or might even not be aware of them being existent. How well fuzzy logic detects anomalies is therefore dependent on the system administrator creating the rules. However, the construction of the if-then rules is easier than describing attacks in a normal fashion, as it mimics human thinking [7]. Fuzzy logic can also work with other data mining algorithms and extend them, providing more abstract and flexible methods for creating rules [10]. Thus, fuzzy logic can be powerful when set up correctly.
- **Bayesian Networks:** When working with a network we often know the statistical features between the variables in our network, but keeping an eye over all of them and how they all correlate is not an easy task. Bayesian networks can close this gap and provide an overview over the variables and their relations with a probabilistic graph model, represented by a direct

doi: 10.2313/NET-2018-11-1_06

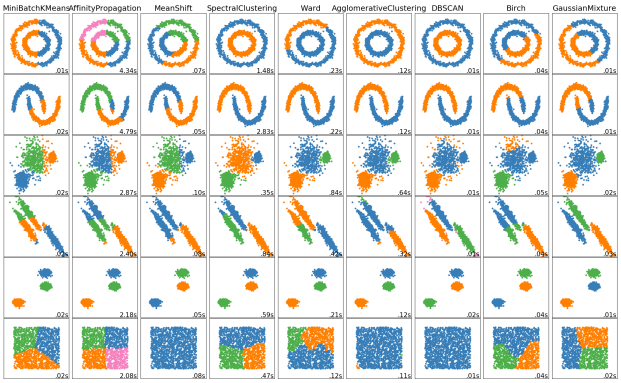


Figure 2: A comparison of the clustering algorithms in scikit-learn [20]

acyclic graph with each node being one system variable and every edge showing that one node is influenced by another. This system could answer questions like with which probability an attack is happening if the system variables have a certain setting [22]. The problem with this technique is that you need to know about the statistical dependencies of your networks variables, which is not given very often, especially not in new networks, as it requires experience in working with networks.

- **Clustering:** This technique doesn't need labeled data, it's an unsupervised algorithm. Clustering is able to discover structures and patterns in any-dimensional data. There exist various methods:
  - Connectivity models group the data points based on the distance between them
  - Distribution models assume, that groups are acquiescent to a statistical distribution
  - Density models use dense regions and connected regions to group data points
  - Graph models sets of connected nodes describe each cluster while every node holds an edge to at minimum another node in the set [5]

  The big advantage for clustering is it's ability to learn from arbitrary data and that it doesn't need a supervisor which defines possible attack scenarios. This freedom also comes with a downside, to get acceptable results we have to choose an appropriate algorithm. As can be seen in figure 2, depending on the organization of the data points, an algorithm could be better than another one. In terms of computing we also should consider, that some algorithms like DBSCAN don't scale very well as they are very memory demanding. Leung and Leckie [13] used the approach fpMAFIA described in their paper, a density- and grid-based high dimensional clustering algorithm. It has a high detection rate, but a rather high false positive rate compared to other approaches like k-nearest neighbour or support vector machines. How well clustering performs is dependent on the choice of the used algorithm and how the data is structured.

- **Ensemble learning:** Ensemble learning refers to the approach of combining several weak learning algorithms to improve their total performance [22]. As it depends on the choice of algorithms and their connection how this technique performs, we will not look any further into it.

- **Evolutionary/Genetic:** Genetic learning is inspired by nature and the principle of evolution. We start with organizing the problem structure in a chromosome like data structure. To solve the problem, the genetic algorithm uses the chromosomes of the best solutions and mixes them together to create new chromosomes. This leads, given some slight randomization, to a better solution every epoch. In computer networks, Khan [11] uses genetic algorithms to develop detection rules. Every chromosome consists of genes like the used service, if the user is logged in, uses a root shell or is a guest. Genetic algorithms have the advantage, that it searches for the best anomaly finding solutions from all directions, whilst needing no prior knowledge of the network. However, this can take a while, the computing power needed is comparatively high [8].

- **Support Vector Machines:** Support Vector Machines (SVMs) are able to classify data points into two different classes. The two classes are divided by a hyperplane, which is outlined by a set of support vectors which have to be members of the training input. In contrast to clustering with the k-nearest neighbours approach, SVMs can handle big dimensions, because an upper bound is set on the margin between the different classes. SVMs have the crucial advantages, that their real-time performance can be better than neural networks and they are easily scalable, in the size of the data points as well as for dimensionality [15]. Mukkamala et al. [15] concluded, that the time to train a SVM is significantly shorter than for neural networks (17.77s and 18min) while delivering the same accuracy in detecting anomalies. The downside with using SVMs is that you can only make binary decisions like is anomalous or not. This could become a problem, if the attacks are very different and we would need to divide into different kinds of anomalies to detect them correctly [15].

- **Self-Organized Maps:** Kohonens Self-Organizing Maps [12] are able to convert many-dimensional data into a two-dimensional map using the internal representation of a map. The method a self-organizing map (SOM) is trained, is related to competitive learning. All points of the map are initialized randomly, then the training data is presented to the map. The map-point, which is fitting the best to the data-point, is pulled towards the data-point. All surrounding map-points are pulled towards the data-point too, but the further they are apart, the less they are pulled. As the training progresses, the points are pulled a little bit less. The SOM creates an internal representation of the given data in the process. In our use-case, the SOM classifies all packet classes, without any prior definitions, just by the input data [21]. A restraint for SOMs is that it only considers the bulk of a traffic class, not the exception that occur seldom, which are no anomalies, but would be classified as one by the SOM [19].

doi: 10.2313/NET-2018-11-1_06

2.1 Conclusion

Anomaly detection with machine learning plays it's advantages if we already have collected a lot of data from the network. This, for example, could be a recording of all packages that are passing the network. The more training data we have, the more accurate our model gets. The training itself can be very costly and tedious, but when trained, it delivers very accurate results. In the last two years hybrid approaches were mainly used, for example combining PSO and K-means with fuzzy logic [10] or combining Naive Bayes and CF-KNN to gain better results than with just one algorithm [16]. A comparison between different implementations in papers can be seen in Table 1.

3. GRAPH BASED ANOMALY DETECTION

3.1 Overview

In section 2 we focused on anomaly detection in data sets which are based on multi-dimensional data points. If we consider, that the relationship between such data points can be important, we should rethink our model. For example, in big networks, the structure of the network and the dependencies are absolutely essential for detecting anomalies. An easy but powerful representation for such cases are graphs. They are described by nodes which are connected with each other by edges. Obviously, our old techniques for detecting anomalies are not compatible with graph-based data, so we need to develop new ideas.

3.2 Graphs and anomaly detection

Before we can discuss how graph based anomaly detection can be beneficial for computer networks, we have to figure out how anomaly detection in graphs works in general.
Akoglu et al. [4] differentiate between anomaly detection in static graphs and in dynamic graphs.
In static graphs, used methods are further divided into structure-based (searching for frequent patterns in the graphs and subgraphs) and community-based (searching for nodes that belong together as a community) methods. For the former, the following approaches exist:

- Feature-based approach: Extracts structural graph-centric features from the graph layout and combines them with other features already known from other sources. This remodels the problem to an anomaly detection problem with multi-dimensional data points, which we already covered in section 2.
- Promixity-based approach: The main goal of this approach is to highlight the closeness of nodes in the graph, as objects that are very close to each other often belong to the same class (e.g if one person is involved in crimes, his or her acquaintances are likely to also be involved).

For the latter, the main idea is trying to find communities, like a group of friends or servers that work closely together, which are densely connected. Detecting anomalies is then basically just finding connections, that are not in between communities.
Static graphs can also have attributes, for example interests of the users of a social network or purposes of a server (http, mail, ftp, ...) for a computer network. The anomaly detection in those graphs are for the most part the same, so the main idea is still spotting structures or communities. For the latter, outliers in a community can now also be spotted, like if there is one smoker in a community of fitness enthusiasts. A new method in attributed static graphs are relational learning based methods, which can detect much more complicated relationships between nodes to group them into normal and anomalous nodes.
In dynamic graphs, we focus on detecting anomalies in dynamic or time-evolving (meaning a sequence) graphs. There exist four ways:

- Feature-based events: In graphs that evolve, consecutive graphs should have similar properties like diameter or degree distribution. This approach considers a time step as anomalous, if the properties change too much. It is very crucial to choose the right properties that will be observed, as they change depending on the problem.
- Decomposition-based events: This approach converts the time-series of graphs into matrices or tensors, which then are interpreted with chosen eigenvectors, eigenvalues or similar indicators.
- Community- or Clustering-based events: Here, the focus is not on monitoring the network as a whole but detecting communities or clusters and watching the structural change in them.
- Window-based events: This approach works with time windows, e.g. a predefined number of previous graphs is taken into account to decide whether the current graph is anomalous or not. They represent the largest category of anomaly detection methods in dynamic graphs.

This is just a very shallow outline of how graph-based anomaly detection works. For more details, look into [4].

3.3 Graph based anomaly detection for computer networks

Computer networks can be easily represented with graphs, as their structure is very similar, with computers and servers acting as nodes and connections between them being edges. Those edges might be weighted with the current load or data flow. We assume, that the weight of edges changes when the computer network is under attack. The two big problems we face are on the one hand, that watching just single nodes and edges will miss all dependencies in the graph, which would destroy one big advantage, graph-based anomaly detection has. On the other hand, big networks with a lot of edges get very expensive to monitor in parallel.
In conclusion, graph based anomaly detection has advantages when having a big structured network and the knowledge, how the structure looks like exactly. This approach is also very powerful when we deal with newly set up networks for which we do not have any previous data.

4. COMPARISON OF THE IMPLEMENTATIONS OF TWO MACHINE LEARNING APPROACHES

doi: 10.2313/NET-2018-11-1_06

Table 1: Comparison of machine learning algorithms used for anomaly detection

Reference	Use-case	Machine learning method	Result
[14]	Detecting attacks with keywords and DARPA Intrusion Detection Data Base [6]	Neural Network	80% of attacks detected with one false alarm per day
[17]	Detecting attacks with DARPA Intrusion Detection Data Base [6]	Neural Network and SOM	24% of all attacks correctly predicted, when trained with one attack 100% prediction rate
[7]	Detecting attacks with own data set	Fuzzy networks	Detected nine TCP port scans and four ping scans in three weeks
[10]	Detecting attacks with generated dataset from CCNx software	Fuzzy anomaly detection based on hybridization of PSO and K-means clustering algorithms over Content-Centric Networks	Detection rate of 100% with a false positive rate of 1,847%
[5]	Detecting attacks with KDD Cup 1999 Data Set	Evolutionary Computations	Detection rate of 94,19% with false positive rate of 5,81%
[13]	Detecting attacks with KDD Cup 1999 Data Set	Cluster detection using fpMAFIA	Performance value of 0,867
[11]	Detecting attacks with KDD Cup 1999 Data Set	Genetic algorithms	Reliability of 94,64% to detect an attack and 1% difference of detecting attacks between training and test data
[15]	Detecting attacks with 1998 DARPA Intrusion Detection Data Set	Support Vector Machine	Detection rate of 99,5% while needing just 17s training time (compared to 18min for neural network)
[19]	Detecting attacks with MIT Lincoln Labratory DARPA Intrusion Detection Evaluation Project data set	Self organizing maps	1,19% false positive rate for DNS, 1,16% false positive rate for HTTP

4.1 The used data set

The data set used for testing the following implementations is the KDD Cup 1999 Data Set. It was used for the Third International Knowledge Discovery and Data Mining Tools Competition to build a network intrusion detector which should be able to detect attacks. The data contains nine weeks of TCP dumps simulating a U.S. Air Force LAN which was under attack to generate the anomalous data. The whole data set is labeled with not just normal or anomalous, it also contains information about the type of the attack. The attacks can be classified into four different categories:

- Denial of service attacks
- Unauthorized access from a remote machine like guessing a password
- Unauthorized access to local superuser privileges
- Surveillance or probing activities like port scanning

The test data also includes attacks which did not occur in the training set to create a more genuine simulation, as in the real world new attacks are discovered all the time.

4.2 Programming language and machine learning library

The most important goal for these implementations were fast prototyping whilst performance could be neglected. The tests should provide an overview about the different approaches and how they perform related to each other, so it's not important that the implementation itself is the fastest. For this reason, Python was chosen as a programming language. It's rather easy to code in Python, providing a lot of built-in features that make preprocessing a lot easier while still providing decent performance. There also exist a lot of libraries for Python that fit our needs.
The chosen machine learning library was scikit-learn [18], as it's available for free and provides packages for classification, regression, clustering or dimensionality reduction and different approaches for preprocessing data.

4.3 Preprocessing

The data from the KDD Cup 1999 Data Set is organized linewise. Every line represents one connection with features like the duration, the used protocol, sent bytes or if it attempted to gain superuser rights. The last element is not a feature, it's the classification of the connection. They are 'normal' in most cases, but also contain attacks like 'buffer_overflow', 'guess_passwd' or 'rootkit'. Not only the classification itself is a string, but also the second, third and fourth feature are no numeric values but a string. Machine learning algorithms can only work with numeric values, so we have to convert all strings to numeric values. At first, all string valued features were taken out of the training data and collected in one array each. From the preprocessing package, we used the LabelEncoder, which provides a normalization of labels. It can also convert hashable and comparable strings into normalized numeric values by hashing them at first and them normalizing them. Every features LabelEncoder instance was fit to the data of the repective feature and then transformed into a numeric value which can be used for machine

doi: 10.2313/NET-2018-11-1_06

learning. After the process, the features were add back into the data set.

4.4 The testing system

The code was tested on a laptop from ASUS, with a Intel Core i5-7200U CPU and 24 GB of memory. The data set was saved on a SSD, the code was also run on a SSD.

4.5 A supervised learning approach

For the supervised learning approach, a standard neural network should be evaluated. From the skicit-learn package, the MLPClassifier was chosen. It implements a multi layer perceptron (MLP) using backpropagation for training, which describes the approach we made in section 2 for the supervised network. The MLPClassifier has some parameters which can be altered, like the use of which activation function for the hidden layers, the used solver, the size of the hidden layers or if it should implement early stopping. Early stopping refers to the technique of reserving 10% of the training data for validation purposes and stopping the training if the results of the validation data set did not improve for more than two epochs.
To find the best parameters for anomaly detection, the algorithm was fed with the training data from the KDD Cup 1999 Data Set and validated with it's testing data set. The results for the different parameters can be found in table 2. One can easily see, that more neurons does not mean better accuracy, but introducing a second layer resulted in just a fifth false alarms. Sgd performed not as well as adam, while being a lot better in recognizing the anomalies, it generates a lot of false alarms. The activation functions didn't have a big impact on the results, just the identity function generates worse results. It took about 4,5 minutes to run on the testing system. The following parameters were tested:

- **Solver:** The solver for weight optimization, choosing from lbfgs, which wasn't used as it's for smaller data sets, sgd, the stochastic gradient and adam, a stochastic-gradient optimizer which works good for bigger data sets.
- **Hidden Layer Size:** The hidden layers are denoted with parentheses and commas, where every part represents the number of neurons in a layer, e.g. (30,10,5) are 30 neurons in the first hidden layer, 10 neurons in the second and 5 in the third.
- **Activation Function:** The activation function used for the neurons of the hidden layer, with the following possibilities: Identity (f(x) = x), Logistic (logistic sigmoid function), Tanh (the hyberbolic tan function) and relu (the rectified linear unit function)
- **Correct Predictions:** How many out of the 311.030 connections were predicted correctly
- **False negatives:** How many percent of the wrong predictions were classified as normal, but are an attack
- **False positives:** How many percent of the wrong predictions were classified as attack, but are a normal connection

4.6 An unsupervised learning approach

For the unsupervised learning approach, the k-means algorithm from the scikit-learn package was chosen. Other algorithms like DBSCAN or Birch, which would also make sense for our use-case and can handle high dimensional data were not able to run on the previously describe testing system as they were too memory demanding.
K-means tries to cluster the data into any desired number of clusters of equal variance. The goal of the algorithm is to choose the centroids of the clusters in such a way, that the distance to all points belonging to it will be as small as possible. This raises a few problems when having elongated clusters or other irregular shapes, because the algorithm works best for convex blobs of points. For high dimensions, the euclidian distance between points become more and more irrelevant, which is also known as the curse of dimensionality.
K-means doesn't have much parameters that can be changed or would be useful changing. The number of clusters for our example is fixed, as we only want to distinguish between normal and anomalous. We could also take the number of different attacks as the number of clusters and it could be working better, but in a normal environment we don't know how many attacks occur in our training sample. If we analyzed the training samples the major advantage that we don't have to look into the data would be completely gone. The programmed algorithm ends with 60.593 wrong predictions. It took about 4,5 minutes on the testing system.

4.7 Comparison of the two approaches

The supervised algorithm outperformed the unsupervised algorithm in nearly every setting. This was expectable, the supervised algorithm knows how to trim it's network to get the best results while the unsupervised algorithm can only depend on the structural organization of the data points when detecting anomalies. We also have to consider the fact, that in a real world environment, we would have to classify the training data before being able to use the supervised algorithm. While the best setup for the supervised variant had about 92% correct predictions (23000 wrong predictions), the unsupervised variant is just 37.000 worse and works without classifying the data beforehand. Nevertheless, with feature reduction or other related techniques, the results of the unsupervised algorithm can be enhanced.

5. CONCLUSION

The paper provides a survey about the different types of anomaly detection existent for computer networks. At first, the different machine learning techniques were explained and evaluated in the terms of their usability in anomaly detection for computer networks. Then we examined graph based anomaly detection in terms of how it works and what benefits it has for anomaly detection in computer networks, especially in comparison to machine learning based approaches. To round this off, we conclude the survey with an overview of machine learning methods used in different papers with their respective outcomes. In the end, a example data set was taken and two algorithms were evaluated, a supervised machine learning algorithm and a unsupervised one. Furthermore the supervised algorithm was tested with different parameters. Nearly all settings outperformed the unsupervised algorithm. Generally, there are a lot of different algorithms that can be used for anomaly detection, but there

doi: 10.2313/NET-2018-11-1_06

Table 2: Result of the supervised learning implementation

Solver	Hidden Layer Size	Activation Function	Correct Predictions	False Negatives	False Positives
adam	(30)	relu	92.33%	95.22%	4.78%
adam	(15)	relu	85.09%	97.78%	2.22%
adam	(45)	relu	91.96%	95.88%	4.12%
adam	(30,10)	relu	92.11%	98.85%	1.15%
adam	(30,20,5)	relu	91.88%	98.82%	1.18%
adam	(80,55,30,10,5)	relu	92.00%	98.88%	1.12%
adam	(30)	identity	81.81%	97.82%	2.18%
adam	(30,20,5)	identity	89.48%	96.71%	3.29%
adam	(30)	logistic	91.89%	96.43%	3.57%
adam	(30,20,5)	logistic	91.94%	97.99%	2.01%
adam	(30)	tanh	91.91%	96.40%	3.60%
adam	(30,20,5)	tanh	91.73%	96.52%	3.48%
sgd	(30)	tanh	90.91%	96.00%	4.00%
sgd	(30,20,5)	tanh	81.39%	98.40%	1.60%

is no best algorithm for all cases. When selecting one, we always have to think about the structure of the data and what our goal is. Newer papers also try to combine different algorithms to use their best properties and combine them with for example fuzzy logic. For future work, one could look into how the different algorithms could be combined the best and which difficulties occur.

6. REFERENCES

[1] The mirai botnet: all about the latest malware ddos attack type. `https://www.corero.com/resources/ddos-attack-types/mirai-botnet-ddos-attack.html`. Accessed: 31.03.2018.

[2] Wannacry ransomware attack summary. `https://www.dataprotectionreport.com/2017/05/wannacry-ransomware-attack-summary/`. Accessed: 31.03.2018.

[3] S. Agrawal and J. Agrawal. Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, 60:708 – 713, 2015. Knowledge-Based and Intelligent Information Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings.

[4] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.

[5] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176, Secondquarter 2016.

[6] R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogrod, and M. A. Zissman. Evaluating intrusion detection systems without attacking your friends: The 1998 darpa intrusion detection evaluation. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 1999.

[7] J. E. Dickerson and J. A. Dickerson. Fuzzy network profiling for intrusion detection. In *Fuzzy Information Processing Society, 2000. NAFIPS. 19th International Conference of the North American*, pages 301–306. IEEE, 2000.

[8] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2):18–28, 2009.

[9] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[10] A. Karami and M. Guerrero-Zapata. A fuzzy anomaly detection system based on hybrid pso-kmeans algorithm in content-centric networks. *Neurocomputing*, 149:1253 – 1269, 2015.

[11] M. S. A. Khan. Rule based network intrusion detection using genetic algorithm. *International Journal of Computer Applications*, 18(8):26–29, 2011.

[12] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, Sep 1990.

[13] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, pages 333–342. Australian Computer Society, Inc., 2005.

[14] R. P. Lippmann and R. K. Cunningham. Improving intrusion detection performance using keyword selection and neural networks. *Computer networks*, 34(4):597–603, 2000.

[15] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 2, pages 1702–1707. IEEE, 2002.

[16] H. H. Pajouh, G. Dastghaibyfard, and S. Hashemi. Two-tier network anomaly detection model: a machine learning approach. *Journal of Intelligent Information Systems*, 48(1):61–74, 2017.

[17] C. Palagiri. Network-based intrusion detection using neural networks. *epartment of Computer Science Rensselaer Polytechnic Institute Troy, New York*, pages 12180–3590, 2002.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and

E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] M. Ramadas, S. Ostermann, and B. Tjaden. Detecting anomalous network traffic with self-organizing maps. In *International Workshop on Recent Advances in Intrusion Detection*, pages 36–54. Springer, 2003.

[20] scikit-learn developers. *A comparison of the clustering algorithms in scikit-learn, available http://scikit-learn.org/stable/modules/clustering.htmlclustering (accessed on 31.03.2018).*

[21] T. Shon and J. Moon. A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177(18):3799–3821, 2007.

[22] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000, 2009.

doi: 10.2313/NET-2018-11-1_06

Longitudinal study of user-space packet processing frameworks in academia

Längsschnittstudie von User-Space Paketverarbeitung Frameworks in der Wissenschaft

Maik Luu Bach
Betreuer: Paul Emmerich
Seminar Innovative Internet-Technologien und Mobilkommunikation SS2017
Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: maik.luu-bach@tum.de

KURZFASSUNG

Es existieren heutzutage viele Packet Processing Frameworks. Sie erfüllen größtensteils denselben Zweck und unterscheiden sich in diversen Eigenschaften. Gegenwärtig ist es möglich Fast Packet Processing mit kostengünstige gebrauchsübliche Hardware zu betreiben. Einige Packet Processing Frameworks stehen dabei heraus. Als Ansatz, um herausfinden welches das meist genutzte Framework ist, wurden wichtige Konferenzen der Datenkommunikation in Betracht genommen.
In dieser Arbeit wird dargelegt, warum gerade Frameworks, wie netmap oder DPDK auch in Facharbeiten eine große Bedeutung haben und wie sie in den letzten Jahren an Popularität gewonnen haben.

Schlüsselworte

packet processing, DPDK, netmap, Java, PDFBox

1. EINLEITUNG

Das Internet produziert immer neue Daten, Tag für Tag. Schätzungen von 2015 besagen, dass sich das Volumen der Daten bis 2025 verzehnfachen wird auf 163 Zettabyte. Das ist eine Wachstumsrate von circa 30% pro Jahr [1]. Diese Datenmenge muss weitertransportiert werden, zum Beispiel innerhalb eines Unternehmens. Bessere und zugleich auch alternative Technologien werden entwickelt, um neue Ansätze Datenpakete zu bearbeiten und zu transferieren. Commodity-Hardware steht im Vordergrund der neusten Forschungen, sodass jeder mit seiner vorhandenen Hardware und gegebener Software effizientes Packet Processing betreiben kann [2].

Frameworks, wie z.B. DPDK [3] und netmap [4] entstehen und Forscher aus aller Welt versuchen eine gute Mischung verschiedenster Frameworks zu finden. Laut Broy ([5]) ist eine Framework „ein halbfertiges Softwaresystem, das noch vervollständigt werden muss“. Zudem liefern die meisten Frameworks schon die Treiber für eine ausführbare Implementation.

In Abschnitt 2 werden die Grundlagen des User Space präsentiert. Im dritten Abschnitt der Arbeit werden die verschiedenen Arten und Zuordnung der Frameworks vorgestellt. Ein kurzer Vergleich wird im Abschnitt 4 dargelegt. Abschnitt 5 werden die Organisationen und dazugehörigen Konferenzen nacheinander genannt. Die Vorgehensweise der Beschaffung der Publikationen wird im sechsten Abschnitt beschrieben. Das anschließende Ergebnisse mit Zusammenfassung stehen am Ende dieser Seminararbeit.

2. USER SPACE

Ein Systemspeicher in Linux ist in 2 Kategorien einzuordnen, in Kernel Space und User Space. Der Kernel Space definiert den Speicher, wo der Betriebssystemkern seinen Code speichert und ausführt wie Treiber und Dateisystem [6].

Das User Space wiederum beschreibt den Teil des Systemspeichers, wo der Nutzer seine Prozesse exekutiert [7]. Darunter zählen nicht nur die Programme des Benutzers, sondern auch die enthaltenen Bibliotheken mit deren Methoden. Die Abbildung 1 veranschaulicht dies vereinfacht dar [8].

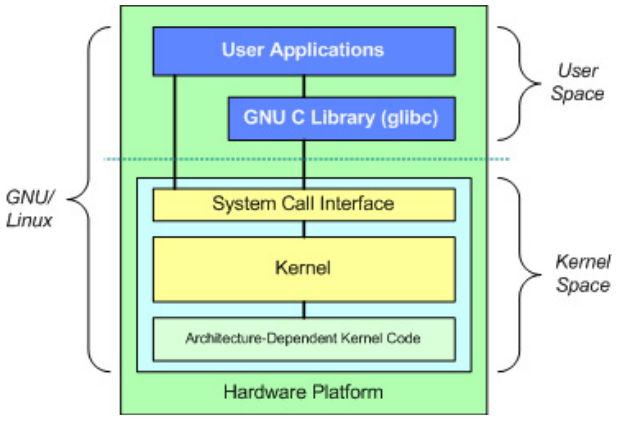


Abbildung 1: Grundlegende Architektur von GNU/-Linux

3. ARTEN VON PACKET PROCESSING FRAMEWORKS

Grundsätzlich kann man heutzutage die Packet Processing Frameworks in 2 Grundkonzepte aufteilen, welche die die CPU als zentrale Einheit zur Übertragung der Pakete nutzt,

doi: 10.2313/NET-2018-11-1_07

und andererseits den Ansatz die GPU in den Mittelpunkt zu stellen.

3.1 Techniken der Paketverarbeitung

Bevor wir zu den einzelnen Frameworks kommen, sind Erläuterungen notwendig zum besseren Verständnis der Thematik. Dazu beschreibe ich in den folgenden Absätzen die „Zero-Copy"-Technik und die Stapelverarbeitung (im Englischen „batch-processing") genannt.

3.2 Zero-Copy Technik

Ein Kernproblem der Paketverarbeitung ist unteranderem der begrenzte Speicherplatz. Zwischen Kernel und User Space wird so unnötig Speicherplatz verbraucht. Um dies zu vermeiden, existiert es die „Zero-Copy"-Technik. Diese Technik ermöglicht es, anhand einer Sammlung von Techniken, die Anzahl der Kopien zwischen Kernel und Geräten oder Kernel und User Space zu verringern. Dadurch ist eine Implementation in Umgebungen möglich, die wenig Arbeitsspeicher besitzen und so kostengünstige Lösung erlauben [12].

3.3 Stapelverarbeitung

Die Stapelverarbeitung ist eine Technik der Datenverarbeitung. Man kann sich einen Stapel vorstellen, wo alle Aufgaben in einem Stapel gesammelt werden und von oben nach unten abgearbeitet werden. Die Verarbeitung verläuft meistens automatisch und sequenziell [9].

3.4 CPU

Durch die Nutzung der Frameworks kann im Prinzip auch ein einfacher Bürocomputer als Router fungieren. Die Vorteile bestehen aus ihrer hohen Flexibilität, weil hauptsächlich nur Software benötigt wird, die schon in den meisten Fällen „open-source" ist. Der Gebrauch von standardardisierter Hardware ermöglicht eine preiswerte Einsetzung der Software-Router. Leider bietet diese Perspektive keine dauerhafte kommerzielle Lösung, da schlicht die Leistung dazu fehlt und spezielle Hardware-Router den Zweck besser erfüllen [13].

3.4.1 Data Plane Development Kit (DPDK)

DPDK ist ein open-source Linux Foundation Projekt mit dem Ziel eine Zusammenstellung von Treibern und Bibliotheken für das Fast Packet Processing zu liefern, unabhängig welche CPU-Architektur der Benutzer besitzt. Außerdem unterstützt DPDK zum Beispiel auch eine Multicore Framework, sodass eine maximal effiziente Weise mit Algorithmen geschaffen wird für das Packet Processing [3]. DPDK kann mit der Zero-Copy Technik angewendet werden und spart damit die Anzahl der Kopien im Arbeitsspeicher [12].

3.4.2 netmap

Die Framework netmap wird genutzt für High-Speed I/O und wird implementiert mit der mitgelieferten VALE Software als Single-Kernel Modul. Durch netmap können auf Netzwerkkarten, Host-Stacks, virtuelle Ports und „Netmap-Pipes" eine Bandbreite von über 14 Mpps erreicht werden ohne teure Hardware einzusetzen [4]. Netmap kann auch mit der Zero-Copy Technik angewendet werden [12].

3.4.3 PF_RING

PF_RING wird, wie die anderen beiden Frameworks zuvor, mit Zero-Copy Technik umgesetzt. Mithilfe von hunderten zusätzlichen Filtern und Paketanalyse versucht PF_RING die Paketerfassung zu optimieren. Es existieren verschiedene Typen von PF_RING zum Beispiel DNA (Direct NIC Access) und ZC (Zero-Copy) [17].

3.4.4 Snabb

L. Gorrie entwickelt seit 2012 die High-Performance Framework Snabb [10]. Diese Framework läuft auf Linux und x86-64 Architekturen. Zur Umsetzung des Systems nutzt man die Programmiersprache Lua und einen Compiler namens „LuaJIT", was es kompatibel macht mit C. Hierdurch besitzt das Framework einen „kernel-bypass" und kann außerhalb der User-Space, sogar im Kernal-Space ausgeführt werden [11].

3.4.5 PFQ

Ein Linux-basierte Framework ist PFQ, mit dem Gedanke wie DPDK, effizient Daten mit Skripten und Algorithmen zu beschleunigen. Die Implementierung von PFQ wird eigens mit der Programmiersprache PFQ-Lang umgesetzt, welche inspiriert von Haskell. Die Ergebnisse von PFQ sind sehr hardware-abhängig [16].

3.5 GPU

Der Vorteil gegenüber den CPU-betriebenen Frameworks ist die I/O Bandbreite. Der aktuelle PCI-Express 4.0 Standard kann auf einen 16-Lanes Steckplatz bis zu 16 GB/s übertragen. Schon in den nächsten Jahren wird ein Standard veröffentlicht, deren Datenrate über 32 GB/s beträgt und neue Möglichkeiten sich dadurch erschließen in der Thematik Performance. Nachteil dieses Ansatzes ist zum Teil die Umsetzung. Mainboards haben eine begrenzte Anzahl von PCI-Express-Slots und die Netzwerkkarte sollte dieselbe Bandbreite haben, um einen Bottleneck zu vermeiden [13].

3.5.1 PacketShader I/O

PacketShader unterscheidet sich vom Ansatz der anderen Frameworks, denn sie nutzt die GPU des Computer für das Packet Processing. Das zeigt das Grafikkarten nicht nur für Spiele und für das Kryptominen gut sind, sondern auch für Paketverarbeitung [13]. Die CPU wird nicht mehr als Bottleneck der Performance angesehen. Die Kosteneffizienz und das Potenzial einer parallelen Datenverarbeitung sind einige Beispiele der Vorteile von PSIO. Zur Zeit ist die einzige Begrenzung 40 GB/s, verantwortlich dafür ist die Einschränkung der Kapazitäten des Dualen I/O Hubs [15].

4. VERGLEICH

Jede Paketverarbeitungs-Framework hat seine Vor- und Nachteile. In diesem Abschnitt beschreibt die Abbildung 2 vereinfacht den Vergleich zwischen den Packet Processing Frameworks in diversen und prägnanten Eigenschaften.

5. BETRACHTETE KONFERENZEN

5.1 ACM

Mit den Gedanken „to advancing the art, science, engineering, and application of information technology" wurde bei einer Sitzung an der Columbia Universität im Jahre 1947 die Association for Computing Machinery gegründet. Sie ist

doi: 10.2313/NET-2018-11-1_07

	netmap	PF_RING	DPDK	pfq	Snabb	PSIO
Memory preallocated,reused	yes	yes	yes	yes	yes	yes
Parallel direct paths						yes
Memory mapping	yes	yes	yes	yes	yes	yes
Zero-copy	yes	yes	yes			
Batch processing	yes			yes		yes
GPU processing						yes

Abbildung 2: Vergleich der Frameworks (abgewandelt von [12])

damit die älteste wissenschaftliche Gesellschaft für die Informatik [18]. Unterteilt ist die Organisation in 34 Special Interest Groups [19].

5.1.1 SIGCOMM

Einer der Untergruppierungen der ACM ist die SIGCOMM (Special Interest Group on Data Communications) spezialisiert in Datenkommunikation und Computernetzwerke. Mit eine allgemein geringe Akzeptanzrate von 13% spricht für das Niveau der alljährlichen Konferenz [20].

5.2 IEEE

Das „Institute of Electrical and Electronics Engineers“ ist der weltweit größter Berufsverband, mit Mitgliederzahlen über 420.000, von Ingenieuren aus den technischen Bereich. Die IEEE publiziert jedes Jahr rund ein Drittel der Literatur aus diesem Segment [21]. Außerdem setzt sich die Zweckgemeinschaft für Standardisierungen ein.

5.2.1 ANCS

Die ANCS wird jedes Jahr mit Kooperation der ACM veranstaltet. Ziel des Forums ist die Forschung der Synergie von Algorithmen und Netzarchitekturen in der Theorie, sowie auch in der Praxis.

5.3 USENIX

Die USENIX ist eine Vereinigung von Entwicklern seit 1975. Sie nehmen sich als Ziel technische Innovationen zu fördern und verbreiten Arbeiten mit einem praktischen Hintergrund. Zudem bietet die Organisation ein neutrales Forum für Diskutionen [23].

5.3.1 ATC

Die USENIX Annual Technical Conference besprechen Forscher der Informatik über Themen wie Virtualisierung, Cloud-Computing und unter anderem auch Netzmanagement [24].

5.3.2 NSDI

Sponsiert wird die NSDI von USENIX mit Kooperation der ACM Gruppen SIGCOMM und SIGOPS. Hauptthemen sind Design und Implementation von Netwerksystemen [25].

5.3.3 OSDI

Alle 2 Jahre findet die OSDI Konferenz statt, wo die Hauptthematik Betriebssysteme stehen. Dabei betrachten die Spezialisten Theorie, sowie auch die Praxis [26].

6. VORGEHENSWEISE

In diesem Kapitel wird die Herangehensweise erläutert. Um die Beschaffung der Dokumente zu vereinfachen und zu beschleunigen, wurden viele Bibliotheken und kleine Hilftools wie Add-ons für diverse Browser. DownThemAll gibt einem die Möglichkeit Dateien von Websiten herunterzuladen [27]. Die Erweiterung an den Zeitpunkt der Arbeit nur für eine Firefox-Version vor Quantum verfügbar. Bei den USENIX Konferenzen war das kein schwieriges Unterfangen. Man öffnete alle relevanten Seiten mit den PDFs in Tabs und markierte die Seiten für den DownThemAll-Manager mit „All Tabs“. Als nächster Schritt wählte man beim Manager, das er nur die gefundenen PDF-Texte herunterlädt. Die Beschaffung der PDFs der SIGCOMM und der ANCS über den Bibliotheksproxy der Universität war begrenzt automatisiert möglich. Ein pro Nutzer festgelegtes, stündliches Downloadkontingent variiert je nach Verlag. Präventiv sperren Verlage den „e-Access“ für paar Stunden.

Nach der Beschaffung der Dokumente ist die Konvertierung der PDFs in TXT-Dateien wichtig um die weitere Bearbeitung zu gewährleisten. Da bietet PDFBox, eine open-source Java-Bibliothek, die notwendigen Methoden dazu bereit [28]. Unter der Angabe des Ordners konvertiert mittels for-each-Schleife jede PDF in eine .txt-Datei [29].

```
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import org.apache.pdfbox.cos.COSDocument;
import org.apache.pdfbox.io.RandomAccessFile;
import org.apache.pdfbox.pdfparser.PDFParser;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.text.PDFTextStripper;

public class javaPDF2txt {

  public static void main(String[] args) {
    String pathOri = "Pfad des Ordners";
    File[] files = new File(pathOri).listFiles();
    for (File file : files) {
      if (file.isFile()) {
        System.out.println(file.getName());
        konvertieren(pathOri + "/" + file.getName(),
            file.getName());
      }
    }
  }

  public static void konvertieren(String path, String
      DateiName) {
    PDFTextStripper pdfStripper;
    PDDocument pdDoc;
    COSDocument cosDoc;
    File file = new File(path);
    try {
      RandomAccessFile randomAccessFile = new
          RandomAccessFile(file, "r");
      PDFParser parser = new PDFParser(randomAccessFile);
      parser.parse();
      cosDoc = parser.getDocument();
      pdfStripper = new PDFTextStripper();
      pdDoc = new PDDocument(cosDoc);
      String parsedText = pdfStripper.getText(pdDoc);
      try (PrintWriter out = new PrintWriter(DateiName +
          ".txt")) {
        // an Dateiname wird ein .txt angehaengt
        out.println(parsedText);
        pdDoc.close();
```


doi: 10.2313/NET-2018-11-1_07

```
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Da die Texte von Konferenzen stammen, die anderen Thematiken ansprechen und besprechen in den Veröffentlichungen sind viele Schriften nicht von Relevanz. Um dies herauszufinden, welche Dokumente von Bedeutung sind, nutzt der StringTokenizer der Java-Bibliothek. Diese Klasse bricht einen String in Tokens auseinander, dabei unterscheidet er nicht zum Beispiel zwischen Variablen und Zahlen. Dazu speichert man die Textdatei in einen String mithilfe eines Streams [31] und der speichereffizienten „Files.lines“ [32]. Mittels der Methoden „nextToken()“ und „contains()“ testet man, ob der Text das Keyword enthält und somit eine Relevanz zeigt.

```
public static void leser(String path) {
  String contents = "";
    try {
      contents = Files.lines(Paths.get(path)).
      collect(Collectors.joining());
      // Txt in einen String rein
    } catch (IOException e) {
      e.printStackTrace();
    }
  StringTokenizer st = new StringTokenizer(contents);
    while (st.hasMoreTokens()) {
      if (st.nextToken().contains("keyword")) {
        relevant = true;
      }
    }
}
```

Bei knapp 1605 Facharbeiten stellt sich dann heraus, dass in 108 Dokumente mindestens einer der vorher gennanten Frameworks erwähnt. Somit sind circa 7% für diese Ausarbeitung von Bedeutung. Damit der Überblick noch vorhanden bleibt, werden alle relevanten Texte in einen separaten Ordner kopiert, mittels der Methode „FileUtils.copyFile(source, destination)“.

7. ERGEBNISSE

Durch das gezielte Lesen von den Sätzen mit den Keywords war eine manuelle Zuordnung möglich, wie sich die Autoren mit den Frameworks auseinander gesetzt haben. Dabei wird unterschieden, ob der Verfasser die Framework erwähnt hat, oder sie implementiert hat und explizit im Text genannt wird. An diesem Punkt wird nochmal unterteilt, ob bei der Umsetzung auch das Backend miteinbezogen wird. Die Abbildung 3 zeigt die Verteilung der Publikationen, in wie vielen und welche Framework dabei genutzt wurde. Dabei stellt sich heraus, dass DPDK, netmap und PF_RING heraussstehen. Seit 2012, nach Veröffentlichung der Arbeit von Rizzo über netmap und im Jahre 2013, dass DPDK von einer opensource Gemeinschaft weiterentwickelt wird, gewannen beide Frameworks zugleich Anerkennung und Beliebtheit. Zudem fällt auf, dass netmap sehr oft genannt wird. Viele Artikel haben aus den Jahren 2015 und 2016 sich netmap als Inspiration genommen und lediglich nur in den Referenzen angeführt. Es existieren wenige Texte in den Jahren nach 2012,

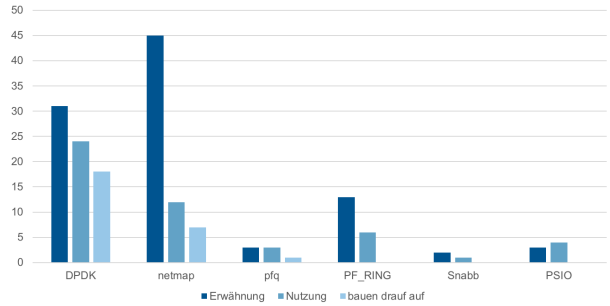


Abbildung 3: Auswertung 2010-2017

deren Inhalt nur eine Framework abhandelt. Wenn Beispiele von Paketverbeitungs-Frameworks gesucht werden, werden meist in den Publikationen DPDK und netmap genannt.

Nicht nur dieses Ergebnis ist interessant, sondern auch die Verteilung nach den Konferenzen nach den Frameworks, wie in Abbildung 4 zu sehen.

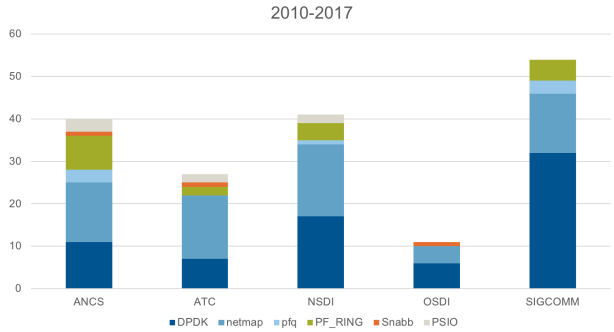


Abbildung 4: Auswertung nach Konferenz

Die SIGCOMM, die eine geringe Annahmequote bei Publikationen hat, neigen eher zu Intel DPDK. Die anderen zeigen entweder ein gleichwertiges Interesse zu DPDK und netmap, oder bevorzugen eher netmap in den Arbeiten. Zudem zeichnet sich heraus, dass die CPU-lastigen Frameworks überwiegen.

8. ZUSAMMENFASSUNG

Die Frameworks DPDK und netmap erweisen sich als gleichwertig populär heraus. Aber gerade in den letzten beiden Jahren schwenkt das Interesse für Implementationen von netmap mehr zu DPDK hin. Aber die Entwicklung Packet Processing Frameworks mit Hilfe von GPUs sollte man nicht ignorieren, weil sie neue Perspektiven geben in Themen wie Bandbreite und vermutlich auch später Performance. Folglich kann sich der Trend nochmal ändern und die zukünftigen Konferenzen dies zeigen.

9. LITERATUR

[1] *Globale Datenmenge wächst bis 2025 auf 163 Zettabyte | t3n*, `https://t3n.de/news/globale-datenmenge-2025-811914`, zuletzt besucht am 1. April 2018

[2] Dominik Schöffmann: *Comparing PFQ: A High-Speed Packet IO Framework*,

doi: 10.2313/NET-2018-11-1_07

https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2016-07-1/NET-2016-07-1_09.pdf, zuletzt besucht am 1. April 2018

[3] *DPDK*, http://dpdk.org/, zuletzt besucht am 1. April 2018

[4] *netmap*, http://info.iet.unipi.it/~luigi/netmap/, zuletzt besucht am 1. April 2018

[5] *05_SW_Architekturen*, https://wwwbroy.in.tum.de/lehre/vorlesungen/mbe/SS07/vorlfolien/05_SW_Architekturen.pdf, zuletzt besucht am 1. April 2018

[6] *Kernal Space Definiton*, http://www.linfo.org/kernel_space.html, zuletzt besucht am 1. April 2018

[7] *User Space Definition*, http://www.linfo.org/user_space.html, zuletzt besucht am 1. April 2018

[8] *Anatomy of the Linux kernel*, https://www.ibm.com/developerworks/linux/library/l-linux-kernel/, zuletzt besucht am 1. April 2018

[9] *What is Batch Processing? - Definition from Techopedia*, https://www.techopedia.com/definition/5417/batch-processing, zuletzt besucht am 1. Mai 2018

[10] *Diving into Snabb*, https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2016-09-1.pdf, zuletzt besucht am 1. April 2018, pages 31-38

[11] *snabb*, https://github.com/snabbco/snabb, zuletzt besucht am 1. April 2018

[12] *A Survey of Trends in Fast Packet Processing*, https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2014-08-1.pdf, zuletzt besucht am 1. April 2018, pages 41-48

[13] *Using GPUs for Packet-processing*, https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2014-08-1.pdf, zuletzt besucht am 1. April 2018, pages 33-40

[14] Rizzo, L.: *Netmap: a novel framework for fast packet I/O*, Proceedings of the 2012 USENIX conference on Annual Technical Conference, USENIX ATC'12, Berkeley, CA, 2012

[15] *PacketShader - GPU-accelerated Software Router*, https://shader.kaist.edu/packetshader/, zuletzt besucht am 2. April 2018

[16] *PFQ I/O*, https://www.pfq.io/, zuletzt besucht am 1. April 2018

[17] *PF_RING*, https://www.ntop.org/products/packet-capture/pf_ring/, zuletzt besucht am 2. April 2018

[18] *ACM History*, https://www.acm.org/about-acm/acm-history, zuletzt besucht am 31. März 2018

[19] *ACM Resources*, https://cacm.acm.org/acm-resources, zuletzt besucht am 31. März 2018

[20] *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, https://dl-acm-org.eaccess.ub.tum.de/citation.cfm?id=3098822, zuletzt besucht am 31. März 2018

[21] *IEEE - IEEE at a Glance*, https://www.ieee.org/about/today/at_a_glance.html, zuletzt besucht am 31. März 2018

[22] *ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, https://sites.google.com/a/ancsconf.org/www/, zuletzt besucht am 31. März 2018

[23] *About USENIX | USENIX*, https://www.usenix.org/about, zuletzt besucht am 31. März 2018

[24] *USENIX ATC '18 | USENIX*, https://www.usenix.org/conference/atc18#about, zuletzt besucht am 1. April 2018

[25] *NSDI '18 | USENIX*, https://www.usenix.org/conference/nsdi18, zuletzt besucht am 1. April 2018

[26] *OSDI '18 | USENIX*, https://www.usenix.org/conference/osdi18, zuletzt besucht am 1. April 2018

[27] *DownThemAll (or just dTa) is a powerful yet easy-to-use Mozilla Firefox extension that adds new advanced download capabilities to your browser*, https://www.downthemall.net, zuletzt besucht am 31. März 2018

[28] *Apache PDFBox | A Java PDF Library*, https://pdfbox.apache.org/, zuletzt besucht am 1. April 2018

[29] *java - How to extract text from a PDF file with Apache PDFBox - Stack Overflow*, https://stackoverflow.com/questions/23813727/how-to-extract-text-from-a-pdf-file-with-apache-pdfbox, zuletzt besucht am 1. April 2018

[30] *StringTokenizer (Java Platform SE 8)*, https://docs.oracle.com/javase/8/docs/api/index.html?java/util/StringTokenizer.html, zuletzt besucht am 1. April 2018

[31] *Java Streams - Collectors joining() example*, http://www.java2s.com/Tutorials/Java/java.util.stream/Collectors/Collectors.joining_.htm, zuletzt besucht am 1. April 2018

[32] *Java 8 API by Example: Strings, Numbers, Math and Files - Benjamin Winterberg*, http://winterbe.com/posts/2015/03/25/java8-examples-string-number-math-files, zuletzt besucht am 1. April 2018

GDPR - Required Changes By Website Operators And Technical Ways To Check For Compliance

Maximilian Muth
Advisor: Quirin Scheitle
Seminar Future Internet SS2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: muthm@in.tum.de

ABSTRACT

Tracking technology on websites has become more advanced and easy to deploy. While it allows website operators to gain useful insights about their site visitors and even additional business models by using the collected data, it reduces the privacy of the consumers.
In 2011, the European Union tried to improve this situation by introducing a directive which required website owners to ask the visitor for consent to store cookies in the web browser and to give site visitors the right to refuse the use of cookies. The directive's goal was to increase awareness about which data is collected and to improve the consumers online privacy. In 2016 the European Union approved the General Data Protection Regulation (GDPR) which aims at protecting online privacy of consumers in multiple ways.
In this paper we present the most important changes by GDPR for website owners and highlight what changed since the "cookie law" directive from 2011. We will also take a look at a website and analyze whether it is already GDPR compliant or what they would need to change to be compliant when the GDPR is implemented on 25th May 2018.
One of our findings from that analysis is, that although a small company already put in effort in order to be compliant, they apparently still did some mistakes.
As a final reminder, this paper is no legal advise but an opinion from an engineering point of view.

Keywords

GDPR; General Data Protection Regulation; European Union Data Protection; GDPR Compliance

1. INTRODUCTION

The protection of natural persons by protecting their personal data is a fundamental right according to Article 8(1) of the Charter of Fundamental Rights of the European Union [7]. One of the recitals in the preamble of the General Data Protection Regulation (GDPR)[1] is the following, which clearly phrases the privacy problems with today's online tracking technologies:

> *Natural persons may be associated with online identifiers provided by their devices, applications, tools and protocols, such as Internet Protocol addresses, cookie identifiers or other identifiers such as radio frequency identification tags.*
> *This may leave traces which, in particular when combined with unique identifiers and other information received by the servers, may be used to create profiles of the natural persons and identify them.*[7]

A simple JavaScript snippet by a tracking or analytics service provider is enough to track website visitors over a period of time, get detailed insights about the users interaction or even recordings of the screen the user sees with precise mouse movement tracking.
By using tracking cookies, it is even possible for providers to track a certain user across multiple websites. Another issue is that advertisement and tracking providers share identifiers and data between each other which allows them to create more detailed user profiles.

This results in increased privacy concerns in many cases and often the site visitors are not even aware of the fact that they are tracked and profiled in such detailed ways by the website operators.
Sometimes it might even not be on purpose that providers implement tracking possibilities in their products, like Apple did with their Apple Push Notification Service (APNs) which enabled user tracking based on TLS Client Certificate Authentication [13].

The European Union tried to improve this situation by introducing a EU directive in 2011[2] which required website owners to ask the visitor for consent to store cookies in the web browser and to give site visitors the right to refuse the use of cookies. The directive's goal was to increase awareness about which data is collected, to reduce the amount of stored data to the necessary needed data and to improve the consumers online privacy.
To enhance the regulations on data protection, the parliament of the European Union approved the GDPR on 14 April 2016 after four years of debate. The General Data Protection Regulation is a replacement for the Data Protection Directive 95/46/EC which has been in place since 1995.

The GDPR will be implemented - and thus enforced - by 25

[1] Official identifier: REGULATION (EU) 2016/679

[2] `http://ec.europa.eu/ipg/basics/legal/cookies/index_en.htm`

doi: 10.2313/NET-2018-11-1_08

May 2018 and there will be penalties on organizations which do not comply to it by that date.

In the first part of this paper we will discuss the new aspects and rule sets of the GDPR with a focus on the technical implementation.

In the second part we will summarize what website operators need to change in order to be compliant to the GDPR and what possibilities exist for consumers to check whether websites are compliant.

2. OVERVIEW

The GDPR introduces some new key concepts which we discuss in the following chapter. We also show under which conditions the GDPR applies.

2.1 New Topics And Regulations

In this section we pick a couple of interesting key changes or strengthened topics in the GDPR. We're focusing on topics which are related to the rights of consumers towards website operators.

2.1.1 Lawfulness of Processing

One of the basic concepts of the GDPR is that the processing of personal data is prohibited unless stated otherwise by the law. The most important reasons why a processing might be allowed are:

1. Processing is necessary for the provided service or contract (Art. 6.1.b GDPR[3])
2. *"Processing is necessary for the purposes of the legitimate interests pursued by the controller or by a third party, except where such interests are overridden by the interests or fundamental rights and freedoms of the data subject"* (Art. 6.1.f).
3. The user provided consent (Art. 6.1.a)

Details about the consent are discussed in the upcoming section. The first reason however, is interesting since it seems to not really be clear when a site operator's legitimate interests (German: *Berechtigte Interessen)* legitimate him to process data without the user's consent.
It looks like for example online marketing and analysis are such legitimate interests [6]. At the first impression it seems like *legitimate interests* might be a loophole for providers to process data without the need of the user's consent. However, in recital 47 GDPR it says the following:

> *"At any rate the existence of a legitimate interest would need careful assessment including* ***whether a data subject can reasonably expect*** *at the time and in the context of the collection of the personal data* ***that processing for that purpose may take place****".*

[3]Used law text citation style: *Art. 7.2.a* refers to point (a) of the second paragraph of Article 7. If not stated otherwise, the cited lawtext is the GDPR

So while the juristical community discusses what exactly legitimate interests are in certain cases [6], in our opinion it looks like the GDPR tries to be strict regarding the term of legitimate interests.
At the current point in time the discussion seems to be still ongoing and we should await first judgments in order to better understand this topic.

2.1.2 Consent

Some of the most interesting changes are related to the consent of the *data subject* (the site visitor) about how the subject's data is processed and stored.
Some key elements of GDPR related to consent are that the request for consent must be obviously presented as such a request and the phrasing must be easy to understand and no legalese (Art. 7.2). It also must be clear why the requested data is needed and what the site operator is using it for (Art. 7.). After consent was given, it shall be as easy to withdraw the consent as it is to give (Art. 7.3).
This is interesting because some websites seem to hide the reasons on why they ask for certain personal data in large privacy policy documents. However, they now need to make sure to tell the user in an understandable way what the personal information is used for at the moment the user gives its consent.
Furthermore the user needs to give consent for storing personal data before the data is actually collected and stored. This is important for the usage of third-party tools like analytic or advertisement frameworks which by default might set cookies and collect personal data like the used browser, resolution, IP address etc. before the user gave his consent. Processing of personal data without the consent of the user or other reasons is not legal (Art. 6.1) and it is the website operator's duty to ask for that consent before collecting personal data.
Site operators need to make sure that they can prove to supervisory authorities that a given user gave consent for the processing of his personal data (Art. 6.1).

2.1.3 Information and Access to Personal Data

Users have a right to request information about whether specific information are stored and processed by a site operator (Art. 15.1.e), as well as for what purpose the information is used and whether it is used for automated decision-making programs or profiling (Art. 15.1.a,e).
But most of all, the user has a right to request the personal information that the operator stores about him. The operator must hand out all stored data in an electronical format, without charging a fee. Website operators seem not to need to provide an automated process for sending out the requested data, but can handle those requests for data by hand, which might be important for small website operators.

2.1.4 Right to Erasure

Right to erasure - also called "Right to be Forgotten" - enables consumers to request the deletion of all personal data stored and generated by the website operator. One of multiple conditions must be met in order to be able to lawfully request the removal from a *controller* (like a website operator), most importantly: the data is not used anymore for the purpose for which it was collected (Art. 17.1.a) or the user has withdrawn his consent (Art. 17.1.b).

doi: 10.2313/NET-2018-11-1_08

So whenever a user wishes to have his data removed he can withdraw his consent and request the deletion.
Another point which enhances the user's privacy is that the controller needs to forward the request for erasure to other parties to whom the personal data was disclosed (Art. 19.). However, the law text also states that this is only needed *unless this proves impossible or involves disproportionate effort* (Art. 19.), so it looks like controllers might argue to not do this due to high effort implementing such a solution.

2.1.5 Right to Object

According to Art. 21.1.a a user can restrict what processing his personal data is used for. This is especially relevant in the case of automated profiling. If the personal data is used for direct marketing purposes, the user can object that usage at any given time (Art. 21.2).
In reality it is questionable how well this works - it is possible that site operators simply stop providing their service to the user in case he requests to restrict the processing of his personal data. As a result, users might not request restriction because they want to use the given service.
At this point in time, we didn't find any resources to validate this claim.

2.1.6 Data Portability

The right for data portability extends the right for access to personal data (Art. 15.) in the way that the user is allowed to take the personal data and give it to another controller / website operator. The first operator is not allowed to hinder the usage of the data by another controller in any means (Art. 20.1).
This idea seems great since it aims at impeding the vendor lock-in strategy [8] which tries to make users stay by making moving to another provider complicated.
The law allows users to migrate between different service providers and even makes sure that the exported data is *"in a structured, commonly used and machine-readable format"* (Art. 20.1).
Nonetheless it is questionable how well the exported data can be used for migrating to another service, since the domains where a *"commonly used format"* like vCard for contact information[4] or iCal for calendar definitions exists, is rather limited in our opinion. The use of common (but non domain-specific) data formats like JSON, XML or CSV is highly beneficial nevertheless.
Phrased differently: What is a *commonly used format* of the data in a social network? Still, other service providers or open-source communities might come up with automated import routines based on this law.

2.1.7 Breach Notification

If a personal data breach occurs, the controller must notify the supervisory authority of the breach within 72 hours (Art. 33.1). A supervisory authority is determined by the member states themselves. In Germany the federal states are responsible in providing such an authority and in Bavaria the supervisory authority is *Bayerisches Landesamt für Datenschutzaufsicht* (BayLDA)[5].
In Germany, data breaches already must be reported according to paragraph 42a BDSG[6] in case the personal data is highly sensitive (like banking or health information) and as a second condition that there is a high risk of big impairments for the subject [1]. So while in Germany a law already required notifications to supervisory authorities, it was pretty loose and many incidents would not require a notification.
This might be different in the future: With the GDPR, every personal data breach needs to be reported to the supervisory authority, which is an improvement of the previous regulation. However, it is not required to notify the affected users unless it *is likely to result in a high risk to the rights and freedoms of natural persons* (Art. 34.1).
In such a case the controller needs to notify the user without any delay and also provide details about the breach like categories and subjects of the affected data as well as consequences (Art. 34.2). The notification must be plain text and easy to understand.
It is important to note, that the supervisory authority has the ability to require the controller to notify the data subject if the controller did not do so by himself.
An interesting exception where a notification is not needed is when the personal data is encrypted (Art. 34.3.f). As a result, the data subject might not be informed that encrypted personal data was leaked. We could argue that this is an issue since the attacker might not be able to break the encryption of the data on the date of the breach. But maybe in the future it is feasible.

At the time of writing, information are appearing that claim the company Cambridge Analytica collected personal information of multiple million Facebook users in order to target US electors with highly purposeful and targeted Facebook advertisements before the presidential election in 2016.[11] Facebook is being criticized as well, for example because they did not notify the affected users that a third party illegally collected their personal data [11]. The discussion is still going on and it is too early to draw conclusions about it in this paper.

This current event stresses the importance of clearer and stricter rules and supervisory related to user's personal data.

2.2 Area Of Application

In the following section we discuss under which conditions the GDPR applies, especially under which conditions a website operator needs to make sure to comply and in which cases consumers have the rights GDPR gives them like the above mentioned strictly regulated request for consent or access to personal information.

When analyzing whether a given law applies in a certain context we need to take a look at two areas of application: material scope (German: *Sachlicher Anwendungsbereich*) and territorial scope (German: *Räumlicher Anwendungsbereich*).

2.2.1 Material Scope

Material scope is given when (semi-)automated processing of personal information occurs or when the processing is done manually but the data is part of a *filing system* as described

[4] `https://tools.ietf.org/html/rfc6350`
[5] `https://www.lda.bayern.de`
[6] Bundesdatenschutzgesetz

doi: 10.2313/NET-2018-11-1_08

in Art. 2.1.
Furthermore material scope is not fullfilled in some defined cases, most importantly it does not apply when a natural subject processes his own personal information (Art. 2.2.c). According to Art. 2.2.a the GDPR also does not apply when the processing of personal data occurs *"in the course of an activity which falls outside the scope of Union law"*. We're not quite sure how it is determined whether an activity is outside the Union law's scope or not, but we could imagine it is related to the territorial scope we discuss in 2.2.2.

We also see, that the material scope is only fulfilled if the processed data is *personal* (German: *personenbezogen*). The European Union provides a definition of personal data in Art. 4. as follows:

> *"'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;"*

This definition is rather coarse and does not provide any examples. However, it also states that personal data must not only be a single data item like a name, but that the combination of non-personal information might also be *personal data*.

Let's think about the power of combining data with a small example: Person P provides the following information to website operator A: City, gender, age group. To another service provider B P gives information about his profession, the range of his income, as well as his first name. While it is hard to track P down with each of those information separately, it is a lot more feasible by combining the two data sets. Of course, for combining, a common identifier is needed, which might be *P's* IP address, or a third-party cookie both websites store and read.
Combining data sets from multiple websites is helpful for creating user-profiles which allows operators to have more information available that can be used for eg. making online advertisement more efficient. So it is good to see the definition also covering the combination of multiple data sets.

2.2.2 Territorial Scope

The law only applies if the material scope and the territorial scope are both fulfilled. According to Art. 3.1 territorial scope is fulfilled if the context of the establishment, controller or processor is in the European Union. It is important to note that it is not relevant whether the processing itself takes place in the Union.
The regulation also applies if the controller or processor is not located in the Union, but the data subject (eg. site user) is, under the conditions that the goods or services are provided in the EU (Art. 3.2.a).
To compare this to the previous situation, territorial scope did not change for organizations located in the EU. Nonetheless, GDPR broadens the territorial scope so that EU privacy rules also apply for organizations outisde the Union if they operate in the Union (like selling their goods or services) - a situation which is referred to as Extraterritoriality[12].
As a result, non-EU organizations might be required to comply to the GDPR for their customer segment within the EU.

Besides privately held companies, it is also interesting to take a look at foreign governments. Some countries are implementing their own privacy endangering laws, like for example the recently discussed *CLOUD Act* in the US[2]. This bill would allow US law enforcement to force companies to hand out personal data about US citizens without the need of search warrants nevertheless whether the data is stored in the US or outside.
However, GDPR explains in Art. 48. that such a disclosure of data is only possible if there are international agreements which explicitly allow it. As a result, it might be possible that US companies must not need hand out data about US citizens if it is stored in the EU. Note, that at the current time, this is speculative and we'd need to wait for further juridical discussions or court judgments.

3. NEEDED CHANGES FOR OPERATORS AND CONTROL POSSIBILITIES

In the previous sections we outlined the most important changes of the GDPR in a theoretical context. Now, we want to focus on website operators and take a deeper look at what they need to change in order to be compliant with the GDPR.
Furthermore, we want to investigate for which of those changes a user can control whether the site operator is compliant or no and which of those changes could be possibly checked automatically.

3.1 Needed Changes

3.1.1 Google Analytics

Google Analytics[7] (GA) or Matomo (previously Piwik)[8] are wide-spread third-party integrations for tracking and analyzing user behavior on websites.
These frameworks usually store a wide range of metrics, like IP addresses, operating system, browser, browser settings, language, viewport resolution, HTTP header information like referrers and much more. Furthermore each page visit as well as the visit duration and interactions within the page are stored.
The question is, what of these information are personal information so the GDPR applies (see. 2.2). According to a judgment in 2016 of the European Court of Justice, IP addresses are personally identifiable information[9], so the site operator needs to ask for consent before allowing his analysis framework to collect and store the IP address.
In this section we will focus on Google Analytics because it has the largest market share. Anyways, most points are valid for other tools like Matomo as well.
There are two possible solutions for website operators to deal with IP addresses in GA:

[7] `https://analytics.google.com/`
[8] `https://matomo.org/`
[9] Judgment in legal matter C-582/14, European Court of Justice

doi: 10.2313/NET-2018-11-1_08

1. Anonymize IP address, so it is not personally identifiable anymore [6]
2. Ask for consent before initializing GA

While the second approach is a bit more difficult to implement (the site needs to remember the state of the consent and the operator cannot just simply put the GA JavaScript snippet in the website), the first approach can be easily achieved. However, both approaches require the operator to take action because GA does not anonymize IP addresses by default.
If IP anonymization is enabled, GA will set the last 8 bit of the IP address to zero (or the last 80 bit for IPv6 addresses).

There are four ways to achieve IP anonymization in GA:[9]

- Set the *anonymizeIp* option when initializing GA via JavaScript: `ga('set', 'anonymizeIp', true)`. This globally enables IP anonymization for every event GA tracks on the site.
- Set the *anonymizeIp* flag for single events: `ga('send', 'pageview', 'anonymizeIp': true);`
- Set the *anonymizeIp* flag via Google Tag Manager (*gtag.js*)
- Set the *anonymizeIp* flag via the web GUI of Google Tag Manager

Although

3.1.2 Webserver Logs

As argued in the previous section, IP addresses are personal information. It is questionable whether monitoring of the webserver access log is a legitimate reason to store the IP address without the user's consent.
The request for consent is mostly on the client-site after the webpage has loaded, however, the webserver already logged the request at that point of time. As a result it is strongly recommended to anonymize the IP addresses in the webserver's log.
Furthermore, depending on the website, personal information might also be part of the URL path and an operator should anonymize this as well if he didn't clearly got the consent of the user.

Since writing log filters and anonymizing scripts is time consuming, it might be a good alternative to disable webserver access logs completely if the operator is not actively monitoring and analyzing them.

3.1.3 Newsletter

Many websites provide email newsletters to their users. Users need to sign up for the mailing list in order to receive the website operator's content per mail.
As it has already been before the GDPR, sending marketing mails to users without their consent is not legal. GDPR strengthens the way controllers need to verify that the user gave his consent, so according to Art. 7.1 the controller must be able to demonstrate that a given user gave consent to receive marketing emails.
It is noteworthy to repeat that this request for consent must have been clear and obvious to the user. Pre-ticked checkboxes for example are not legal (Recital 32, GDPR), the same goes for automatic newsletter signups for example after an order in an online shop.
Also, it is not allowed to require a sign-up for the newsletter in order to receive a service like the download of an eBook (Art. 7.4). This topic seems to be heavily discussed within the juristic community since it partly renders freemium models[10] useless.[5]

Since the controller must be able to demonstrate the consent, a *soft opt-in* is not enough. Soft opt-in means that a user gave his consent to receive the newsletter on the website itself by ticking a checkbox. Since anyone can do so with any email address, it is required to do an *double opt-in* which basically is a mail asking for the consent after the requested to be put on the newsletter list on the webpage.[5]

3.1.4 Social Media Integrations, Comment Systems, Gravatar, Ad Networks

Many website operators integrate some third-party tools like social media integrations (Facebook, Twitter, Xing, Linkedin, ...), comment systems like Disqus, avatar systems like Gravatar or advertisment networks like AdSense or Doubleclick. The challenge is, that many of those integrations send the user's personal data like IP addresses to the service provider [10].

For each of those integrated third-party tools, the website operator is in charge of requesting consent for the forwarding of personal data to the third-party provider. So the operator should mention all those services and state what the data is used for when requesting consent of the user. If the user did not yet provide consent, those third-party integrations should not be initialized.

3.1.5 Privacy Policy

The privacy policy should clearly state which data the website operator collects and stores, as well as which data is provided to third-party integrations including the reasons.
If the operator is relying on the user's consent to do so, the request for consent should be repeated in the privacy policy.
For some third-party integrations like Goolge Analytics detailed information need to be provided, like for example a way to opt-out of the data collection (remember, we argued that it is GDPR compliant to use GA with an anonymized IP without asking for consent, so the operator needs to make sure to provide a way to allow the user to prevent the pseudoanonymized data collection of GA).
In the case of GA there are two ways to achieve an opt-out, one of them is by setting a cookie and the other option is to suggest the user to install a browser extension provided by Google[11][6].

3.2 Technical Ways To Check For Compliance

One of the problems with many privacy related laws and regulations is that it is often hard to check for the users

[10] `https://en.wikipedia.org/wiki/Freemium`
[11] `https://chrome.google.com/webstore/detail/google-analytics-opt-out/fllaojicojecljbmefodhfapmkghcbnh`

doi: 10.2313/NET-2018-11-1_08

whether controllers really behave accordingly. This is especially the case for "average" or non-technically skilled users. In this chapter we discuss which of the above mentioned measures a website operator should take can be controlled and checked by users - both, in a manual and maybe even automated way. By finding possibilities to automatically check for conformity and making the test results publicly available, it is imaginable to help users to preserve their privacy.

3.2.1 IP Anonymization Of Google Analytics

One of the most important measures a user can check is, whether Google Analytics anonymizes IP addresses. Because if not - the user would have needed to provide consent. In 3.1.1 we investigated the different ways how website operators can enable the IP anonymization in GA.

A user can manually check, whether IP address anonymization is enabled or not, by inspecting the source-code of the website and searching for the JavaScript snippet which sets the anonymization flag (`ga('set', 'anonymizeIp', true);`). If this flag is present, the page is probably anonymizing the user's IP address.

If the flag is not set in the body of the page, or the user want's to make sure that the flag is not overridden by single events[12], he can check the network communication between his browser and Google Analytics servers. This can be easily done with the Developer Tools in Google Chrome or Firefox.
Each event which is tracked in GA will perform a *HTTP GET* or *POST* request to *https://www.google-analytics.com/r/collect*[13] with additional data being passed as URL parameters or in the HTTP POST body. If IP anonymization is enabled (either by setting the global flag or by configuring it on the given event), the request contains the parameter `aip=1`, for example: *HTTP GET https://www.google-analytics.com/r/collect?v=1aip=1&....* If the *aip* parameter is not present, the given event will be stored without an anonymized IP address.[9]

The monitoring of the outgoing AJAX requests might be even checked automatically in order to detect whether a given page is not compliant to the GDPR. One solution might be a browser extension[4] which analyzes the requests to the servers of GA and alerts the user in case the IP address is not anonymized. Such a browser extension could also store this information for later research.
On a larger scale it should be possible to run a web crawler (with JavaScript enabled) and test this on an arbitrary number of pages and log the results to a database for further analysis.

3.2.2 Newsletter Double-Opt-In

As elaborated in 3.1.3 a double-opt-in is required for newsletter sign-ups. This means, if a user subscribes to an e-mail newsletter on a website, he must receive a mail asking for confirmation before the operator is allowed to send him mails via the newsletter.

Obviously, this can be easily checked manually. Furthermore, it would be possible to implement automatic checks for the double-opt-in by using a web-crawler which performs sign-up events on newsletters and a program which is connected to the same e-mail account in order to check whether a confirmation mail has arrived.
There are some challenges, one would face when implementing this system. For example it is not given that the domain of the website where the crawler signed up for the newsletter is the domain-name of the mail server from which the confirmation mail arrives. A possible solution to this might be using a mail server which supports mail aliases and sign-up with a uniquely identifiable e-mail address for each website.

3.2.3 Privacy Policy

If a user wants to check the privacy policy, the first check might be whether the policy says which data is collected and for what reason. Furthermore, he needs to make sure that third-party integrations like Google Analytics or Facebook are mentioned if they are embeded in the website (the user might take a look in the source-code of the given page or use a browser extension like Wappalyzer[14] to detect those third-party integrations).
For integrations like GA an opt-out possibility must be proposed.

Besides, it might be hard for the user to evaluate whether the privacy policy conforms to all given laws, since it is not trivial to for a layman.

A check which can be automatically done is to search for keywords like *Google Analytics* or *Matomo* within the privacy policy if the crawler detects that such third party integrations are used (by either looking for corresponding JavaScript objects or cookies).

3.2.4 Breach Notification

In order to check whether the controller notifies a user about a data breach, the user needs to know that a breach occurred. This, however, is a circular condition, so it is probably non-trivial to check whether operators notify accordingly.
An approach which might be working was introduced by Joe DeBlasio et al. with a system called Tripwire[3] which is based on password-reuse. The idea is to use unique e-mail addresses to register on websites and to use the same password for the website and the email account itself. You then monitor over a longer period of time whether a successful login to the email account occurs. If such a login happens, it is interpreted as compromise of the the website for which the email address was used because the attacker used the password of the website account to access the corresponding mail account (the attacker checked whether the victim re-used his password).

While this might work in our situation, it is obviously not a perfect test environment for checking breach notification

[12] Each manual event which is sent to GA could override the anonymizeIp flag if the developer wants it to behave like this. However, this is probably a rare case. An event can be any interaction like a page view or a click on a button.
[13] In case of the legacy *ga.js* the target is `http://www.google-analytics.com/_utm.gif`
[14] `https://github.com/AliasIO/Wappalyzer`

doi: 10.2313/NET-2018-11-1_08

compliance to the GDPR.
First of all, it only works if the passwords get stolen or forwarded to third-parties. However, there might be other personal data that leaks which requires a notification.
Next, although some websites will do so, most websites probably don't store passwords in plaintext but hashed using salted cryptographic hash functions like PBKDF2 or bcrypt, thus, an attacker might not be able to retrieve the actual password from the leaked hashed password in a short amount of time. As a result, it is not really possible to check the time it takes for the controller to notify the data subject.

Beside checking the controller, we could also check how the supervisory authority handles information of breaches by site operators. This way we could understand how well authorities in different countries implement their part of the GDPR and react to breach notifications by controllers.
Therefore, a fake website with fake users could be set up. We could then fake a breach and notify the corresponding supervisory authority of the breach and log how they react.

While this paper focuses on websites, it is noteworthy that data-stealing and breach notifications is also a highly interesting topic for mobile applications.

3.2.5 *Access To Personal Data & Data Portability*

It might be interesting to check whether a service sends out all information it stores about a user when the user requests access to its data or whether the exported data set is not complete. Therefore, a user could manually add personal data to a given service, exactly logging which information were provided. The user can then compare his log with the data the operator sends him after a request according to Art. 15..

The right for access to personal data can also be used to find out what other data sources a controller used in order to complement a user profile.

3.2.6 *Measures Which Are Hard To Check*

As we discussed it is feasible to check some measures manually or even automatically. However, there are a couple of measures which are harder to check or where the possible approaches are not feasible in reality:

1. Does the operator store records of processing activities? (Art. 30.)
2. Do webserver logs contain personal data?
3. Is personal information given to third-party providers?
4. Is personal data really deleted after an according request? (Art. 17.)
5. Usage of personal data for additional/other purposes than consent was given for? (Art. 7.)

4. WEBSITE ANALYSIS FOR COMPLIANCE

For this section, we pick a website and analyze it for compliance with the GDPR. We therefore use the techniques described in 3.2 and also focus on the mentioned measures, namely IP Anonymization, Newsletter and Privacy Policy.
At the current point in time - where the GDPR is not implemented yet - it is hard to check for other measures like Access To Personal Data or Breach Notifications.

4.1 PlusPeter

PlusPeter is a brand provided by *PlusPeter GmbH, Berlin* which offers online printing services for students.
We decided to investigate PlusPeter because it is a rather small start-up and they collect personal information about students which makes them attractive to analyze since they are collecting personal data which the users want to know well protected.

4.1.1 *Overview And Objects Under Analyis*

There are two domains which belong to the service. The first domain is `www.pluspeter.com` which is running a WordPress instance and provides landing pages and detailed information about their services.
The second domain is `app.pluspeter.com` which runs the web application where students can login and use the service.
We are taking a look at both domains.

4.1.2 *www.pluspeter.com*

Third Party Integrations

The site uses Google Analytics which is integrated via Google TagManager. Furthermore, one of Google's advertisment networks - Doubleclick - is used.

When inspecting the network traffic as in Figure 1, we first see a request to *google-analytics.com/r/collect* which indicates that an event is being tracked in GA.
This request does not contain the *aip* parameter and thus, does not instruct GA to anonymize the IP. However, the GA service returns a status code of *302* and redirects the request to the same endpoint on the domain *stats.g.doubleclick.net*, which the browser will fetch in a second request.
This second request to the doubleclick network contains the *aip* parameter, and thus, instructs GA to anonymize the IP address (3.2.1).

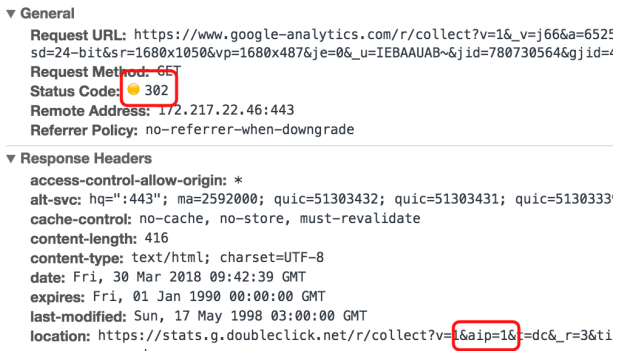


Figure 1: Network request to GA on pluspeter.com

We performed this check on multiple subpages and with different interactions within each page.

To sum it up, the first request to GA did not provide the flag for IP anonymization, but according to the HTTP response code, the request was not processed by the GA collector but

doi: 10.2313/NET-2018-11-1_08

just redirected to a domain belonging to Doubleclick. This second request uses IP anonymization, so we can say that www.pluspeter.com uses GA with anonymized IP addresses.

Newsletter
Although the privacy policy mentions a newsletter, PlusPeter does not seem to provide a public newsletter we could check for compliance. However, we can inspect the rest of the privacy policy.

Privacy Policy
Google Analytics is correctly mentioned and the privacy policy[15] (last change 30.03.2017) also states how to disable the tracking of GA. The policy also covers the data which is collected when using the contact form of the site.

The business model of PlusPeter probably includes using the provided data about students (for example the topic of studies and the current semester) to allow companies to add targeted advertisements to the documents PlusPeter prints for the students for free. This is also mentioned in the privacy policy, where the policy explains that personal data is used for self-promotion of own services, as well as the promotion of services of other companies.
The document also says that personal data is forwarded to selected companies for postal advertisements. This is noteworthy, since it allows PlusPeter to not only use the data to offer targeted advertisments to their customers, but it also allows them to forward the corresponding data as is to companies.
Note, that according to the GDPR the users are able to object the forwarding of their personal data (2.1.5).

Access To & Removal Of Personal Data
PlusPeter's privacy policy already contains a section related to the access to personal data where they state that the user can request insights about what data PlusPeter stores. The user can also request the data to be changed or removed.
These sections have probably been preparations for the implementation of GDPR in May 2018.

The public website (`www.pluspeter.com`) does not have any further forms and seems not to collect any other data as already discussed.

4.1.3 app.pluspeter.com

Third Party Integrations
The app also uses Google Analytics as we can see by inspecting the network traffic of the browser.

Figure 2: Request to GA on app.pluspeter.com

Figure 2 shows that the tracking using GA within the app does not anonymize the IP address - unlike `www.pluspeter.com`.

It is noteworthy that this is not only the case for logged in users, but also if new users visit the registration page[16].

In this case, we did not provide consent for collecting our IP address on the registration page. As a result, this is probably not compliant with the GDPR (s. 2.1.2).

Request Of Consent
According to Art. 7.7 GDPR the website operator needs to clearly state what data is collected and for which purpose it is used. It is not enough to simply state it in legalese in an enormous privacy policy.

In the case of PlusPeter, there are two forms where the user activly enters data.
The first form is the registration form in which the user types in his email address and a password.
The second form is located in the profile settings where the user enters his personal data as well as information about the study progress (name, birthday, university- and private mail address, phone number, address of residence, university, degree, major / study path and the current semester of enrollment).

As a screenshot of the second form shows (figure 3), there is no text stating what the entered data is used for, nor what exactly is stored.
This is probably not compliant to the GDPR (s. 2.1.2).

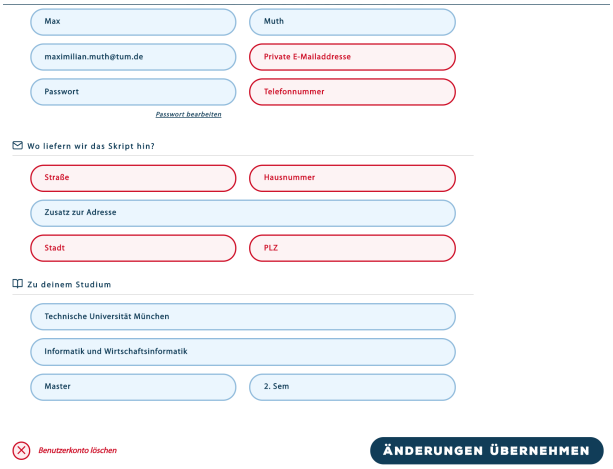


Figure 3: Profile Information on app.pluspeter.com

Privacy Policy
The privacy policy we discussed for `www.pluspeter.com` is also applicable to the application at `app.pluspeter.com`.

In the section about GA the policy states, that the IP addresses will be anonymized. This is true in case of `www.pluspeter.com`, but is not true on `app.pluspeter.com` as shown above (even for visitors who are not registered for the service).

[15] `https://web.archive.org/web/20180330092452/https://www.pluspeter.com/datenschutzerklarung/`

[16] `https://app.pluspeter.com/students/registration/`

doi: 10.2313/NET-2018-11-1_08

As a result, this claim in the privacy policy is wrong. We are not sure how penalties for misinformation in privacy policies are regulated, it would probably need a lawyer to look at the specific case in order to name a penalty.

Summary
While it is interesting to see, that the privacy policy already covers some changes related to the GDPR with the date of last change on March 2017 (more than one year before the implementation of the GDPR), PlusPeter's websites still have some behaviors which are likely not compliant with the GDPR.

The app does not anonymize IP addresses when sending tracking events to GA, although the privacy policy states so and the visitors did not gave consent for that.

Furthermore, the app doesn't state what the collected data is used for next to the form. The information is partly available in the privacy policy, but that's not enough for being compliant with the GDPR (2.1.2).

Nevertheless, we see that PlusPeter put in some effort in trying to be compliant, like anonymizing IP address collection on the WordPress site or stating the possibility to request personal data. Regardless of these efforts, they still made some mistakes.

5. CONCLUSION

We presented the new key concepts of the GDPR which will be implemented May 25th, 2018. The focus was identifying needed changes website operators need to perform in order to be compliant with the new ruleset.
Most important topics we discussed were the general lawfulness of processing, the way it is required to get consent from users, concepts like access to personal data and its removal and how to handle data breaches.
With these understandings, we identified the required changes for most website operators, namely not collecting personal data without the user's consent with forms, integrations like Google Analytics or webserver logs, changing the way newsletter sign-ups are performed and monitored (Double-Opt-In), changes in the integration of social media services and comment systems as well as required changes in the privacy policy.

While discussing all of this, we noticed that it might be hard for smaller website operators to understand which changes are needed and what behavior is not allowed in order to be compliant. This was also shown in our analysis of the websites by PlusPeter GmbH, who already put in some effort to be compliant with the GDPR, but still missed some points or did mistakes.

Overall, it looks like the GDPR aims at improving the user's privacy by putting the user more in control of what data is processed in which ways. On the other hand, it increases the workload for small site operators to be compliant which might be even not viable for some operators.

As a final reminder, this paper is no legal advise but an opinion from an engineering point of view.

6. REFERENCES

[1] Bayerisches Landesamt für Datenschutzaufsicht. EU-Datenschutz-Grundverordnung (DS-GVO) - Das BayLDA auf dem Weg zur Umsetzung der Verordnung - Umgang mit Datenpannen - Art. 33 und 34 DS-GVO. `https://www.lda.bayern.de/media/baylda_ds-gvo_8_data_breach_notification.pdf`, 2016. [Online; accessed March 2018].

[2] Catalin Cimpanu. US Congress Passes CLOUD Act Hidden in Budget Spending Bill. `https://www.bleepingcomputer.com/news/government/us-congress-passes-cloud-act-hidden-in-budget-spending-bill/`, 2018. [Online; accessed March 2018].

[3] J. DeBlasio, S. Savage, G. M. Voelker, and A. C. Snoeren. Tripwire: inferring internet site compromise. In *Proceedings of the 2017 Internet Measurement Conference*, pages 341–354. ACM, 2017.

[4] Derric Gilling. Capturing AJAX API Requests From Arbitrary Sites With a Chrome Extension. `https://dzone.com/articles/how-we-captured-ajax-api-requests-from-arbitrary-w`, 2017. [Online; accessed March 2018].

[5] Dr. Thomas Schwenke. MailChimp, Newsletter und Datenschutz. `https://drschwenke.de/mailchimp-newsletter-datenschutz-muster-checkliste/`, 2016. [Online; accessed March 2018].

[6] Dr. Thomas Schwenke. Datenschutz und ePrivacy 2018 – Änderungen für Onlinemarketing, Tracking und Cookies. `https://drschwenke.de/datenschutz-eprivacy-online-marketing-cookies/`, 2017. [Online; accessed March 2018].

[7] EUROPEAN PARLIAMENT. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). `http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=en`, 2016. [Online; accessed March 2018].

[8] J. Farrell and P. Klemperer. Coordination and lock-in: Competition with switching costs and network effects. *Handbook of industrial organization*, 3:1967–2072, 2007.

[9] Google. IP Anonymization in Analytics. `https://support.google.com/analytics/answer/2763052`. [Online; accessed March 2018].

[10] J. R. Mayer and J. C. Mitchell. Third-party web tracking: Policy and technology. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 413–427. IEEE, 2012.

[11] Netzpolitik.org. Cambridge Analytica: Was wir über „das größte Datenleck in der Geschichte von Facebook" wissen. `https://netzpolitik.org/2018/cambridge-analytica-was-wir-ueber-das-groesste-datenleck-in-der-geschichte-von-facebook-wissen/`, 2018. [Online; accessed March 2018].

[12] Slaughter and May Legal Services. New rules, wider reach: the extra- territorial scope of the GDPR.

https://www.slaughterandmay.com/media/2535540/new-rules-wider-reach-the-extraterritorial-scope-of-the-gdpr.pdf, 2016. [Online; accessed March 2018].

[13] M. Wachs, Q. Scheitle, and G. Carle. Push away your privacy: Precise user tracking based on tls client certificate authentication. In *Network Traffic Measurement and Analysis Conference (TMA), 2017*, pages 1–9. IEEE, 2017.

doi: 10.2313/NET-2018-11-1_08

Was ist Privatsphäre?

Benedikt Müller
Betreuer: Marcel von Maltitz
Seminar Future Internet SS2018 Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: muellben@in.tum.de

KURZFASSUNG

Während täglich immer mehr Menschen Zugang zum Internet erhalten und gleichzeitig der Wettbewerb zwischen datenverarbeitenden Unternehmen immer monopolistischer wird, ist es von zentraler Bedeutung, die Daten der Menschen zu schützen. Das wiederum kann nur funktionieren, wenn man einen sehr klaren Begriff von Privatsphäre besitzt. Diesen Begriff versucht diese Arbeit zu vermitteln, indem sie zu Beginn einen historischen philosophischen Diskurs über Privatheit und Öffentlichkeit führt. Daraufhin werden einige grundlegende Konzepte betrachtet, die sich an verschiedene Ideen aus der Natur, der Technikphilosophie, der Rechtsprechung und der Kapitalismuskritik bedienen. Außerdem wird das Konzept der kontextuellen Integrität vorgestellt, welche mehrere Sphären für die verscheiden Bereiche des menschlichen Lebens definiert, in denen es angemessen ist, bestimmte Informationen zu teilen und andere nicht. Zuletzt geht es um Datenschutzziele, welche ähnlich wie die Schutzziele der IT-Sicherheit eine Hilfe für die Implementierung von Systemen bieten. Allerdings schützen diese, wenn sie erreicht werden, nicht nur die personenbezogenen Daten, sondern auch die Privatsphäre.

Schlüsselworte

Privatsphäre, Individualität, Öffentlichkeit, Schutzziele, Kontextuelle Integrität

1. EINLEITUNG

„Ich will nicht in einer Welt leben, in der alles, was ich sage, alles was ich mache, der Name jedes Gesprächspartners, jeder Ausdruck von Kreativität, Liebe oder Freundschaft aufgezeichnet wird“. Dies war die Antwort von Edward Snowden auf die Frage, wieso er seinem bisherigen Arbeitgeber, dem amerikanischen Geheimdienst, den Rücken zugekehrt hat und durch die Weitergabe von geheimen Dokumenten, die eine bereits vermutete globale Überwachungsstruktur bestätigt haben, eine weitreichende Diskussion über Datenschutz los getreten hat. Gleichzeitig gibt dieses Zitat aber auch einen Ausblick auf die Antwort der Frage was Privatsphäre ist, die in dieser Arbeit gegeben werden soll. Blickt man in die Vergangenheit, stellt man fest, dass man sich bereits sehr früh mit Konzepten von Privatheit und Öffentlichkeit beschäftigt hat. Jenes Konzept, dass das menschliche Leben in zwei verschiedene Sphären unterteilt, hat sich allerdings vielfach gewandelt. Wenn man es heute betrachtet, macht man dies meist aus der Sicht von Datenschutz. Einen weitreichenden Einfluss auf die Entwicklung verschiedener Auffassungen von Privatsphäre hat auch die Erfindung neuer Technik, wie zum Beispiel der Fotoapparat, das Telefon oder das Internet. Als am Anfang des 19. Jahrhunderts die Diskussion um ein Recht auf Privatsphäre bei einigen Juristen beginnt, sieht man das Problem noch bei den Medien, die durch Fotografien und Tonbandaufzeichnungen immer weiter in das Privatleben von Personen eindringen[17]. Heute ist das Machtverhältnis zwischen großen datenverarbeitenden Unternehmen und deren Kunden mehr in den Fokus der Datenschützer geraten. Zuletzt im Fall von Cambridge Analytica, einem Unternehmen, das ohne Erlaubnis von Facebook oder deren Nutzern Millionen von Nutzerdaten abgreifen konnte, ohne das dies bemerkt wurde.[1] Diese Gefahren ergeben sich dadurch, dass die wirtschaftliche Struktur, in der wir leben dafür sorgt, dass Unternehmen zu Monopolen tendieren[9] und auf Grund ihrer global agierenden Struktur schwieriger in Schranken zu weisen sind, was sie immer mächtiger werden lässt. Um diese Entwicklung in den Griff zu bekommen, muss der Staat und seine Behörden die Entwicklung, immer tiefer in das Privatleben einzudringen, nachahmen, was wiederum zu weiteren Grenzüberschreitungen führt und die Grundrechte der Bürger weiter verletzt. Deswegen ist der Schutz von Grundrechten ein weiterer Punkt, der in die Betrachtung von Privatsphäre mit einfließen muss. All diese Punkte werden nun nachfolgend betrachtet mit dem Ziel, eine genauere Vorstellung von Privatsphäre zu vermitteln, mit dem Hinblick auf die Implementierung von konkreten Systemen.

2. PRIVATHEIT UND ÖFFENTLICHKEIT

Privatheit und Öffentlichkeit lassen sich nicht so einfach definieren, vielmehr müssen beide Begriffe in ihrem historischen, philosophischen und politischen Kontext betrachtet werden. Dieser befindet sich in einem steten Wandel, wodurch sich auch unser Verständnis dieser Begriffe fortlaufend verändert. Bereits im fünften Jahrhundert vor Christus wird die Ambivalenz von Aristoteles aufgezeigt[4]. Das antike Verständnis beruht auf starken regionalen Grenzen. So wird Öffentlichkeit in der Stadt auf Plätzen und Märkten gelebt, während Privatheit alles umfasst, was zu Hause mit der Familie stattfindet. Dadurch ergibt sich auch die damalige Funktion der beiden Sphären. Während Privatheit nur dem Fortbestand der Menschheit dient, ist die Öffentlichkeit für Fortentwicklung verantwortlich. Weitere Funktionen der privaten Sphäre sind für ihn der Freiraum zur Muße und eine selbstbestimmte Lebensführung. [6] Gleichzeitig versteht Aristoteles den Menschen aber auch als Zoon Politikon, ein Wesen, welches zum einen Gemeinschaft bildet, aber zum anderen in ihr auch ein Anteil zum vollendeten Leben erhalten kann.

doi: 10.2313/NET-2018-11-1_09

Habermas schreibt zu dieser Konzeption folgendes:

„Das Reich der Notwendigkeit und der Vergänglichkeit bleibt im Schatten der Privatsphäre versunken. Ihm gegenüber hebt sich die Öffentlichkeit, im Selbstverständnis der Griechen, als ein Reich der Stetigkeit und der Freiheit ab.“[4]

Man könnte auch sagen, dass die Griechen die Vernunft verstaatlicht haben. Die Kategorien, die Aristoteles geschaffen hat, änderten sich erst in der Epoche der Aufklärung wieder. Zwar teilt sich die Öffentlichkeit durch die Entstehung des Christentums in eine kirchliche und eine staatliche Sphäre, doch der Zugang zu ihnen bleibt genauso elitär, wie auch die Kategorien in denen gedacht wird.

Um zu verstehen, wie sich das Selbstverständnis der Menschen durch die Aufklärung gewandelt hat, bietet sich folgendes Zitat von Kant an, in dem er die Frage beantwortet, was Aufklärung ist.

„Aufklärung ist der Ausgang des Menschen aus seiner selbst verschuldeten Unmündigkeit. Unmündigkeit ist das Unvermögen, sich seines Verstandes ohne Leitung eines anderen zu bedienen. Selbstverschuldet ist diese Unmündigkeit, wenn die Ursache derselben nicht am Mangel des Verstandes, sondern der Entschließung und des Mutes liegt, sich seiner ohne Leitung eines anderen zu bedienen.“[8]

Kant fordert die Menschen auf, ihre Vernunft wieder selbst zu benutzen und dadurch kehrt sich das Verhältnis um.[12] Während zuvor die Kirche und der Staat die Deutungshoheit über Wahrheit, Moral und das Gute innehaben, soll nun jeder Mensch selbst über all diese Punkte entscheiden, indem man sein Verstand benutzt. Man muss hierbei aber berücksichtigen, dass sowohl das Konzept von Aristoteles, wie auch das von Kant einem Ideal entsprechen und nicht die Wirklichkeit abbilden. Dies erklärt, weshalb bereits 200 Jahre nach der Aufklärung Adorno in einem Brief an seinen Kollegen Horkheimer bereits von einer Epoche spricht, „welche die Privatsphäre zu liquidieren sich anschickt.“[2] Das größte Problem sieht Adorno darin, dass die Kontrollfunktion, die Kant der öffentlichen Meinung zuschreibt, nur legitim ist, wenn sie sich in ihrer Wahrheit ebenfalls kontrollieren lässt. Allerdings lässt sich das nur als „statistischer Durchschnittswert der Meinung aller einzelnen“[3] verwirklichen. Ein weiteres Eindringen in die Privatsphäre findet außerdem durch den Einzug von Radio und Fernsehen in der ersten Hälfte des 20. Jahrhunderts statt, die einen „Übergang vom kulturräsonierenden zum kulturkonsumierenden Publikum forcierten.“[7] Das von Kant angestrebte Ideal wird mit ihnen deutlich verfehlt. Dies liegt zum einen an ihrer Struktur, welche sich immer aus Sender und Empfänger zusammensetzt und somit keinen Widerspruch zulässt und zum anderen an ihrer Absatzorientierung, wodurch sie nur bereits bekannte Muster wieder hervorbringen.[7] Gleiches gilt auch für das Medium Internet, welches zwar bidirektional funktioniert, aber den Menschen trotzdem nicht ermöglicht sich frei zu entfalten.[12] Damit wird deutlich, wie sich die private Sphäre im Laufe der Zeit immer mehr verkleinert hat. Hierbei wurde die in den letzten Jahren bekannt gewordene Überwachungsstruktur der Geheimdienste noch nicht in die Beurteilung miteinbezogen.

3. GRUNDLEGENDE KONZEPTE VON PRIVATSPHÄRE

Aus dem vorherigen Kapitel geht hervor, wie weit die Privatsphäre bereits eingeschränkt ist und macht damit deutlich, dass es wichtig ist, noch Bestehendes zu schützen. Allerdings lassen sich aus der Betrachtung noch keine konkreten Handlungsanweisungen ableiten, welche aber zum Beispiel unter dem Gesichtspunkt einer Privatsphäre freundlichen Implementierung von Software nötig sind. In diesem Kapitel sollen nun Konzepte betrachtet werden, welche in den späteren Kapiteln als Grundlage dienen um praktische Handlungsempfehlung zu erstellen.

3.1 Persönlicher Freiraum

Das Konzept von Privatsphäre und des persönlichen Freiraums lässt sich bis zur Tierwelt zurückverfolgen.[18] Es ist weitläufig bekannt, dass viele Tiere ein eigenes Revier besitzen, in dem sie entweder alleine leben oder es mit anderen Artgenossen teilen und es vor Eindringlingen schützen. Als Beispiel lassen sich Löwen nennen, die mit ihrem Rudel zusammen ein Gebiet besetzen, welches sie sowohl vor fremden Artgenossen, aber auch vor anderen Raubtieren verteidigen. Der Hauptgrund dafür liegt vermutlich in der Ressourcensicherung. Ein weiterer Mechanismus zum Aufteilen des territorialen Abstands zwischen Individuen wurde von E. T. Hall beobachtet[18]. Er bezeichnet den Raum, welchen Tiere zwischen einander freiwillig frei lassen, als „personal distance“[5]. Dieser lässt sich zum Beispiel bei Vögeln, die sich auf eine Stromleitung setzen, beobachten. Beim Menschen wird hierbei meist von der Intimsphäre gesprochen. Beide Konzepte finden sich beim Menschen aber auch in einem größeren Kontext wieder. Beispiel hierfür ist des Menschen eigenes Haus, das ihm einen Rückzugsort bietet und das er mit einem Schloss an der Tür vor ungebetenen Gästen schützt. Hierbei macht es allerdings keinen Unterschied, ob der Mensch in einer Villa lebt oder nur ein Zimmer als sein Zuhause ansieht. Es geht darum, dass man sich ein Umfeld schafft, in dem man gewisse Dinge erwarten kann, um nicht immer dem Unbekannten und den damit verbundenen Stress ausgesetzt zu sein. In anderen Worten, dass mit allen den Menschen zur Verfügung stehenden Sinnen nichts Neues wahrgenommen wird. In Bezug auf das eigene Haus, möchte man also nicht, dass man dort zum Beispiel ein neues Parfüm riecht, denn dies ist ein Zeichen, dass in den persönlichen Bereich eingedrungen wurde. Eine wichtige Erkenntnis zu diesem sehr grundlegenden Verständnis von Privatsphäre ist, dass dieser Raum, den man sicher erzeugt hat, nicht nur physischer Natur sein muss. Genauso lässt sich so ein Raum zum Beispiel auch im Internet schaffen, indem man bei sozialen Netzwerken einen Account anlegen kann und diesen mit vertrauten Dingen, wie Bildern und Informationen füllt. Allerdings ergibt sich dabei ein sehr großes Problem. Während man zum Schutz von physischen Räumen noch alle Sinne zur Kontrolle zur Verfügung hat, besitzt man im Internet oft überhaupt keine Möglichkeit, den dort entstandenen, privaten Raum zu schützen, sondern wird gezwungen den Schutz des privaten Raums einer anderen Partei anzuvertrauen.

doi: 10.2313/NET-2018-11-1_09

3.2 Das Recht in Ruhe gelassen zu werden

Ein weiteres Konzept zum Schutz der Privatsphäre erstellen die beiden Rechtswissenschaftler S. D. Warren und L. D. Brandeis erstmals 1890[17]. Sie erkennen bereits sehr früh, welche Eingriffe die Massenmedien, zum Beispiel durch in Zeitung abgedruckte Fotografien intimer Momente, in das private Leben der Menschen ermöglichen und auch die damit verbundenen Gefahren. Als Grundlage für ihr Konzept dient ihnen das Recht auf körperliche Unversehrtheit und das Recht auf Besitz. Beides sehen sie als Grundpfeiler für menschliches Zusammenleben. Allerdings verändert sich die Interpretation im Laufe der Zeit. Während man anfangs eine reine physische Unversehrtheit gewährleistet wird, wurde später auch anerkannt, dass sowohl die Gefühle, wie auch der Verstand eines gewissen Schutzes bedürfen. Außerdem wurde akzeptiert, dass Produkte des Verstandes, wie zum Beispiel Kunst aber auch Handelsgeheimnisse, einen Besitz darstellen, den es ebenfalls zu schützen gilt. Da aber der Schutz des geistigen Eigentums, also der Schutz von konkreten Erzeugnissen geistiger Arbeit bzw. der Schutz vor Verleumdung, womit der Schutz vor falschen bzw. Rufschädigenden Behauptungen gemeint ist, nicht ausreichen um Personen vor dem Eindringen der neuen Medien in die Privatsphäre zu schützen, erstellten die beiden Juristen das Konzept, welches sie als „Recht in Ruhe gelassen zu werden“betiteln. Darunter verstehen sie, dass jeder Mensch selbst entscheiden soll, in welchem Maß seine Gedanken, Empfindungen und Gefühle mit anderen geteilt werden sollen und wenn sie geteilt wurden, welches Maß an Aufmerksamkeit ihnen zukommen soll.[17] Dabei machen sie auch kein Unterschied, in welcher Form diese Informationen bereitgestellt werden. Das Konzept geht damit deutlich über das des Copyrights hinaus, da Copyright nur bereits sehr gefestigte Produkte des Menschen schützt. Es vermittelt, auch wenn es noch keine klaren Möglichkeiten bietet ein Recht auf Privatsphäre davon abzuleiten, bereits eine genauere Vorstellung, was es zu schützen gilt.

3.3 Recht auf Kontrollierbarkeit der eigenen Daten

Alan Westin entwickelt in seinem 1967 erschienen Buch Privacy and Freedom[18] das Konzept von Privatsphäre weiter. Seine Arbeit gilt als Grundlage dessen, was die meisten Menschen heutzutage unter Privatsphäre verstehen. Er unterteilt sie in vier verschiedene Kategorien, Abgeschiedenheit, Vertrautheit, Anonymität und Zurückgezogenheit. Mit Abgeschiedenheit ist eine physische Trennung von anderen Individuen gemeint. Unter die Kategorie der Vertrautheit fallen enge Freundschaften oder Beziehungen. Anonymität beschreibt das Bedürfnis des Menschen in der Öffentlichkeit unerkannt zu bleiben. Die vierte Kategorie macht sein Konzept besonders. Sie besagt, dass der Mensch bei gewissen Bereichen einen psychologischen Schutzwall zieht, der vor ungewollten Eindringen schützen soll. Diesen Wall müssen andere respektieren, wenn man gewisse Informationen nicht teilen will[18]. Als ein Beispiel für eine Überschreitung des Grenzwalls nennt er die Praktik Menschen mittels Lügendetektoren zu befragen, da durch diese Art von Befragung möglicherweise Informationen gewonnen werden, welche der Befragte nicht teilen möchte. Dieses Konzept sieht also auch einen Einschnitt in die Privatsphäre, wenn zum Beispiel Facebook Daten benutzt, die es bekommen hat, wenn Personen nicht wissen, dass wenn sie ihren Freunden Nachrichten schreiben, diese ebenfalls von Facebook gelesen werden, dies aber eigentlich nicht wollen. Man kann auch von einem Recht auf Kontrollierbarkeit der eigenen Daten sprechen. Westin betrachtet bereits den Kontext in dem Informationen geteilt werden, wie es später auch Nissenbaum macht. In seinen Kategorien wird das Umfeld, in denen sich die Informationen befinden, immer größer doch die Menge an Informationen, die preisgegeben werden, immer kleiner.

3.4 Beherrschbarkeit der Maschinerie

Die Überschrift dieses Abschnitts geht auf ein Zitat von Wilhelm Steinmüller aus dem Jahr 2009 zurück[14], in dem er den Diskurs nicht aus der bisherigen Perspektive betrachtet, in der es um den Schutz des Privaten geht, sondern, dass die Technik, so wie sie geschaffen wurde, nicht mehr kontrollierbar ist. Als anschauliches Beispiel lässt sich das E-Mail-Protokoll betrachten. Zur Zeit seiner Erfindung hatte das Internet noch eine überschaubare Anzahl an Teilnehmer. Außerdem wollte man alles möglichst einfach halten, um nicht den Überblick zu verlieren, weshalb zu Beginn noch keine Sicherheitsmechanismen berücksichtigt wurden. Dies war außerdem auch noch nicht nötig, da zumeist nur Informationen zwischen Universitäten ausgetauscht wurden, die früher oder später meist eh veröffentlicht wurden. Als aber immer mehr Menschen begannen vertrauenswürdigere Informationen auszutauschen, wurde es immer wichtiger kontrollieren zu können, was mit den über E-Mail vermittelten Daten passiert. Ein ähnlicher Verlauf lässt sich bei sehr vielen technischen Innovationen finden, welche einen Einfluss auf die Privatsphäre haben. Deswegen sollte man schon bei der Implementierung darauf achten, dass sich das System trotz zukünftiger unbeabsichtigter Anwendungen noch kontrollieren lässt.[13] Denn mit den Verlust der Kontrolle über die Technik, verliert man gleichzeitig auch die Kontrolle über die Daten.

3.5 Machtverhältnisse

Aufgrund der Enthüllungen über die globale Überwachungsmaschinerie der Geheimdienste der USA und Großbritanniens durch Edward Snowden aber auch durch Enthüllungen über nichtstaatliche Konzerne wie Facebook, wird das Bild von „mächtigen Organisationen als Risikogeber[n]“[15] in der Öffentlichkeit, sowohl in Deutschland, wie auch in den USA immer stärker verankert. Das Problem des Schutzes der Privatsphäre lässt sich auch aus dem Blickwinkel des Konflikts zwischen Privatperson und öffentlicher Organisation sehen. Während früher dieser Konflikt hauptsächlich zwischen Arbeiter und Unternehmen bestand, besteht er heutzutage außerdem noch zwischen Verbraucher und Unternehmen. Auf beide Konfliktsituationen lässt sich die Kritik, die Karl Marx im ersten Band des Kapitals festgehalten hat, erweitern. Dabei unterscheidet Marx zwischen Ausbeutung und Entfremdung. Der Verbraucher gibt sobald er seine Daten mit dem Unternehmen teilt meist seine gesamten Besitzrechte an ihnen ab. Dadurch wird es für ihn unmöglich, zu bestimmen was weiter mit ihnen passiert. Zum Beispiel hat man bei Facebook keinen Einfluss auf den Algorithmus, der mit den Daten, die man hochlädt, den Newsfeed generiert. Bei Marx entfremdet man sich, da man das Produkt seiner Arbeit nicht mehr wiedererkennt[10]. Im Bezug auf Datenverarbeitung findet Entfremdung statt, wenn sich man in dem Ergebnis der Verarbeitung nicht wieder erkennt. Wenn man

doi: 10.2313/NET-2018-11-1_09

zum Beispiel bei seinem Newsfeed auf Facebook nicht mehr erkennt, ob die eigenen Interessen für den Inhalt des Feeds verantwortlich sind oder Entscheidungen von Facebook. Facebook gibt dabei offen zu, dass es gewisse Aufmerksamkeitsströme so lenkt, dass die Nutzer die Anwendung möglichst lange nutzen und dadurch möglichst viele persönliche Informationen preisgeben. Ein weiterer Punkt der Kritik von Marx bezieht sich auf den Ausbeutungscharakter der kapitalistischen Produktionsweise, bei welcher der Arbeiter nur einen geringen Teil des von ihm geschaffenen Wertes erhält und der Rest dem Kapitalgeber zufällt. Analog verhält es sich mit den den Betreibern von Webdiensten, welche ihre Dienste zwar kostenlos bereitstellen, was dem Lohn des Arbeiters entspricht, sie allerdings mit den Daten ihrer Nutzer deutlich mehr verdienen wie sie für den Betrieb ihrer Dienste eigentlich benötigten, was sie wiederum als Profit behalten. Gleichwohl trifft auch die Prognose von Marx, dass sich mit der Weiterentwicklung der Technik nicht die Bedingungen der Arbeiter verbessern sondern sie immer mehr ausgebeutet werden können[10], auf den Konflikt zwischen Unternehmen und Verbraucher zu. Ein Beispiel dafür ist das Smartphone, welches von Firmen wie Facebook und Google als weiteres Mittel benutzt wird um noch mehr Daten über den Benutzer zu sammeln. Sie ermöglichen ihm zwar weiterhin die kostenlose Nutzung ihrer Dienste nun auch unterwegs allerdings sammeln sie jetzt auch Daten wie den Aufenthaltsort. Der Schutz von Privatsphäre ist also auch verbunden mit marxistischer Systemkritik.

Ein weiteres Problem für eine gerechte Machtverteilung ergibt sich aus der Neigung industrieller Prozesse hin zu Standardisierung, Modularisierung, Technisierung und Automatisierung.[13] All diese Punkte bergen Gefahr für die Privatsphäre. Zum einen gibt es keinen Standardmenschen, da alle Menschen auf unbegrenzte Art und Weise unterschiedlich sind. Dies ist in einem computerisierten System aber nicht abzubilden, weshalb persönliche Daten unter Umständen in Schubladen gesteckt werden, in die sie nicht richtig hinein passen und durch die Modularisierung werden diese Zuteilungen noch weiter verschärft. Automatisierung verbindet letztlich alle genannten Punkte, deswegen ist es nötig, dass Unternehmen offenlegen, wie genau sich der Bearbeitungsprozess der personenbezogenen Daten zusammensetzt um ein möglichst gerechtes Machtverhältnis sicherzustellen. In der Praxis besteht die Sicherstellung meist aus zwei Elementen, der Datenschutzaufsicht und der verwendeten Technik. Laut Art. 37 Abs. 1 der neuen Datenschutzgrundverordnung besteht die Pflicht, einen Datenschutzbeauftragten zu benennen, wenn personenbezogene Daten von einer öffentlichen Stelle verarbeitet werden oder wenn die Kerntätigkeit des Unternehmens in der Verarbeitung solcher Daten besteht. Der Datenschutzbeauftragte ist dann für die Überwachung der Organisation verantwortlich, sodass geltende Gesetze eingehalten werden. Eine andere Möglichkeit ein gewisses Datenschutzniveau sicherzustellen, ist die erzwungene Verwendung bestimmter Absicherungstechniken. Beim Online-Banking dürfen Banken zum Beispiel die Daten ihrer Kunden nicht unverschlüsselt übertragen.

4. KONTEXTUELLE INTEGRITÄT

Mit dem konzeptionellen Gerüst der kontextuellen Integrität hat die amerikanische Professorin Helen Nissenbaum ein Diskurs geschaffen, der die Sicht auf ein Recht auf Privatsphäre stark beeinflusst hat. Es ist nicht an eine bestimmte Zeit oder einen bestimmten Ort gebunden,[11] sondern entsteht aus den Dichotomien, vertraulich nicht vertraulich, privat öffentlich und privat staatlich. Die zugrundeliegende Betrachtung von Nissenbaum ist, dass alles was der Mensch macht in einen gewissen Zusammenhang passiert. Sie nennt diese Zusammenhänge „Sphären des Lebens“. Das Leben jedes Menschen besteht aus vielen solchen Sphären, zum Beispiel gibt es die bereits bekannte Sphäre der Familie, die der Medizin oder die der Freunde. Diese Sphären bieten eine Plattform auf der verschiedene Normen gelten. Laut Nissenbaum lassen sich in so gut wie allen Sphären zwei Arten von Normen wiederfinden. Sie bezeichnet die beiden als Normen der Angemessenheit und des Informationsflusses.

4.1 Angemessenheit

Mit einem Beispiel lässt sich verdeutlichen was unter Angemessenheit verstanden wird. Im Kontext eines Arztbesuchs ist es angemessen mit dem behandelnden Arzt über bestimmte Krankheiten zu sprechen, während man mit dem Arbeitgeber solche Informationen in der Regel nicht teilt. Wie sehr sich die Normen in Hinsicht ihrer beschränkenden Funktion, Ausdrücklichkeit und Vollständigkeit von Sphäre zu Sphäre unterscheiden, spielt hierbei keine Rolle. Nissenbaum hält fest, dass es keine Sphäre gibt, ohne eine Norm, welche die Handhabung von Informationen regelt. Angemessenheit definiert die verschiedenen Kontexte und entscheidet, welche Informationen zu ihnen gehören.

4.2 Informationsfluss

Ein anderer Teil an Normen, ist laut Nissenbaum für den Fluss bzw. die Verteilung von Informationen verantwortlich. Als Beispiel dient ihr die Sphäre der Freunde, in der es meist sehr klar ist, welche Informationen mit anderen Freunden geteilt werden dürfen und welche geheim zu halten sind. Aus dieser Betrachtung ergeben sich verschiedene Prinzipien für die Verteilung von Informationen. So lassen sie sich entweder frei nach dem Willen der Informationsträger teilen oder müssen vertrauenswürdig, möglicherweise sogar geheim gehalten werden. Des Weiteren kann auch ein Anspruch, ein dringendes Bedürfnis oder eine Verpflichtung bestehen bestimmte Informationen zu teilen. Wenn man zum Beispiel in der Schule zu spät in den Unterricht kommt muss man den Lehrer meist über den Grund des Zuspätkommens informieren.

4.3 Probleme und Unterschiede

Ein bedeutender Unterschied zu den bisherigen Konzepten ist, dass Information nach Nissenbaum immer an ihren Kontext gebunden ist. Es gibt also keine per se öffentlichen Daten mehr, die von jeden abgegriffen werden dürfen. In Bezug auf Facebook bedeutet das, dass Daten nur für die Personen einsehbar seien sollen mit denen man sie teilen möchte und nicht für Facebooks Werbeabteilung oder die Öffentlichkeit. Es können allerdings Probleme entstehen, wenn man sich nur an bestehenden Normen orientiert. Es wird dadurch erschwert auf technische Innovationen möglichst zeitnahe zu reagieren. Außerdem besteht die Gefahr, dass ein Konzept, welches so stark an praktische Erfahrungen gebunden ist, die moralische Autorität verliert, die es benötigt, um weitläufig akzeptiert zu werden. Um diesen Problemen entgegenzuwirken schlägt Nissenbaum vor, als grundlegendes Prinzip immer den jetzigen Zustand zu bevorzugen und zu betrachten,

doi: 10.2313/NET-2018-11-1_09

inwieweit dieser durch technische Neuerungen beeinflusst wird oder inwieweit bestehende Normen erweitert werden können. Zum Beispiel lassen sich die Normen, die für Briefe bestehen, ohne weiteres auf E-Mails erweitern. Jedoch sagt sie auch, dass es trotz des Status auch möglich sein soll, bestehende Normen durch neue Verfahren zu verändern. Dies soll aber passieren, nachdem man untersucht hat, in welcher Art und Weise sich die neuen Verfahren auf grundlegende Werte auswirken. Einen der sechs von Nissenbaum aufgelisteten Grundwerte bezeichnet sie als Schutz vor Schäden durch Information. Damit meint sie, dass Informationen über eine Person nicht an Andere geraten sollen, welche dieser Person schaden können. Informationen über die Adresse und den Verdienst, welche Einbrechern helfen können ihr nächstes Ziel zu bestimmen, ist ein Beispiel für solche Daten. Ein weiter Grundwert, den Nissenbaum nennt, bezieht sich auf die gerechte Behandlung von Informationen, sodass Unternehmen oder der Staat durch ihre Macht sie nicht unbegrenzt ausnutzen können. Des Weiteren sollen Menschen nicht in der Freiheit und Eigenständigkeit ihrer Handlungen beschränkt werden. Dieser Punkt geht auf die Ideale zurück, die Kant in der Epoche der Aufklärung festgehalten hat. Nissenbaum nennt unter Berücksichtigung dieses Punktes, auch das Recht auf Privatsphäre, „das Recht über die Kontrolle der eigenen Daten“[11] Ein weiterer Punkt, welcher vor dem Eingriff neuer Techniken geschützt werden muss, ist der Erhalt von wichtigen menschlichen Beziehungen, da die Kontrolle über persönliche Informationen eine Kernbedingung ist, um Vertrauen und Intimität herzustellen. Zudem benennt Nissenbaum als Vorteil für den Staat, dass durch eine geschützte Privatsphäre, welche durch neue Verfahren nicht eingeschränkt werden sollen. Zuletzt erwähnt sie außerdem noch verschiedene Werte, die für weniger Regulierung sprechen Dazu gehören das Recht auf freie Meinungsäußerung, eine freie Presse, ökonomische Aspekte sowie eine transparente Regierung und Sicherheit. [11]

5. DATENSCHUTZZIELE

Aus den vorherigen Kapiteln wurde ersichtlich, dass Datenschutz und der Schutz der Privatsphäre sich nicht nur aus dem Bedürfnis nach Privatheit ergibt, sondern als Schutz von Individualität und Freiheit des Menschen gesehen werden kann. In Anbetracht dieser Aufgabe entwickelt Prof. Dr. Andreas Pfitzmann und Martin Rost eine Methode, die bereits bekannte Schutzziele der IT Sicherheit im Kontext des Datenschutz betrachtet, aber auch neue Schutzziele beinhaltet. Als elementare Schutzziele definieren sie Verfügbarkeit, als „gesicherter Zugriff auf Information innerhalb eine festgelegten Zeit“, Vertraulichkeit, als „Gesicherter Nichtzugriff auf Informationen“, Integrität, als „gesicherte Echtheit von Informationen“, sowie Kontingenz, als „gesichert nicht gesicherte Echtheit von Informationen“.[16] Das Schutzziel Kontingenz entsteht aus den beiden Schutzzielen Zugreifbarkeit und Nichtzugreifbarkeit, welche in einem konträren Zusammenhang stehen, welcher zu einem Problem werden kann wenn beide gleichzeitig angestrebt werden. Kontingenz soll gewährleisten, dass der Mensch gewisse Dinge erfolgreich abstreiten können, auch wenn sie ihm von „respektabler Seite“ unterstellt werden. Pfitzmann und Rost sprechen auch davon nicht durch Technik eingeengt zur werden. Von diesen vier Elementarschutzzielen lassen sich einige weitere ableiten, zum Beispiel Verbindlichkeit, Verdecktheit, Anonymität oder Unbeobachtbarkeit. All diese Ziele lassen sich allerdings nicht klar von denen der IT-Sicherheit abgrenzen. Grund dafür ist, dass Datensicherheit eine Grundbedingung für Datenschutz ist. Spezielle Datenschutzziele hingegen sind laut Pfitzmann und Rost Transparenz und Unverkettbarkeit.

5.1 Transparenz

Transparenz bedeutet, dass ein Systemteil für eine Entität beobachtbar gemacht wird. Damit ist die Transparenz eines Systems die wichtigste Voraussetzung dafür, dass ein System kontrollierbar ist. Des Weiteren unterscheiden sie noch ob es sich erkennen lässt, ob ein System nicht transparent ist.

5.2 Unverkettbarkeit

Vollständige Transparenz in Blick auf Datenschutz ist allerdings nicht ausreichend, deswegen wird das Konzept der Unverkettbarkeit eingeführt. Dieses Ziel wirkt darauf hin, dass Daten nicht miteinander verkettet werden können. Dadurch wird erreicht, dass es zum Beispiel für den Staat nur möglich ist, die Daten der Bürger für den Zweck zu benutzen für den sie erhoben wurden, auch wenn unterschiedliche Behörden verschiedene Daten möglicherweise im selben Rechenzentrum speichern. Die Aufgabe von Datenschützern ist laut Pfitzmann und Rost bestehende Verkettungen unter Bedingungen zu stellen oder aufzulösen.

5.3 Schutzmaßnamen

Ein Vorteil der Schutzziele wie Transparenz und Unverkettbarkeit ist, dass von ihnen leicht bestimmte Maßnahmen abgeleitet werden. Um zum Beispiel das Ziel der Kontingenz bei verschlüsselten Nachrichten zu erreichen, müsste ein Verschlüsselungsalgorithmus gewählt werden, der unabhängig von der Länge der eigentlichen Nachricht stets Nachrichten mit einer gleichen Länge erzeugt.[16] Um Transparenz zu gewährleisten muss bereits bei der Konzeption eines neuen Systems eine genaue Dokumentation erfolgen. Des Weiteren muss man in der Lage sein, Vorgänge sowohl während des Betriebs mittels Monitoring, wie auch im Nachhinein durch Protokolle überwachen zu können.

6. FAZIT

Das Thema Privatsphäre ist äußerst facettenreich und verdient eine noch weitaus tiefere und umfangreichere Betrachtung, die im Rahmen dieser Arbeit allerdings nicht möglich ist. Ein Anliegen dem sie aber genüge getan hat, ist die Betonung der Bedeutung des Themas. Denn Jeder der glaubt Privatsphäre besteht, wenn er sich mit den ihn zur Verfügung stehenden Mitteln schützt, verschließt die Augen vor der Realität. Deswegen kann ich auch der häufig im universitären Umfeld anzutreffenden libertären Meinung, dass man als Konsument doch die Wahl hat sich zu schützen beziehungsweise gewisse Dienste nicht zu benutzen, aus mehreren Gründen nicht zustimmen. Zum einen ist man oft nicht in der Lage sich gewisse Dinge auszusuchen, zum Beispiel besitzt nicht jeder genug Geld um das Land in dem man lebt zu wechseln um besser durch Datenschutzgesetze geschützt zu sein. Außerdem sind die zeitlichen und monetären Ressourcen oft zu beschränkt, um sich umfangreich mit allen Eigenschaften einer technischen Innovation zu beschäftigen um dann ein Urteil zu fällen, wie weit sie die Privatsphäre beeinträchtigt und inwieweit man damit leben kann. Aber vor allem widerspricht es einem grundlegenden Prinzip der

doi: 10.2313/NET-2018-11-1_09

Demokratie, dem Vertrauen. Martin Rost[14] schreibt dazu folgendes:

„Es geht um ein begründetes, begründbares, nicht blindes Systemvertrauen. Vertrauen zu gewähren und ebenso abzufordern, macht moderne Gesellschaften so besonders effektiv."

Man könnte sogar weiter gehen und behaupten, dass es nur lohnenswert ist, ein System überhaupt zu erhalten, wenn man ihm vertrauen kann, dass es Dinge ins Positive verändert. In Bezug auf Deutschland und vielen anderen Ländern macht es Sinn, Privatsphäre zu verstärken und gefährdende Verfahren einzuschränken, damit ein möglicherweise verlorengegangenes Vertrauen der Bürger zurückgewonnen wird. Ein Zeichen des verlorengegangen Vertrauens, nicht nur gegenüber dem Staat, sondern auch zu den Unternehmen, wie zum Beispiel Google, ist die Entwicklung, verschiedener Privatsphäre freundlicher und zum Teil dezentraler Systeme. Mit Nextcloud ist es zum Beispiel leicht möglich, einen eigenen Datenspeicher zu erstellen auf den man von Überall zugreifen kann, auch gibt es viele Messaging Apps, die die Nachrichten ihrer Benutzer Ende-zu-Ende verschlüsseln und nicht wie Facebook mitlesen. Damit sich diese Herangehensweise aber sowohl bei der Entwicklung von Systemen, wie auch bei den Anwendern weiter verankert, muss das Verständnis von Privatsphäre in allen Schichten noch erweitert werden.

7. LITERATUR

[1] T. Adams. Facebook's week of shame: the Cambridge Analytica fallout. *The Guardian*, 2018.

[2] T. W. Adorno. Offener Brief an Max Horkheimer. *Die Zeit*, 1965.

[3] T. W. Adorno. *Gesammelte Schriften in 20 Bänden: Band 10: Kulturkritik und Gesellschaft. Prismen. Ohne Leitbild. Eingriffe. Stichworte.* Suhrkamp Verlag, Frankfurt am Main, 2003.

[4] J. Habermas. *Strukturwandel der Öffentlichkeit.* Suhrkamp Verlag, Frankfurt am Main, 1962.

[5] E. T. Hall. *The Hidden Dimension.* 1966.

[6] B. Hans. *Privatheit – und Öffentlichkeit*, pages 35–117. Springer Fachmedien Wiesbaden, Wiesbaden, 2017.

[7] P. Hölzing. Öffentlichkeit und Privatheit. *dis|kurs*, 8(1), 2012.

[8] I. Kant. Beantwortung der Frage: Was ist Aufklärung? *Berlinische Monatsschrift*, 4(12):481–494, 1784.

[9] R. Luxemburg. *Die Akkumulation des Kapitals. Ein Beitrag zur ökonomischen Erklärung des Imperialismus.* 1913.

[10] K. Marx. *Das Kapital, Erster Band: Der Produktionsprozeß des Kapitals.* 1867.

[11] H. Nissenbaum. Privacy as contextual integrity. *Washington Law Review*, pages 101–139, 2004.

[12] T. Noetzel. Am Anfang Kant, am Ende Adorno? Zum philosophisch-politischen Diskurs über Öffentlichkeit und Privatheit. *Zeitschrift für Bürgerrechte und Gesellschaftspolitik*, 37(4):39–48, 1998.

[13] M. Rost. Zur Soziologie des Datenschutzes. *Datenschutz und Datensicherheit*, 37(2):85–91, 2013.

[14] M. Rost. Was meint eigentlich Datenschutz? *Der Landkreis, Zeitschrift für kommunale Selbstverwaltung*, 2014.

[15] M. Rost. Bob, es ist Bob! *FiFF-Kommunikation*, 2017.

[16] M. Rost and A. Pfitzmann. Datenschutz-Schutzziele – revisited. *Datenschutz und Datensicherheit*, 6:353–358, 2009.

[17] S. D. Warren and L. D. Brandeis. The Right to Privacy. *Harvard Law Review*, 4(5):pp. 193–220, 1890.

[18] A. F. Westin. *Privacy and Freedom.* 1967.

Analyse der Ende-zu-Ende-Verschlüsselung von Nextcloud

Emmanuel Syrmoudis
Betreuer: Dr. Holger Kinkelin
Seminar Future Internet SS2018
Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: emmanuel.syrmoudis@tum.de

KURZFASSUNG

Diese Seminararbeit untersucht die Ende-zu-Ende-Verschlüsselung (E2EE) der Filehosting-Software Nextcloud. Nextcloud ermöglicht das Selfhosting von Daten und damit Unabhängigkeit von Filehosting-Anbietern wie Dropbox. Die E2EE soll dabei sicherstellen, dass vertrauliche Daten nur auf Endgeräten entschlüsselt werden können. Im Mittelpunkt der Arbeit steht die Sicherheit der E2EE. Zu ihrer Analyse wird zunächst die Implementierung sowie algebraische, logische, Tool-basierte und informelle Analysemethoden vorgestellt und anschließend eine informelle Analyse der E2EE durchgeführt. Dabei wird insbesondere eine Schwachstelle beim Schlüsselaustausch aufgezeigt, mit der die Schutzziele Vertraulichkeit, Integrität und Authentizität verletzt werden können. Abschließend werden mögliche Verbesserungsvorschläge gemacht.

Schlüsselworte

Nextcloud, Ende-zu-Ende-Verschlüsselung, Cloud, Filehosting, Selfhosting

1. EINLEITUNG

Die moderne Gesellschaft wird immer digitaler, Menschen auf der ganzen Welt vernetzen sich über das Internet und stellen ihre Daten in die „Cloud“. Urlaubsphotos landen auf Facebook oder Instagram, Gedanken werden auf Twitter oder Snapchat geteilt, Termine über Doodle vereinbart, E-Mails über GMail verschickt und Dokumente auf Dropbox gespeichert. Auch in die Geschäftswelt und in den Bildungssektor hat die Digitalisierung Einzug gehalten, so wird beispielsweise Slack zur Kommunikation innerhalb von Arbeitsgruppen genutzt und offizielle Diskussionsforen zu manchen Kursen der TUM werden auf Piazza eingerichtet.

All diese Beispiele haben eine Gemeinsamkeit: Die Daten der Nutzer dieser Dienste werden auf den Servern des jeweiligen Anbieters gespeichert. Dort erwecken sie natürlich Begehrlichkeiten von mehreren Seiten. So könnten Staaten und deren Geheimdienste Anspruch auf diese Daten erheben, sei es zur Terrorbekämpfung oder zur Bewertung von Personen im chinesischen Sozialkredit-System [13]. Auch für Werbekunden sind die Daten interessant, ermöglichen sie doch eine zielgerichtetere Ausspielung von Werbeanzeigen. Gerade für Firmen wie Facebook und Google, die einen Großteil ihrer Dienste kostenlos anbieten, ist dies ein wesentlicher Bestandteil des Geschäftsmodells.

Da die Datennutzung für den einzelnen Nutzer meist keine unmittelbar spürbaren negativen Konsequenzen hat, wird diese oft toleriert. Dass sie aber eine Auswirkung auf die gesamte Gesellschaft haben kann, zeigte sich beispielsweise im US-amerikanischen Präsidentschaftswahlkampf 2016. Damals hatte die Datenanalysefirma Cambridge Analytica, im Auftrag der Kampagne des republikanischen Bewerbers Donald Trump, Werbung auf Facebook geschaltet. Hierbei nutzte sie detaillierte Daten von ca. 50 Millionen Facebook-Nutzern um als potentielle Wähler identifizierte Nutzer zu mobilisieren und Nutzer, die eher den Demokraten zugeneigt waren, zu demobilisieren [5]. Schlussendlich gewann der republikanische Kandidat.

Beispiele wie dieses zeigen, wie wichtig es ist, persönliche Daten nicht einfach unüberlegt fremden Anbietern zu überlassen. Eine Alternative hierzu ist *Selfhosting*, das Speichern der Daten auf eigenen Servern statt bei fremden Anbietern. Diese Seminararbeit befasst sich mit Nextcloud, einer Selfhosting-Alternative zu Filehosting-Anbietern wie Dropbox oder Onedrive. Nextcloud kann auf eigenen Servern installiert werden und ermöglicht unter anderem das Hochladen von Dateien, das Synchronisieren zwischen mehreren Geräten und das Teilen mit anderen Nutzern.

Statt auf eigenen Servern, bei denen man vollen und exklusiven Zugriff auf die Hardware hat, kann Nextcloud auch auf angemieteten Servern in einem fremden Rechenzentrum betrieben werden. Dies ist vor allem für kleinere Unternehmen interessant, hat aber wiederum den Nachteil, dass die Daten in fremde Hände gelangen. Hier schafft die Ende-zu-Ende-Verschlüsselung (E2EE) Abhilfe, die Nextcloud in Version 13 optional eingeführt hat [4] und die Hauptthema dieser Arbeit ist. Inhalte aus Ordnern bei denen die E2EE aktiv ist, können nur von berechtigten Nutzern eingesehen werden, nicht aber vom Serverbetreiber oder sonstigen Dritten. Die Ver- und Entschlüsselung der Daten erfolgt direkt auf den Endgeräten, wo auch die nötigen Schlüssel gespeichert sind.

Damit unterscheidet sich die E2EE auch von der ebenfalls von Nextcloud angebotenen serverseitigen Verschlüsselung. Bei dieser werden die Dateien auf dem Server ver- und entschlüsselt, der eingesetzte Schlüssel wird ebenfalls auf dem Server gespeichert. Das ist sinnvoll, wenn externe Speicher eingebunden werden, bietet aber keinen Schutz der Daten vor dem Betreiber des Servers. Bei der serverseitigen Verschlüsselung muss der Server also vertrauenswürdig sein, eine wirksame Ende-zu-Ende-Verschlüsselung setzt dieses Ver-

doi: 10.2313/NET-2018-11-1_10

trauen nicht voraus.

Die Ende-zu-Ende-Verschlüsselung ist auch dann sinnvoll, wenn man Nextcloud auf eigenen Servern betreibt und sensible Daten, wie etwa Firmengeheimnisse, vor einer Kompromittierung des Servers durch etwaige Sicherheitslücken schützen will.

Ziel dieser Seminararbeit ist es, eine Bewertung der Sicherheit der Nextcloud-E2EE abzugeben. Hierfür werden in Kapitel 2 zunächst zentrale Elemente der Implementierung und deren Funktionsweise vorgestellt. Anschließend werden in Kapitel 3.1 ein Angreifermodell und Schutzziele aufgeführt, anhand derer eine formalere Einordnung der Qualität von kryptographischen Protokollen möglich ist. Mehrere Methoden, mit denen kryptographische Protokolle analysiert werden können, präsentiert Kapitel 3.2. Kapitel 4 widmet sich dann der konkreten Analyse der Nextcloud-E2EE, mögliche Verbesserungsvorschläge zeigt Kapitel 5 auf. Abschließend verweist Kapitel 6 auf alternative Softwarelösungen; Kapitel 7 fasst die Ergebnisse der Arbeit zusammen.

2. IMPLEMENTIERUNG DER NEXTCLOUD-E2EE

Dieser Abschnitt beschreibt die zentralen Elemente der Ende-zu-Ende-Verschlüsselung von Nextcloud. Grundlage hierfür sind die Ausführungen im von Nextcloud veröffentlichten Whitepaper in der Version vom 20. September 2017 [4].

Die Nextcloud-E2EE stützt sich auf zwei zentrale Komponenten: Public-Key-Kryptographie bei der der Server als Zertifizierungsstelle fungiert und eine Metadaten-Datei, die für die Verschlüsselung auf Orderebene eingesetzt wird.

2.1 Public-Key-Kryptographie

Da das E2EE-Protokoll es ermöglichen soll, Dateien mit anderen Nutzern zu teilen, setzt Nextcloud Public-Key-Kryptographie ein. Damit ist es im Gegensatz zu symmetrischer Kryptographie möglich, dass Nutzerin Alice Daten verschlüsselt, die Nutzer Bob mit seinem Schlüssel entschlüsseln kann, ohne dass Alice und Bob vorher ein gemeinsames Geheimnis festlegen müssen. Alice verschlüsselt die Daten mit Bobs öffentlichem Schlüssel, Bob entschlüsselt sie anschließend mit seinem privaten Schlüssel.

Verwendet Alice erstmals die Nextcloud-E2EE, so erzeugt die Nextcloud-Anwendung auf ihrem Gerät ein Public-Key-Paar. Zudem erzeugt die Anwendung ein Passwort, mit dem der *private key* verschlüsselt wird. Dieses Passwort hat die Form eines *mnemonics*, mit einer Länge von 12 Worten, die aus einer Liste von 2048 Worten zufällig ausgewählt werden. Es hat damit eine Entropie von 132 bit und könnte beispielsweise so aussehen: „ghost eight waste vicious copper dizzy lonely bench lava slab split forward“.

Wie Abbildung 1 veranschaulicht, wird der mit dem mnemonic als Passwort unter Verwendung des AES-Algorithmus verschlüsselte *private key* auf den Server hochgeladen. Sobald Alice Nextcloud auf einem anderen Gerät verwendet, lädt dieses den verschlüsselten *private key* herunter, fragt Alice nach dem mnemonic und speichert den entschlüsselten *private key* anschließend auf dem Gerät.

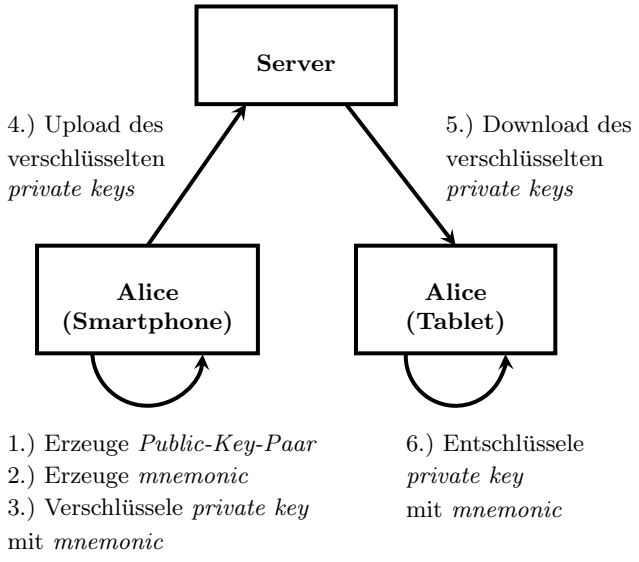


Abbildung 1: Austausch des *private keys* zwischen mehreren Endgeräten

Damit Alice ihren Ordner mit Bob teilen kann, benötigt sie Bobs *public key*. Um die öffentlichen Schlüssel zu verteilen, bietet der Server diese zum Download an und nimmt zusätzlich die Funktion einer Zertifizierungsstelle (CA) ein. Dieser Vorgang wird in Abbildung 2 dargestellt. Nachdem Alice ihr Schlüsselpaar erzeugt hat, erstellt sie einen Certificate Signing Request (CSR). Der CSR enthält Alice' *public key* und ihre User ID. Sie sendet den CSR an den Server, der dann ein X.509-Zertifikat [9] ausstellt.

Um ihren Ordner mit Bob zu teilen, lädt sie also dessen Zertifikat vom Server herunter, prüft die Signatur des Zertifikats und verwendet ab jetzt Bobs *public key*. Dabei vertraut sie darauf, dass der Server zum Zeitpunkt des Downloads des Zertifikats nicht kompromittiert ist.

2.2 Metadaten-Datei

Nachdem E2EE vom Serveradministrator aktiviert wurde, können die Nutzer auf Orderebene entscheiden, ob der komplette Ordner verschlüsselt werden soll. Entscheidet sich Alice für die Verschlüsselung eines Ordners, wird darin eine Metadaten-Datei im JSON-Format angelegt, in der die verschlüsselten Dateien verwaltet werden. Abbildung 3 zeigt den schematischen Aufbau dieser Datei, Tabelle 1 gibt einen Überblick über die verwendeten Schlüssel.

Der Kopf der Metadaten-Datei enthält eine Liste von *metadataKeys*. Der erste *metadataKey* wird beim Anlegen der Datei erzeugt und mit dem *public key* von Alice verschlüsselt. Hierbei wird der RSA-Algorithmus verwendet. Teilt Alice den Ordner mit Bob und Charlie, so wird die Liste der *metadataKeys* zusätzlich auch mit den *public keys* von Bob und Charlie verschlüsselt und kann somit sowohl von Alice, von Bob, als auch von Charlie entschlüsselt werden. Entschließt sich Alice dazu, den Ordner nicht mehr mit Charlie zu teilen, erzeugt sie einen neuen *metadataKey*, hängt diesen an die Liste der *metadataKeys* an und verschlüsselt die

doi: 10.2313/NET-2018-11-1_10

Tabelle 1: Übersicht der eingesetzten Schlüssel und Passwörter

Bezeichnung	Aufgabe	Algorithmus
encryptionKey	Verschlüsselung des Inhalts einer einzelnen Datei	AES/GCM/NoPadding
metadataKey	Verschlüsselung der geschützten Inhalte in der Metadaten-Datei	AES/GCM/NoPadding
Public-Key-Paar	Verschlüsselung der metadataKeys	RSA/ECB/OAEP+Padding
mnemonic	Passwort zur Verschlüsselung des private keys auf dem Server	AES/GCM/NoPadding

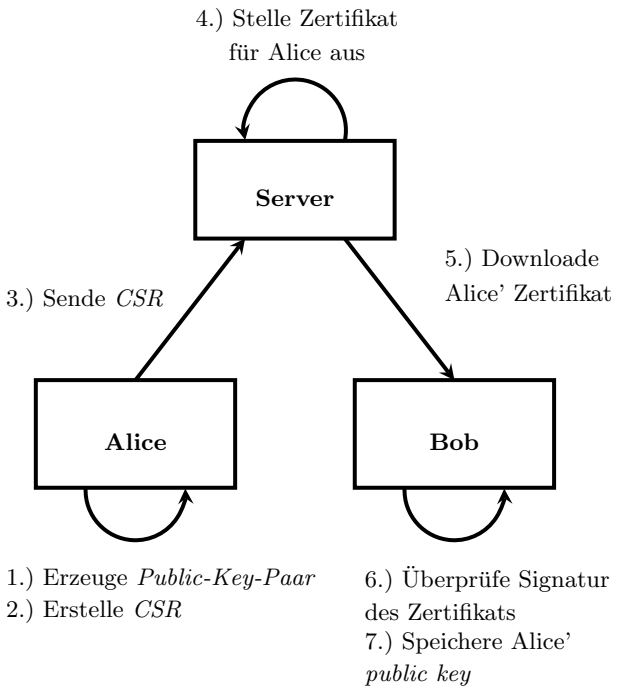


Abbildung 2: Austausch der *public keys* zwischen mehreren Nutzern

```
{
  "metadata": {
    // Liste der metadataKeys (RSA-verschluesselt)
    "metadataKeys": {
      "0": "aeltester metadataKey",
      "1": "neuester metadataKey"
    },
    "sharing": {
      // Public Keys aller berechtigten Nutzer (AES-verschluesselt mit neuestem metadataKey)
      "recipient": {
        "alice": "PUBLIC KEY",
        "bob": "PUBLIC KEY"
      },
    },
    "files": {
      // UUID der verschluesselten Datei
      "ioDuSxIfa...": {
        // verwendeter metadataKey
        "metadataKey": 1,
        // geschuetzte Daten (AES-verschluesselt mit angegebenem metadataKey)
        "encrypted": {
          // encryptionKey mit dem die Datei verschluesselt wurde
          "key": "...",
          "filename": "nuclearcodes.txt",
          "mimetype": "plain/text"
        },
        "initializationVector": "...",
        // Galois/Counter Mode Authentication Tag
        "authenticationTag": "..."
      }
      // analog dazu weitere Dateien...
    }
  }
}
```

Abbildung 3: Verkürzte Darstellung der Metadaten-Datei

Liste nur noch mit ihrem und Bobs *public key*.

Außer den *metadataKeys* enthält die Metadaten-Datei eine Liste der *public keys* aller berechtigten Nutzer (in obigem Beispiel Alice und Bob), sowie eine Liste der Dateien, die sich in dem Ordner befinden. Jede Datei wird durch eine zufällige Zeichenfolge (UUID) identifiziert. Ihr zugeordnet sind ein *encryptionKey*, der Dateiname, der MIME-Type, ein Initialisierungsvektor, ein Galois/Counter Mode Authentication Tag, sowie ein Verweis auf einen der *metadataKeys*. *encryptionKey*, Dateiname und MIME-Type sind dabei mit dem angegebenen *metadataKey* AES-verschlüsselt, die restlichen Elemente sind im Klartext einsehbar.

Bei der Verschlüsselung der einzelnen Dateien kommt AES als Blockchiffre mit Galois/Counter Mode (GCM) als Betriebsmodus zum Einsatz. GCM ermöglicht authentifizierte Verschlüsselung, wodurch sowohl Vertraulichkeit, als auch Integrität und Authentizität erreicht werden können [14]. Integrität und Authentizität können dabei mit Hilfe des GCM Authentication Tag überprüft werden.

Da RSA ausschließlich zur Verschlüsselung der *metadataKeys* mit den *public keys* verwendet wird und für alle anderen Verschlüsselungsvorgänge AES zum Einsatz kommt, wird die Verwendung teurer asymmetrischer Verschlüsselung auf ein Mindestmaß beschränkt und ansonsten effizient anwendbare symmetrische Verschlüsselung genutzt.

doi: 10.2313/NET-2018-11-1_10

Lesen einer Datei. Will ein berechtigter Nutzer, beispielsweise Bob, den Inhalt einer Datei lesen, benötigt er dazu den *encryptionKey*, mit dem der Inhalt dieser Datei verschlüsselt wurde. Um diesen zu erhalten, öffnet er zunächst die Metadaten-Datei. Nun kann Bob die Liste der *metadataKeys* mit seinem *private key* entschlüsseln. Im „files"-Abschnitt der Metadaten-Datei sucht er dann nach der gewünschten Datei.

Kennt er die UUID der Datei bereits, kann er einfach zur gewünschten Datei springen und dort in Erfahrung bringen, mit welchem *metadataKey* die geschützten Metadaten der Datei verschlüsselt wurden. Er entschlüsselt sie mit dem passenden *metadataKey* und kennt nun den *encryptionKey* mit dem der Inhalt der Datei verschlüsselt wurde. Diesen *encryptionKey* sowie den Initialisierungsvektor nutzt er schlussendlich um die Datei zu entschlüsseln. Zudem kann er mit dem GCM Authentication Tag Integrität und Authentizität der Datei prüfen. Falls er die UUID der Datei noch nicht kennt, erstellt er zunächst eine Auflistung aller Dateien, die sich im Ordner befinden indem er auf dieselbe Weise die Dateinamen aller Dateien entschlüsselt.

Schreiben einer Datei. Zum Schreiben einer Datei müssen analog zu oben zunächst die *metadataKeys* in Erfahrung gebracht werden. Die Nutzerin, die die Datei schreiben will, hier Alice, generiert dann einen zufälligen *encryptionKey*, sowie einen zufälligen Initialisierungsvektor und verschlüsselt die Datei mit AES unter Nutzung des Galois/Counter Mode. Anschließend lädt sie die verschlüsselte Datei auf den Server hoch und aktualisiert bzw. erstellt den entsprechenden Eintrag in der Metadaten-Datei. Zur Verschlüsselung des *encryptionKeys*, des Dateinamen und des MIME-Types nutzt sie den neuesten *metadataKey*.

3. GRUNDLAGEN UND ANALYSEMETHODEN

3.1 Angreifermodell und Schutzziele

Vor dem Entwurf oder der Analyse kryptographischer Protokolle oder Systeme ist es sinnvoll, einige formale Annahmen über die Fähigkeiten eines möglichen Angreifers zu treffen, sowie Ziele festzulegen, die das System erfüllen soll.

Ein weit verbreitetes Angreifermodell ist das *Dolev-Yao-Modell* [11]. In diesem Modell kontrolliert der Angreifer das Netzwerk und kann Nachrichten senden, empfangen, abfangen und modifizieren. Dabei kann er jegliche Teile einer Nachricht verändern, beispielsweise auch die Identität des Absenders. Zur Analyse der Nextcloud Ende-zu-Ende-Verschlüsselung bietet es sich an, das Angreifermodell um die Annahme zu erweitern, dass der Angreifer den Server kontrolliert. Somit kann er, analog zum Dolev-Yao-Modell, Dateien auf dem Server lesen, erstellen, löschen und verändern.

In der Regel unterscheidet man folgende Schutzziele [12]:

Vertraulichkeit Nur berechtigte Nutzer können den Inhalt einer Datei lesen. Im Kontext des eben festgelegten Angreifermodells wäre dieses Schutzziel beispielsweise verletzt, wenn der Angreifer über eine Sicherheitslücke an einen der eingesetzten Schlüssel gelangt und so den Dateiinhalt einer verschlüsselten Datei lesen kann.

Integrität Unberechtigten Nutzern ist es nicht möglich, unbemerkt den Inhalt einer Datei zu verändern. In unserem Angreifermodell hat der Angreifer die Möglichkeit Dateien zu verändern. Ist das Schutzziel erfüllt, können berechtigte Nutzer jedoch feststellen, dass eine unautorisierte Änderung vorgenommen wurde. Manchmal wird *Autorisierung* auch als separates Schutzziel betrachtet.

Authentizität Die Echtheit und Überprüfbarkeit einer Nachricht oder eines Dokuments ist sichergestellt. Nutzer können also überprüfen, ob eine Datei tatsächlich von einem berechtigten Nutzer bearbeitet wurde.

Verfügbarkeit Die angebotenen Dienste sind verfügbar und in ihrer Nutzbarkeit nicht eingeschränkt. Angriffe auf dieses Schutzziel werden als *Denial of Service* bezeichnet und beabsichtigen beispielsweise eine Nicht-Erreichbarkeit des Servers aus dem Internet.

Verbindlichkeit Es ist nicht möglich, durchgeführte Handlungen abzustreiten. Ändert Benutzerin Alice beispielsweise eine Datei, so kann sie gegenüber Benutzer Bob nicht glaubwürdig abstreiten, diese Änderung vorgenommen zu haben.

Privatheit Die Privatsphäre und die personenbezogenen Daten der Nutzer werden geschützt. Wie dieses Schutzziel umzusetzen ist, ist oftmals in Datenschutzgesetzen festgelegt.

Nextcloud selbst gibt an, dass die E2EE die Schutzziele Vertraulichkeit, sowie Authentizität und Integrität erfüllen soll [4], wobei Integrität und Authentizität hier einhergehen. Ein Angreifer soll also keine Möglichkeit haben, verschlüsselte Dateiinhalte im Klartext zu lesen und Dateien nicht unbemerkt verändern können.

Privatheit wird im Rahmen dieser Arbeit alleinig als die Privatheit der Daten der Nutzer aufgefasst und damit auch vom Schutzziel Vertraulichkeit abgedeckt. Um eine umfassende Privatheit zu gewährleisten, sind Maßnahmen erforderlich, die über den bloßen Einsatz einer E2EE hinausgehen, da hierbei beispielsweise auch der Umgang mit Metadaten berücksichtigt werden muss. Die übrigen Schutzziele liegen nicht im typischen Aufgabenbereich einer Ende-zu-Ende-Verschlüsselung und werden daher in dieser Arbeit nicht behandelt.

3.2 Analysemethoden

Nachdem ein Angreifermodell und die angestrebten Schutzziele festgelegt wurden, verbleibt die Auswahl einer passenden Analysemethode. Dieser Abschnitt stellt einige dieser Methoden kurz vor. Einen tieferen Einblick geben beispielsweise [15] oder [8].

3.2.1 Algebraische Analyse

Die aufwendigste, aber auch zielführendste Methode ist die algebraische Analyse. Hierbei wird das Protokoll oder System in ein formales algebraisches Modell überführt und die Eigenschaften dieses Modells mathematisch bewiesen.

doi: 10.2313/NET-2018-11-1_10

3.2.2 Modallogik

Besonders in den 1990er Jahren verbreitet, war die Anwendung von Modallogik zur Analyse von kryptographischen Protokollen. Die meistverwendete dieser Logiken ist die BAN-Logik [7]. Ausgehend von bestimmten Annahmen („Alice glaubt, dass nur Bob und Charlie Schlüssel X kennen“) und der gesendeten Nachrichten, werden dann wiederum neue Annahmen gebildet. Die BAN-Logik kann nur Aussagen zur Authentizität treffen. Mit der GNY-Logik besteht eine Weiterentwicklung, die allerdings deutlich komplizierter ist und deshalb kaum genutzt wird. [15]

3.2.3 Tool-basierte Analyse

Abgesehen von der „händischen“ Analyse, können kryptographische Protokolle auch mit der Hilfe von Tools analysiert werden. Zwei der bekanntesten Analysetools sind AVISPA [6] und Scyther [10]. Scyther beispielsweise definiert eine eigene Sprache, in der Protokolle modelliert werden können. Nachdem dies geschehen ist, analysiert Scyther dieses Modell und trifft Aussagen über die Erfüllung der Schutzziele, sowie über mögliche Angriffe.

3.2.4 Informelle Analyse

Deutlich weniger aufwendig als algebraische Analysen, dafür auch mit geringerer Aussagekraft, sind informelle Analysemethoden. Sie sind geeignet, um sich einen Überblick über mögliche Schwachstellen des Protokolls oder Systems zu verschaffen und Angriffsziele zu identifizieren. Auch wenn das untersuchte System zu komplex ist um es formal zu modellieren, können stattdessen informelle Methoden zur Analyse eingesetzt werden. Dazu können beispielsweise die Komponenten des Systems einzeln untersucht und Annahmen darüber getroffen werden, auf welche Weisen ein Angreifer eines der Schutzziele verletzen könnte.

Ein mögliches Hilfsmittel hierbei ist das Erstellen von „Attack Trees“ [16]. Damit kann modelliert werden, was zur erfolgreichen Ausführung eines Angriffs nötig ist und folglich eine Einschätzung der Erfolgswahrscheinlichkeit vorgenommen werden. Das Angriffsziel ist hierbei der Wurzelknoten und die möglichen Wege, das Ziel zu erreichen, dessen Kinder. Diese Kindknoten können selbst ebenfalls Kinder haben. Schneier [16] wählt als Beispiel das Öffnen eines Tresors. Dazu kann beispielsweise das Schloss geknackt werden oder die Kombination in Erfahrung gebracht werden. Um die Kombination herauszufinden, kann sie in schriftlicher Form gefunden werden oder dem Opfer selbst entlockt werden, ... Findet man schließlich einen realistischen Weg, so ist auch der Angriff als solcher durchführbar.

4. ANALYSE DER NEXTCLOUD-E2EE

In Abschnitt 3.1 wurde ein Angreifermodell definiert und die Schutzziele Vertraulichkeit, Authentizität und Integrität festgelegt, dieser Abschnitt untersucht nun, ob die eben vorgestellte Implementierung der Nextcloud E2EE diese Schutzziele erfüllt.

Das Angreifermodell besagt, dass der Angreifer sowohl das Netzwerk als auch den Server kontrolliert. Da, wie in Abschnitt 2 gesehen, der Server sowieso entweder Empfänger oder Sender aller Nachrichten ist (es findet keine direkte Kommunikation zwischen mehreren Nutzern oder den Endgeräten eines Nutzers statt), richtet sich das Augenmerk hier auf das Lesen, Schreiben, Löschen oder Modifizieren von Dateien auf dem Server durch den Angreifer. Zur Kommunikation über das Netzwerk sei hier noch angemerkt, dass alle Nachrichten über TLS gesendet werden [4].

Für die angestrebten Schutzziele gilt es nun, eine geeignete Analysemethode aus den in Abschnitt 3.2 vorgestellten Methoden auszuwählen. Da insbesondere auch das Schutzziel Vertraulichkeit von Interesse ist, ist die BAN-Logik hierfür nicht geeignet. Ideal wäre eine algebraische Analyse, die aber im Rahmen einer Seminararbeit nicht zu leisten ist. Deshalb beschränkt sich die Analyse auf eine informelle Untersuchung der Lese- und Schreibvorgänge und der Möglichkeiten, die sich daraus für den Angreifer ergeben.

Die Attack Trees in den Abbildungen 4 und 5 stellen Möglichkeiten dar, wie der Angreifer verschlüsselte Dateien lesen (Verletzung des Schutzziels Vertraulichkeit), bzw. schreiben (Verletzung der Schutzziele Integrität und Authentizität) kann. Da alle Schlüssel und Blockgrößen größer oder gleich 128 bit sind, kann Brute-Force generell als nicht möglich betrachtet werden. Die folgenden Abschnitte analysieren, ob der Angreifer anderweitige Möglichkeiten hat, an einen der Schlüssel zu gelangen.

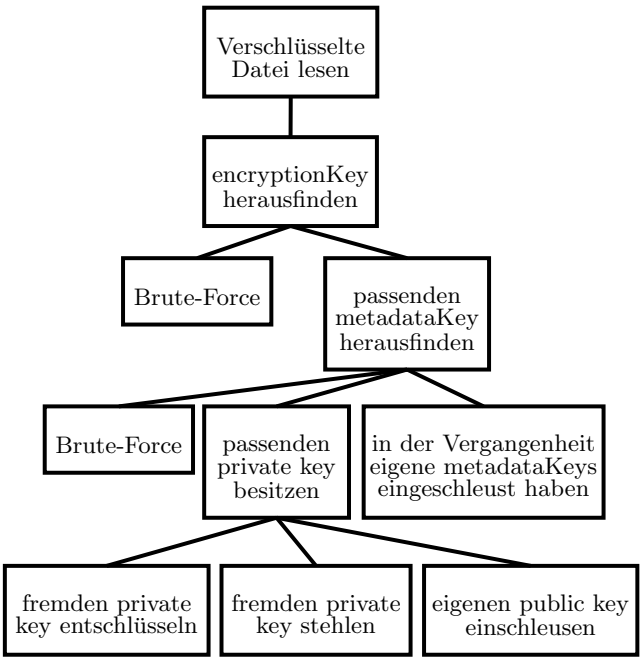


Abbildung 4: Attack Tree für das Lesen einer Datei

4.1 Schlüsselaustausch

Ein möglicher Angriffspunkt ist der in Abschnitt 2.1 und in den Abbildungen 1 und 2 beschriebene Schlüsselaustausch zwischen mehreren Endgeräten eines Nutzers bzw. zwischen mehreren Nutzern.

Betrachten wir zunächst den Fall der Übertragung des *private keys* auf ein weiteres Endgerät (Endgerät 2) eines Nutzers. Der *private key* wurde mit dem *mnemonic* als Passwort auf Endgerät 1 verschlüsselt (unter Verwendung von

doi: 10.2313/NET-2018-11-1_10

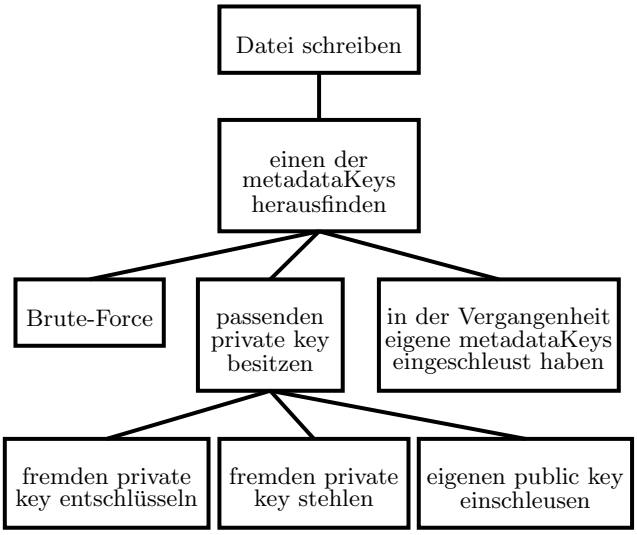


Abbildung 5: Attack Tree für das Schreiben einer Datei

AES-GCM) und anschließend auf dem Server gespeichert. Das *mnemonic* verbleibt ausschließlich beim Nutzer. Da das *mnemonic* eine Entropie von 132 bit hat ($12 \cdot 11$: 12 zufällig ausgewählte Wörter aus einer Liste von $2^{11} = 2048$ Wörtern) und die verwendete Blocklänge 128 bit beträgt, kann der Angreifer den *private key* nicht mittels Brute-Force im Klartext erlangen. Weiter hat der Angreifer die Möglichkeit, dem Endgerät 2 eine modifizierte Version des verschlüsselten *private keys* zu senden, beispielsweise einen *private keys*, den der Angreifer selbst erzeugt hat. Da zur Verschlüsselung GCM eingesetzt wurde, kann das Endgerät 2 die Verletzung der Integrität aber erkennen und den Schlüssel folglich ablehnen. Der Angreifer kann somit lediglich verhindern, dass das Endgerät 2 den *private key* erhält, was aber weder die Vertraulichkeit der Daten, noch deren Integrität oder Authentizität gefährdet.

Anders sieht es im Fall der Übertragung des *public keys* zu anderen Nutzern aus. Hier übernimmt der Server die Funktion einer CA, stellt also Zertifikate für einzelne Nutzer aus. Will Alice den *public key* von Bob abrufen, kann der Angreifer einfach ein Zertifikat mit seinem eigenen *public key*, aber der Identität von Bob erstellen. Teilt Alice dann einen Ordner mit Bob, kann der Angreifer die Liste der *metadataKeys* mit seinem *private key* entschlüsseln und hat folglich Zugriff auf alle Dateien in diesem Ordner.

Anzumerken ist, dass im Falle der Zertifikate „Trust On First Use (TOFU)“ gilt, das Zertifikat eines neuen Nutzers also nur einmal abgefragt und diesem dann vertraut wird. Ist ein Server zum Zeitpunkt der Abfrage vertrauenswürdig und wird erst später kompromittiert, hat der Angreifer hier keine Möglichkeit mehr, seinen eigenen *public key* einzuschleusen.

4.2 Dateioperationen

Betrachten wir nun verschiedene Dateioperationen. Der Angreifer ist in der Lage, Dateien zu lesen. Ist er nicht im Besitz des passenden *encryptionKeys*, kann er den Dateiinhalt nicht entschlüsseln, die Vertraulichkeit des Inhalts bleibt also gewahrt. Des weiteren ist der Angreifer in der Lage, neue Dateien anzulegen oder bestehende Dateien zu modifizieren. Den Inhalt der Datei, sowie Initialisierungsvektor und den *encryptionKey* kann er frei wählen und den GCM Authentication Tag selbst berechnen. Um insbesondere den GCM Authentication Tag in die Metadaten-Datei einzutragen, muss der Angreifer in Besitz mindestens eines *metadataKeys* sein. Ist dies nicht der Fall, so können Nutzer die Modifikation anhand des fehlerhaften Authentication Tags erkennen und die Schutzziele Integrität und Authentizität bleiben gewahrt.

Der Angreifer muss also in Besitz des *encryptionKeys* einer verschlüsselten Datei oder eines *metadataKeys* kommen, um eines der festgelegten Schutzziele verletzen zu können.

Im Besitz des *encryptionKeys* ist zunächst der Nutzer, der die Datei zuletzt gespeichert hat. Zudem ist der *encryptionKey* mit dem zum Speicherzeitpunkt neuesten *metadataKey* verschlüsselt in der Metadaten-Datei abgelegt. Dieser *metadataKey* ist mit den *public keys* aller zu diesem Zeitpunkt berechtigten Nutzer verschlüsselt in der Metadaten-Datei abgelegt. Zudem wird er beim Hinzufügen eines neuen Nutzers auch mit dessen *public key* verschlüsselt. Zugriff auf den Dateiinhalt haben also alle zum Speicherzeitpunkt berechtigten Nutzer sowie alle später hinzugefügten Nutzer. Andere Nutzer (wie der Angreifer) können keinen Zugriff auf den Dateiinhalt erlangen, das Schutzziel Vertraulichkeit ist erfüllt.

metadataKeys werden immer vom Besitzer des Ordners erzeugt. Außer dem Besitzer kennen Benutzer, die zu irgendeinem Zeitpunkt zur Menge der berechtigten Nutzer gehört haben, zumindest den ältesten *metadataKey*. Andere Nutzer haben keine Möglichkeit, Kenntnis eines *metadataKeys* zu erlangen. War der Angreifer nie in der Menge der berechtigten Nutzer, sind die Schutzziele Integrität und Authentizität erfüllt. Auch von einem Benutzer, der aus der Menge der berechtigten Nutzer entfernt wurde, würde man erwarten, dass er diese Schutzziele nicht verletzen kann. Dies ist aber nicht der Fall, da der Angreifer Einträge in der Metadaten-Datei auch mit einem alten *metadataKey* verschlüsseln kann.

4.3 Ersetzen der metadataKeys

Wie bereits in Abschnitt 2.2 beschrieben, werden die *metadataKeys* mit den *public keys* aller berechtigten Nutzer RSA-verschlüsselt in der Metadaten-Datei gespeichert. Da alle *public Keys* unverschlüsselt auf dem Server gespeichert werden (siehe Abschnitt 2.1), kennt diese auch der Angreifer. Zudem werden die Namen der Ordner und die Liste der Nutzer mit denen ein Ordner geteilt wurde unverschlüsselt auf dem Server gespeichert, da die Verwaltung der Ordner außerhalb der E2EE-App erfolgt [4].

Der Angreifer kennt also die berechtigten Nutzer sowie deren *public keys*. Er kann eigene *metadataKeys* erzeugen und diese mit den passenden *public keys* verschlüsseln. Anschließend kann er die ursprünglichen *metadataKeys* in der Metadaten-Datei durch die eben erzeugten ersetzen. Befinden sich bereits im Dateien im Ordner, werden diese durch das Ersetzen der *metadataKeys* unlesbar. Um seinen Angriff zu verschleiern, kann der Angreifer die vorhandenen Dateien löschen und eigene Dateien darin ablegen, die zum Namen des Ord-

doi: 10.2313/NET-2018-11-1_10

ners passen.

Bleibt der Angriff unentdeckt, kann der Angreifer fortan alle in diesem Ordner gespeicherten Dateien lesen und schreiben und damit die Schutzziele Vertraulichkeit, Integrität und Authentizität verletzen.

4.4 Einspielen alter Dateiversionen

In der Metadaten-Datei befindet sich kein Zeitstempel, der angibt, wann sie zuletzt bearbeitet wurde. Der Angreifer hat daher die Möglichkeit, Metadaten-Datei und Ordnerinhalt durch alte Versionen zu ersetzen, ohne dass dies von den Endgeräten erkannt werden kann.

5. VERBESSERUNGSVORSCHLÄGE

In Abschnitt 4 wurden mehrere Schwachstellen in der Ende-zu-Ende-Verschlüsselung von Nextcloud identifiziert. Dieser Abschnitt soll Verbesserungsvorschläge aufzeigen, mit denen sie beseitigt oder zumindest abgemildert werden können.

Das größte Problem in der Implementierung der Nextcloud-E2EE liegt im Schlüsselaustausch der *public keys*. Hier übernimmt der Server gleichzeitig die Funktion einer Zertifizierungsstelle und muss daher beim erstmaligen Teilen von Ordnern mit neuen Nutzern vertrauenswürdig sein.

Hier sollte man die Möglichkeit schaffen, optional andere CAs zu verwenden. Zur Überprüfung der Zertifikate könnte man dann beispielsweise den Stammspeicher des Betriebssystems verwenden. Des weiteren sollte man Nutzern ermöglichen, fremde *public keys* von Hand zu überprüfen. Dies könnte beispielsweise durch einen QR-Code geschehen, der in der Nextcloud-App angezeigt wird und den andere Nutzer dann persönlich mit ihrer Nextcloud-App scannen können. Auch die Anzeige eines Fingerabdrucks des Public Keys, der dann über andere Kanäle (z.B. telefonisch) abgeglichen werden kann, wäre denkbar.

Ein weiteres Problem liegt darin begründet, dass die Integrität und Authentizität der Metadaten-Datei nicht sichergestellt wird. Dadurch kann der Angreifer sowohl die *metadataKeys* durch selbst erstellte Schlüssel ersetzen, als auch alte *metadataKeys* zum erstellen und bearbeiten von Einträgen in der Metadaten-Datei nutzen.

Um das Ersetzen der *metadataKeys* erkennen zu können, sollte der Besitzer des Ordners die Liste der *metadataKeys* mit seinem *private key* signieren. Anhand dieser Signatur kann dann auf den Endgeräten geprüft werden, ob die *metadataKeys* wirklich vom Besitzer des Ordners erzeugt wurden.

Zudem sollten diejenigen Abschnitte der Metadaten-Datei, die von allen berechtigten Nutzern bearbeitet werden dürfen, mit dem neuesten *metadataKey* authentifiziert werden, beispielsweise in Form eines HMAC. Dieser HMAC wird nach jedem Bearbeiten der Metadaten-Datei neu berechnet und umfasst die Abschnitte „sharing“ und „files“. Damit wird auch verhindert, dass ein Angreifer, der ehemals zu den berechtigten Nutzern gehört hat, die Integrität und Authentizität der Dateien gefährden kann. Da er den neuesten *metadataKey* nicht kennt, kann er keinen passenden HMAC berechnen und die Nutzer die Verletzung der Integrität erkennen.

Um ein Wiedereinspielen alter Versionen des Ordners erkennen zu können, sollte ein Zeitstempel in der Metadaten-Datei gespeichert werden, der ebenfalls vom oben vorgeschlagenen HMAC authentifiziert wird. Der Zeitstempel wird immer dann aktualisiert, wenn die Datei bearbeitet wurde. Speichern die Nextcloud-Apps auf den Endgeräten den zuletzt gesehenen Zeitstempel, sowie den zu diesem Zeitpunkt neuesten *metadataKey*, können sie alte Versionen erkennen.

6. ÄHNLICHE PROTOKOLLE UND ALTERNATIVEN

Im Bereich der Ende-zu-Ende-Verschlüsselung bietet sich ein Vergleich der Nextcloud-E2EE zum Messaging-Protokoll Signal [2] an, welches unter anderem in der gleichnamigen Kommunikationssoftware, aber auch in Whatsapp oder Google Allo eingesetzt wird. Während der Anwendungszweck ein anderer ist, teilen Signal und die Nextcloud-E2EE die Problematik, dass die Kommunikation über einen zentralen Server läuft, aber nur auf den Endgeräten ver- und entschlüsselt werden soll und daher Public Keys zwischen Nutzern ausgetauscht werden müssen. Im Gegensatz zur Nextcloud-E2EE könnte ein Angreifer, der seinen eigenen Public Key einschleust allerdings nur zukünftige Kommunikation mitlesen, bereits gesendete Nachrichten können nicht nachträglich entschlüsselt werden. Die Software Signal bietet ähnlich zu den Vorschlägen in Abschnitt 5 die Möglichkeit Public Keys per QR-Code oder in Form einer „Safety Number“ zu verifizieren.

Als mögliche Filehosting-Alternativen zur Nextcloud-E2EE können Seafile [1] und Boxcryptor [3] betrachtet werden. Seafile ist ähnlich wie Nextcloud eine Filehosting-Software, die auf eigenen Servern betrieben werden kann. Auch Seafile bietet die Möglichkeit einer Ende-zu-Ende-Verschlüsselung. Im Gegensatz zu Nextcloud, können verschlüsselte Dateien aber nicht mit anderen Nutzern geteilt werden. Zudem schützt Seafile die Metadaten nicht, ein Angreifer könnte also beispielsweise die Dateinamen auslesen [1]. Seafile nutzt AES mit CBC als Betriebsmodus. Damit wird nur die Vertraulichkeit der Dateiinhalte sichergestellt, nicht aber Integrität und Authentizität, die auch nicht auf anderem Wege umgesetzt werden.

Boxcryptor ist eine Software, die mit bestehenden Filehosting-Diensten, wie z.B. Dropbox, genutzt werden kann. Ihr Einsatzzweck ist speziell die Ende-zu-Ende-Verschlüsselung von Ordner auf eben diesen Diensten. Die E2EE ist dabei ähnlich umgesetzt wie die E2EE von Nextcloud. Auch ein Teilen von einzelnen Dateien oder Ordnern mit anderen Nutzern ist möglich. Hier übernimmt der Anbieter von Boxcryptor die Verteilung der Public Keys und muss daher als vertrauenswürdig erachtet werden. Generell muss zur Nutzung von Boxcryptor dessen Entwickler vertraut werden, da Anwendungen anders als bei Nextcloud und Seafile nicht quelloffen zur Verfügung gestellt werden. Wie Seafile stellt auch Boxcryptor nur die Vertraulichkeit, nicht aber die Integrität und Authentizität des Dateiinhalts sicher. [3]

7. ZUSAMMENFASSUNG

doi: 10.2313/NET-2018-11-1_10

Zusammenfassend kann festgestellt werden, dass die Ende-zu-Ende-Verschlüsselung von Nextcloud prinzipiell sinnvoll konstruiert ist, aber aufgrund zweier Schwachstellen, die die Schutzziele Vertraulichkeit, Integrität und Authentizität gefährden, noch nicht für den alltäglichen Gebrauch geeignet ist.

Eine Schwachstelle betrifft das erstmalige Teilen von Ordnern mit einem anderen Nutzer. Da der Public Key des neuen Nutzers in Form eines Zertifikats vom Server heruntergeladen und auch von diesem signiert wird, muss der Server zu diesem Zeitpunkt vertrauenswürdig sein. Die zweite größere Schwachstelle resultiert aus einer fehlenden Authentizitätsprüfung in der Metadaten-Datei. Hier kann es einem Angreifer unter Umständen gelingen, eigenes Schlüsselmaterial einzuschleusen, welches die Nutzer dann zur Verschlüsselung der Dateien verwenden.

Werden die in dieser Arbeit beschriebenen Schwachstellen behoben, ist die einzige Information, die ein Angreifer über einen Ordner mit aktiver E2EE in Erfahrung bringen kann, die Anzahl der Dateien, die sich darin befinden und deren Größe. Die Nutzung von Nextcloud mit aktiver Ende-zu-Ende-Verschlüsselung ist dann eine gute Möglichkeit, vertrauliche Daten über mehrere Geräte hinweg zu synchronisieren und wo nötig auch mit anderen Nutzern zu teilen. Leider kommt die E2EE auch mit einigen Komforteinbußen, so ist es konstruktionsbedingt nicht möglich, sie im Webbrowser zu nutzen (dieser müsste Javascript-Code ausführen, der von einem kompromittierten Server modifiziert werden könnte) oder die Dateiversionierung zu nutzen [4]. Da die E2EE aber auf Ordnerebene aktiviert werden kann, besteht die Möglichkeit, jeweils zwischen dem Schutz der Daten und dem Nutzungskomfort abzuwägen.

Im Vergleich zu den Alternativen Seafile und Boxcryptor stellt die Verwendung der Ende-zu-Ende-Verschlüsselung von Nextcloud die beste der angeführten Lösungen dar um vertrauliche Daten in der Cloud zu schützen. Kommt ein Selfhosting der Daten nicht in Frage, kann auch der Einsatz von Boxcryptor in Betracht gezogen werden.

8. LITERATUR

[1] Security features. Seafile Server Manual. `https://manual.seafile.com/security/security_features.html`.

[2] Technical information. Signal. `https://signal.org/docs/`.

[3] Technical overview. Boxcryptor. `https://www.boxcryptor.com/en/technical-overview/`.

[4] End-to-end encryption design. Nextcloud, 20. September 2017. `https://nextcloud.com/endtoend/`.

[5] The antisocial network. The Economist, 22. März 2018.

[6] A. Armando et al. The avispa tool for the automated validation of internet security protocols and applications. In *International conference on computer aided verification*, pages 281–285. Springer, 2005.

[7] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, Feb. 1990.

[8] H. Comon and V. Shmatikov. Is it possible to decide whether a cryptographic protocol is secure or not? *Journal of Telecommunications and Information Technology*, pages 5–15, 2002.

[9] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, May 2008. `https://tools.ietf.org/html/rfc5280`.

[10] C. J. Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In *International Conference on Computer Aided Verification*, pages 414–418. Springer, 2008.

[11] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[12] C. Eckert. *IT-Sicherheit: Konzepte-Verfahren-Protokolle*. Walter de Gruyter, 2013.

[13] A. Landwehr. China schafft digitales Punktesystem für den „besseren“ Menschen. `https://heise.de/-3983746`.

[14] D. McGrew and J. Viega. The galois/counter mode of operation (gcm). *NIST Modes of Operation Process*, 20, 2004.

[15] C. A. Meadows. Formal verification of cryptographic protocols: A survey. In *International Conference on the Theory and Application of Cryptology*, pages 133–150. Springer, 1994.

[16] B. Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

doi: 10.2313/NET-2018-11-1_10

Evaluation of Distributed Semantic Databases

Philipp Trucksäß
Advisor: Jan Seeger
Seminar Future Internet SS2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: philipp.trucksaess@in.tum.de

ABSTRACT
In this paper, several implementations of *RDF* data stores are compared, analyzing their unique innovations and evaluating their benefits and shortcomings for various applications. The goal is to give an overview of popular implementations and identify potential for further research.

Keywords
RDF, SPARQL, Distributed Semantic Databases

1. INTRODUCTION
With the rising popularity of the *Semantic Web*, *Semantic Databases* are of increasing importance, as in the often cited *DBPedia* project, the *Linked Open Data* platform of *Wikipedia* [7]. The goal of the *Semantic Web* is to provide a way to describe data and relationships between data items, which makes it easier to process for machines. This is particularly relevant because modern systems have to process an ever increasing amount of information.
For instance, *IoT* applications are becoming more prevalent with the growing acceptance of smart home appliances. Those make use of sensor networks which produce gigabytes of data every seconds, and with them comes a need for a standardized way to integrate data from different sources, and to store and process that data in an efficient way. Semantic databases can help with that feat.
The **R**esource **D**escription **F**ramework, usually known as *RDF*, standardizes the interchange of semantic data on the web. The term "Semantic Data" refers to the structure of the data as "**S**ubject", "**P**redicate" (or property), "**O**bject", which is analogous to that of an English sentence. Because of this focus on *triples*, the respective databases are called *Triple Stores*.
The subjects and objects can be thought of as nodes in a graph, where edges between those nodes represent predicates. Even more so than for a traditional relational database system, this graph structure calls for a distributed implementation.
While some approaches simply provide an *API* for *RDF* and its associated query language *SPARQL* on top of an established central database architecture, I present some of the more innovative distributed approaches and analyze their capabilities and unique features.

2. BACKGROUND
RDF as a standard is maintained by the *W3C*. For the purpose of comparing different *RDF* data stores it is helpful to have a basic understanding of the *RDF* schema and the *SPARQL* query language, which is most commonly used for examples and benchmarks by the presented stores.

2.1 RDF Schema
The most recent version of the *W3C RDF Schema Recommendation* (at the time of writing) specifies the vocabulary to be used for *RDF* data [16] [17]. It defines an extension to *RDF*, focused on describing groups and their relationships.
At its core, the schema provides a framework to model relationships and properties of different kinds of *resources*, which are organized in *classes*. The term is used not unlike its meaning known from object-oriented programming languages in this context. In *RDF* triples, these resources can take the place of *subject*, *property* (*predicate*) or *object*.
There are several different concrete syntaxes implementing the schema, such as the simple *JSON RDF* or *Turtle*. Those can be extended to impose entailment regimes, e.g. by *OWL*, the web ontology language which can describe complex logical contexts between resources.The only concrete syntax used in the standard is `namespace:name`.
Resources are specified by literal strings or numerals and pairs of resources can be related to each other via *properties*. Those properties implement hierarchical relationships, so a more specific property implies a more general property, that it is a sub-property of. Two properties are special; the *range* property refers to resources with a certain property, which themselves are instances of a number of classes. The *domain* property specifies, that resources with a specific property have to be instances of one or multiple classes. In the same vein, several other descriptors of hierarchical relationships and groupings are available.
Further, the *RDF* vocabulary also allows specifying labels and comments to provide human-readable descriptions, and set-theoretical groupings like `rdf:Bag`, as well as a syntax to describe *RDF* statements, e.g. `rdf:subject`, and more.

2.2 SPARQL
The "*SPARQL* Query Language for *RDF*" specifies a language and a protocol for interacting with *RDF* graph content [15]. *SPARQL* supports queries which use similar keywords to regular *SQL* queries, as seen in 1.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <"http://www.w3.org/1999/02/22
-rdf-syntax-ns/>
SELECT ?name
WHERE {
```


doi: 10.2313/NET-2018-11-1_11

```
    ?student foaf:name ?name.
    ?student hears ?lecture.
    ?professor holds ?lecture.
    ?professor foaf:interest ?subject;
        rdfs:label "RDF".
} GROUP BY ?student ?name
```

Listing 1: SPARQL example query

This example shows the use of triple patterns in the `WHERE` clause, where a `;` indecates that the previous object is the subject of the following triple. Each identifier preceded by a `?` is a variable, and literal values are bound to the query solution. The example also demonstrates how more complex relationships between multiple variables can be queried. Those relationships represent paths in the data graph. A common pattern in *SPARQL* is the introduction of several variables which occur in several places throughout the query, often connected by multiple predicates.
A problem which all presented data stores try to solve one way or another, is that *SPARQL* queries with many interrelated variables tend to be quite expensive if the underlying data storage model is based on a relational database. The relational data model was designed to retrieve information about indexable data items. In the semantic data model, quick access to any of the stored resources is needed, so, as shown later, a need for extensive index structures exists.
Each node in the query graph, in which two edges meet, implies the need for a table join. These joins are unavoidable in relational databases, but they do in general not scale well, as they are so called "pipeline breakers", so they should be kept to a minimum. The number of joins that are required even for simple *SPARQL* queries like 1 is significantly higher than usually seen in *SQL* queries of comparable complexity. Therefore, an alternative approach to relational modelling of *RDF* data, which avoids large numbers of joins is presented e.g. in 3.2.1. The method used there is graph exploration, which lends itself to distributed parallel execution while only requiring one final join phase in the end, but it requires a more complex message passing system.

3. OVERVIEW OF RELEVANT DISTRIBUTED RDF SYSTEMS

With the diversity of implementations, which have been proposed focusing on optimizing various, but not necessarily mutually exclusive, parts of the *RDF* architecture, it is not straightforward to draw comparisons between them. Kaoudi et al. [7] devise useful categories for popular *RDF* systems. One of the two main groups are the *MapReduce* based implementations that operate on key-value stores and aggregate partial results, which can then be reduced to produce the final solution for a query. They are conceptually closer to traditional relational database systems and the reduction phase relies heavily on joins.
On the other hand, there are the graph based implementations, which generally map entities, i.e. subjects and objects of the *RDF* triples, to adjacency lists containing in- and outgoing edges, which represent their properties. That representation allows for the use of graph exploration to answer queries.
These two groups can be further separated as discussed by Öszu et al. [10] and Peng et al. [12] with some correlation with the previously introduced groups. The category they label as "cloud based approaches" has a strong overlap with *MapReduce* implementations, as they make use of cloud platforms' native file systems, which already come with their own *MapReduce* implementations. The term "partitioning-based approaches" is employed for fragmented *RDF* data, which is distributed over several stores which individually operate on any kind of centralized stores and are coordinated by a master node.
"Federated systems" are those which send sub-queries over a set of *SPARQL* endpoints and assemble the partial solutions afterwards. As a last group they propose "partial query evaluation approaches", which are similar to federated ones, except that the full query is sent to each partition and partial matches are propagated under the condition that they contain partition crossing edges.
Another fundamental difference between the systems, regarding the style of distribution, is pointed out by Hammoud et al. [4]. Here, four *Quadrants* are distinguished, where:

- **Quadrant I** represents fully centralized systems.
- **Quadrant II** describes systems with distributed data, where the full query is sent to each partition.
- **Quadrant III** contains those systems where both the data and the query get partitioned and distributed.
- **Quadrant IV** replicates the full data over distributed nodes and partitions the query.

In the following, representatives across the spectrum are evaluated in more depth. In particular, the maturity of the implementation, the type of architecture with respect to unique design decisions, and the reasoning tasks that are supported, are given special attention.

3.1 Cloud Based Triple Stores

Approaches to utilize cloud architectures by building on top of a distributed file system, such as *Hadoop's HDFS* used by Hammoud et al. [4] are among the earliest proposed distributed *RDF* systems. The approach is rather straightforward, as it builds on a preexisting cloud infrastructure, with built in resilience and elasticity, as well as data partitioning backed up by redundancies.
The core technology behind these straightforward cloud based triple stores is the *MapReduce* programming model [2]. `map` and `reduce` are common functions offered by functional programming languages, which are used for list processing. The *MapReduce* implementation brings their functionality to key-value pairs stored in distributed clusters. In the context of e.g. *SPARQL* queries, the mapping phase produces matches for a query from all relevant partitions, which are then reduced i.e. combined incrementally into intermediate query solutions up to the final answer.
The partitions are simply represented as individual files. An individual file can be used to represent various levels of abstraction over the triples. Each subject can be stored as one line in a file, containing all properties associated with the subject which is sometimes known as horizontal partitioning [7]. To execute a query on such a data structure

doi: 10.2313/NET-2018-11-1_11

Name	Approach	Basis	Partitioning	Architecture
H_2RDF+	MapReduce	HBase	Vertical (permutations)	Master-Slave
EAGRE	MapReduce	e.g. Cassandra	Space-Filling Curve	Master-Slave
S2RDF	Data Parallel Computation	Spark (Hadoop)	ExtVP/Vertical	Master-Slave
Trinity.RDF	Graph Exploration	Trinity	Horizontal	Master-Slave
TriAD	Graph Exploration	Custom Implementation	Horizontal	Master-Slave
gStore	Local Partial Match	Custom Implementation	Index-based	Master-Slave / Decentralized
DREAM	Join Vertices	Any	N/A	Master-Slave

Table 1: An overview of distributed RDF databases

may require a full scan of the data. Alternatively, extensive indexing can be used to represent any constellation of the triples and facilitate direct access for any query, but not without massive data duplication. A popular scheme is the storage of all six possible permutations of subject, predicate, object, each as the key of a key-value store without associated value as used by Zheng et al. [19].
Another popular ordering is the partitioning of one file per property, so all edges of a certain kind can be found in one place without redundant data. Such a vertical partitioning scheme is employed by Husain et al. [5] among others. For the purpose of stability, each partition is usually replicated over several storage points.

3.1.1 *H₂RDF+*

An example of a *MapReduce* based distributed *RDF* store was introduced with *H₂RDF+* by Papailliou et al. [11]. It builds upon the *NoSQL* key-value store *HBase*, which in turn employs *HDFS* for data storage. All six permutations of subject, predicate, object per triple are stored as keys with empty values. A separate store is used as a dictionary to map string labels to IDS with different lengths, where more common values receive shorter IDs, utilizing byte level variable length encoding. The partitioning of the data is left to the underlying *HBase* implementation, which distributes its input into several ranges of key-value mappings.
When using *MapReduce* for *RDF* queries, results are produced by joining triple patterns over a join variable which they share. *H₂RDF+* focuses on two efficient merge join algorithms, one for multi-way merge join and one for sort-merge join. For the sorted content of the index table the "MapReduce Merge Join Algorithm" joins multiple triple patterns which have a variable in common. The biggest partition produces its share of the queried range via a map-only job and the other partitions are merge-joined by local scanners.
Intermediate results are unsorted, or rather not ordered by the variable to join over, therefore they are handled by the "MapReduce Sort-Merge Join Algorithm". The biggest partition over the join variable is used to generate a global ordering for the reducers, which can then operate on sorted ranges. If only intermediate results are joined, hash partitioning can be used to perform a hash join instead. The joins can be performed in a distributed or a centralized version, depending on the workload.
To minimize query execution time, the optimal join order is decided for every step iteratively by an online planner which calculates costs based on statistics from previous queries. Only joins of a certain size profit from distributed execution, smaller ones are run centrally. The planner runs on a master node, but centralized joins can also be executed on any slave node.
The merits of the adaptive join execution are more measurable for big inputs and complex queries where it yields performance benefits over simpler approaches. Especially for less selective queries, the use of sorted ranges is useful and it also enables various range queries. Complex reasoning tasks are possible, but they may require an inflationary number of joins, so they can quickly become unfeasible.

3.1.2 *EAGRE*

For distributed queries in a cloud based system, I/O operations need to be coordinated with care, or they represent a significant bottleneck. A *MapReduce* based implementation with the aim to target that issue is *EAGRE* ("**E**ntity-**A**ware **G**raph comp**RE**ssion") [19]. *EAGRE* strives to incorporate semantic and structural data into its data format in a more sophisticated way than a simple key-value store of previous cloud-based *RDF* databases such as *Trinity* which will be described in 3.2.1. Queries are answered with the *MapReduce* method, with a focus on minimizing disk I/O and network traffic or rather on a desirable trade off between network traffic costs and the benefit of distributed I/O.
To limit the expenses of *MapReduce*, intermediate results are estimated using a *Bloom Filter* and the special *Consulting* protocol which facilitates the exchange of information about the specific value ranges of each compute node, so the search space can be limited in the scheduling phase. To leverage this range based optimization, the data needs to be partitioned while keeping its inherent order. Zhang et al. [19] limit the evaluation of *SPARQL* to `SELECT` queries, but for those the partitioning is favorable for range and order constraints.
EAGRE's approach to modeling *RDF* data focuses on the subjects and describes them as *Entities*, each with a key-value collection of its properties and respective objects. Based on shared description keys, the entities are categorized into *Entity Classes*, by which they can then be grouped to generate a so called "Compressed RDF Entity Graph". The grouping is first performed in a randomized distributed fashion, and subsequently the resulting structure is partitioned using *METIS* while preserving the data locality as well as possible. Within each compute node the layout of the entity classes is determined using the *Space Filling Curve* algorithm for high dimensional data, to effectively place the connected entities together and save unnecessary disk I/O. This rather involved partitioning scheme revolves around I/O optimization, but because of its complexity, it has its downsides when used with dynamic data, as updates of the data set become expensive.

doi: 10.2313/NET-2018-11-1_11

The evaluation of a query then utilizes the *Consulting* to minimize the variable ranges to be read so only a minimum amount of data blocks needs to be read. The final *MapReduce* is postponed until the most beneficial variable set has been determined, so it operates on a minimal data set and network payload is as small as possible.
While disk I/O speeds have increased since *EAGRE* was first introduced, network communication continues to motivate elaborate scheduling optimization [1].

3.1.3 S2RDF

The last implementation to be discussed here, which is based on a cloud store, is *S2RDF* [14]. Under the realization that the query patterns usually encountered with cloud-based triple stores are in practice more limited than what the *SPARQL* specification allows, because the full range of possible queries is often undesirably slow to execute, the relational partitioning schema *ExtVP*, which strives to minimize the input sizes of any kind of query pattern.
The underlying in-memory cluster computing system *Spark* runs on top of *Hadoop* data sources. It uses *Resilient Distributed Datasets* to store the data, which allows parallel operations on the contained elements and provides fault tolerance. Like *MapReduce* the *data parallel computation* model operates on multiple records of data in parallel, and on top of this, *Spark* comes with multiple utilities like the *Cataclyst* optimizer, which is originally intended for heavily optimized computations in a relational database interface. The data is stored in a column layout, with the benefit of performance improving compression while the table schema is preserved.
The *ExtVP* **ext**ended **v**ertical *p*artitioning scheme was designed to minimize I/O, as well as the number of necessary join operations, while not catering to any particular query shape (such as specifically star-shaped queries). The basic vertical partitioning uses one table per property, with one column each for the occurring subjects and objects. This leads to rather unbalanced tables and many intermediate results which are discarded later. To avoid a good part of these, *ExtVP* uses precomputed joins for all pairs of property tables, which was found to be still more space efficient than e.g. storing all six permutations of tuples, as presented in 3.1.1. The possible benefit is greatest, if the join is of high selectivity, which also leads to a memory efficient representation, which provides a good heuristic for cases that justify precomputation.
Queries are processed as algebraic trees, which are traversed bottom-up. The joins are ordered to provide maximum selectivity, i.e. triples with common variables are preferably joined, to limit necessary network traffic.
Schätzle et al. [14] found *S2RDF* to be significantly faster than previous frameworks for common query shapes. They also found the speed up for the less common query shapes, which motivated the development in the first place, to be even more pronounced. That also means that complex inference tasks can be performed efficiently, as the precomputed joins reduce the necessary workload. *S2RDF* is a good example, how synergies between meticulously optimized traditional relational database engines and *RDF* systems can be utilized, in this case in the form of *Spark*.

3.2 Graph based triple stores

Many of newer additions to the plethora of *RDF* data management systems are found in the family of graph based variants. They are usually based on cloud architectures as well, but they use less of their native features, and utilize custom implementations of e.g. index structures instead, as in the work of Zou et al. [20]. They are motivated by the demand for graph operations, which are not trivially supported in *MapReduce* systems, where the connections between the vertices have to be derived from joins as discussed by Zheng et al. [18]. In graph-oriented storage schemes, the nouns of the triples readily provide all edges connected to them by means of adjacency lists. This way, less duplication of data is required and especially more complex queries can be executed faster.
The partitioning is usually done by *METIS*, a software package for graph partitioning where the number of adjacent vertices distributed to separate compute nodes is minimized [8]. Independently of the partitioning scheme, a nonzero amount of fragment boundary crossing edges can not be avoided in general. One possible way of dealing with this is to replicate the *n-hop* neighboring nodes of a partition's border nodes for each partition. In a federated method, a query then will have to be able to crawl through any number of partitions, crossing between them when an edge spanning fragment boundaries is followed. However, the federated approach has only limited potential for parallelization [7]. A more efficient use of clustered computing power is achieved by the *Partial Evaluation* method, where matches are first computed for every partition independently, accepting also sub-matches of only parts of the query, and then combining them to reach a final solution. The combination of the partial matches can either be performed on a central server, or in a distributed fashion, which makes better use of the available computing power and also puts less contention on the network connections, but not without an overhead in scheduling and synchronizing network communication [10].

3.2.1 Trinity.RDF

Trinity.RDF was first presented by Zeng et al. [18] as an attempt to store graph data in its native graph form instead of a set-based approach, in order to support the full range of operations only available for graphs, such as reachability queries. The graph is stored in memory, distributed in a cloud to make random accesses feasible.
The "Sideways Information Passing" technique or *SIP* is employed in conjunction with graph exploration for dynamic optimization of parallel execution. It lets filters on identifiers be shared between compute nodes processing similar such identifiers. The main assumption of Zeng et al. [18] is that graph exploration can be performed more efficiently than equivalent joins in a table-based storage format.
The data is partitioned by randomly hashing the nodes across a cluster, where each machine receives a disjoint partition of the graph. The data is committed to *Trinity* key-value stores. One machine is called a "proxy" and represents a master node which coordinates the query plan and assembles the final result from the slaves holding the data partitions. A special server holds mappings from literal strings to unique IDs, so the *Trinity* stores only need to retain fixed-size identifiers and the partitions are more memory efficient. Every *RDF* entity is represented by a node whose identifier is the key for the key-value store, and is used to retrieve adjacency lists of all incoming and outgoing edges, which represent the predicates and respective subjects or objects. In the context of *Trinity.RDF* this key-value store is con-

doi: 10.2313/NET-2018-11-1_11

sidered a "local predicate index", where predicates can be found for any subject or object. On top of that, the "global predicate index" of each machine maps each predicate to all occurring subjects and objects with that predicate.
Because the random partitioning may incur massive costs in network traffic, the adjacency lists are further split up depending on the machine owning the edge's receiver node, so the internal nodes on one machine can be explored first, and the collective information leading the exploration to the next machine can be sent as a block.
Trinity.RDF was found to be more memory efficient than other more heavily indexed implementations, but it is important to know that the assumption that graph exploration outperforms join-based query processing proved to be correct only for rather specific complex queries.

3.2.2 TriAD

Introduced by Gurajada et al. [3], the *TriAD* (for "Triple Asynchronous Distributed") *RDF* engine puts a strong focus on parallelization as its *shared-nothing* architecture enables several nodes in a cluster, as well as several cores on the same machine to operate on part of a query completely autonomously. It is presented as a direct successor to *Trinity.RDF*, trying to avoid its perceived shortcomings.
The essential proposition to enable this high level of scalability is *TriAD's* custom asynchronous message passing protocol, which allows strongly multithreaded query execution without a big synchronization overhead. Still, the compute nodes are hierarchically ordered, with a master node whose purpose it is to receive and optimize all queries. The master node is also keeping track of the summary graph, a simplified mapping of the *RDF* data graph, where each summarized node contains enough information about a set of data nodes to prune the nodes which do not contain relevant triples from a query plan.
The slave nodes contain indices for all six permutations of the triples, so all possible point queries can be answered directly via the most appropriate index structure. To partition the graph between the slave nodes, triples are hashed and distributed horizontally, depending on the summarized super-node they belong to, to allow for pruning.
The horizontal partitioning is suited for graph exploration, which is used to answer queries starting from opportune nodes and collecting data while traversing potentially many of the graph partitions. Practically, the range of exploration is still limited enough to be efficient because of the locality-preserving partitioning scheme and the join-ahead pruning algorithm. However, there is only direct support for point queries, and range queries may require a scan of the entire graph, as the ranges may spread over several partitions.

3.2.3 gStore

Another interesting graph based representation of *RDF* data is *gStore* [20]. More so than other implementations presented here, *gStore* stores data in custom data structures specifically tailored to the needs of *RDF* graphs. At their core is the *vertex signature* representation of the entities and classes present in the graph. This binary representation works similarly to a *Bloom Filter* over the adjacent edges and neighboring vertices. Each vertex is identified by an adjacency list, where edges and neighbors are hashed and the results are combined via bitwise `OR`.
This encoding is used to create a signature graph. The query graph is encoded analogously, and a *VS*-tree*, a **v**ertex **s**ignature tree, is used to answer the query. The *VS*-tree* summarizes the signature graph at different resolutions, so while it is matched against the query down to the leaf nodes, the search space can be pruned efficiently on each level
Eventually the positive matches generate small materialized aggregates. For caching purposes all transactions are collected in a transaction database managed by a trie structure called *T-index*. The queried predicates are ordered by frequency in order to minimize the number of materialized views that have to be maintained.
While the problem of updating the used data structures efficiently is addressed, there is no way to avoid updating entire paths in both the *VS*-tree* and the *T-index* when a single triple changes, which makes maintenance of highly dynamic data rather expensive.
In its original form *gStore* is not a distributed system but an addition to *gStore* has been introduced by Peng et al. [12] with the distributed *Local Partial Match* technique, although it could also be employed in conjunction with other graph-based query engines. It is motivated by the problem of finding matches which cross compute node boundaries in an efficient way.
The idea is that, given a complete query graph, each node explores its local data set and maximizes the sub-graph matching the query. The partitions are disjoint, so nodes do not share any vertices, but crossing edges between the partitions are available at both ends. If an incomplete match is actually a fragment of a solution to the query, any edge where only one end has a match in the current partition must be a crossing edge. Based on this proposition, all compute nodes produce maximal local partial matches in parallel. Special `NULL` values which match anything are introduced for crossing edges.
Once the partial matches have been found, the assembly phase is started. Assembly can be performed either centrally or in a distributed manner, employing the *Bulk Synchronous Parallel* model. If the final assembly is performed centrally by a master node, an iterative pairwise join of local partial matches is performed. First, matches are partitioned into sets where each sets only contains matches which cannot be joined with each other to reduce the amount of possible joins to be considered. Then, crossing edges need to be closed with matches from other sets until a complete match for the query is found, or the partial match can be refuted because there is no viable join partner left.
Distributed assembly works similarly, but each compute node only evaluates joins for a subset of the matches and then sends the results to other nodes for which they are relevant, indicated by shared crossing edges. Distributed assembly still implies that queries are scheduled by a master node and executed by slaves, but introduces a higher level of peer to peer communication.
This approach shifts the bulk of the network communication away from graph exploration, as the boundaries between partitions are only crossed once the maximally available local information has been computed already. The authors benchmarked this method in comparison to previous implementations and found the greatest speed gain when used for complex queries. Their benchmark results show though, that the scalability of *Local Partial Match* strongly depends on the cost of communication during the assembly phase, so highly selective queries perform better due to the reduced

doi: 10.2313/NET-2018-11-1_11

amount of information to be transmitted.

3.2.4 DREAM

The "**D**istributed **R**DF **E**ngine with **A**daptive Query Planner and **M**inimal Communication" by Hammoud et al. [4] presents a paradigm shift compared to the other systems mentioned here. Instead of partitioning the *RDF* data set, it simply replicates it over distributed compute nodes and partitions incoming queries instead in an attempt to avoid communication overhead and data passing as well as scheduling issues.

By the time of *DREAM's* introduction, it was the first system to employ this approach of partitioning, motivated by the fact that the largest *RDF* data sets did not exceed 2.5 TB in size, which makes replication of the entire set feasible. The unique feature of *DREAM* is its query planner, which is run on a central master node, while the slave nodes are kept agnostic as to which storage system is deployed to them locally.

The query planner first generates a query graph and then partitions it into basic sub-graphs, which may be exclusive or shared depending on the query as a property of their join vertices, where one join vertex represents an exclusive sub-graph, and multiple of them a shared one. From the sets of join vertices a directed set graph is generated, where each node represents a set of join vertices, and considering the direction of their associated edges. On the directed set graph the most opportune query plan is determined using a cost function. The join vertex sets of the query plan are then sent to one slave node each, where the query is started in parallel. Once the candidates for a join vertex set are known on one machine, only their IDs need to be broadcast to the other machines who share them, as every one of them holds the full data graph. The cost function deciding the lowest cost plan takes the interdependencies into account and strives to minimize the idle time of slaves waiting for auxiliary data from other sub-graphs. The query planner can adapt the number of slave nodes deployed for any particular query to the involved workload, bounded by the number of join vertices.

The *DREAM* approach is most suited for minimizing query response times and adaptive work-balancing, but does not put much emphasis on memory efficiency, and the join vertex approach is not a good fit for range queries.

4. CONCLUSION

All of the presented systems address different issues in the organization of *RDF* data. While it may seem like there is a trend towards graph based systems, that may be caused by them leaving more room for variation, because there is more development from the ground up. Future development will likely be directed by technological progress, e.g. work by Mittal et al. [9] indicates that the widespread availability of *Storage Class Memory* will have a strong impact on the architecture of database systems in general, and especially systems like *triAD*, which are already designed as in-memory stores, can definitely profit from such hardware advances.

On the front of network communication cost, traditional *RDBMS* have already paid attention to faster interconnection technologies like *InfiniBand* [13], which require software changes to reap their full potential benefits, which might become more relevant for *RDF* systems as well.

There is also the sector of *IoT* applications, where different technological parameters require a stronger focus on memory efficiency as presented by Jiang et al. [6], where an architecture for *IoT* data storage based on *Hadoop* is introduced. Those memory constraints imply that implementations like *DREAM* are not the best match. *IoT* also puts different constraints on network data exchange, so optimizations like those in *EAGRE* are quite relevant for the sector.

It appears that future research in the field of *RDF* data storage systems has to focus on adapting to hardware paradigms which seem to be yet unexplored for triple stores. Possibly, synergies with relational database systems can be used further to the advantage of *RDF* systems, as discussed in the context of *S2RDF*, since those tend to be on the forefront of technological advances.

5. REFERENCES

[1] L. Cheng, S. Kotoulas, T. E. Ward, and G. Theodoropoulos. Improving the robustness and performance of parallel joins over distributed systems. *J. Parallel Distrib. Comput.*, 109(C):310–323, Nov. 2017.

[2] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[3] S. Gurajada, S. Seufert, I. Miliaraki, and M. Theobald. Triad: A distributed shared-nothing rdf engine based on asynchronous message passing. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 289–300, New York, NY, USA, 2014. ACM.

[4] M. Hammoud, D. A. Rabbou, R. Nouri, S.-M.-R. Beheshti, and S. Sakr. Dream: Distributed rdf engine with adaptive query planner and minimal communication. *Proc. VLDB Endow.*, 8(6):654–665, Feb. 2015.

[5] M. Husain, J. McGlothlin, M. M. Masud, L. Khan, and B. M. Thuraisingham. Heuristics-based query processing for large rdf graphs using cloud computing. *IEEE Transactions on Knowledge and Data Engineering*, 23(9):1312–1327, Sept 2011.

[6] L. Jiang, L. D. Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu. An iot-oriented data storage framework in cloud computing platform. *IEEE Transactions on Industrial Informatics*, 10(2):1443–1451, May 2014.

[7] Z. Kaoudi and I. Manolescu. Rdf in the clouds: A survey. *The VLDB Journal*, 24(1):67–91, Feb. 2015.

[8] G. Karypis and V. Kumar. Metis—a software package for partitioning unstructured graphs, partitioning meshes and computing fill-reducing ordering of sparse matrices. 01 1997.

[9] S. Mittal and J. S. Vetter. A survey of software techniques for using non-volatile memories for storage and main memory systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(5):1537–1550, May 2016.

[10] M. T. Özsu. A survey of RDF data management systems. *CoRR*, abs/1601.00707, 2016.

[11] N. Papailiou, I. Konstantinou, D. Tsoumakos, P. Karras, and N. Koziris. H2rdf+: High-performance distributed joins over large-scale rdf graphs. In *2013 IEEE International Conference on Big Data*, pages 255–263, Oct 2013.

doi: 10.2313/NET-2018-11-1_11

[12] P. Peng, L. Zou, M. T. Özsu, L. Chen, and D. Zhao. Processing sparql queries over distributed rdf graphs. *The VLDB Journal*, 25(2):243–268, Apr. 2016.

[13] W. Rödiger, T. Mühlbauer, A. Kemper, and T. Neumann. High-speed query processing over high-speed networks. *Proc. VLDB Endow.*, 9(4):228–239, Dec. 2015.

[14] A. Schätzle, M. Przyjaciel-Zablocki, S. Skilevic, and G. Lausen. S2rdf: Rdf querying with sparql on spark. *Proc. VLDB Endow.*, 9(10):804–815, June 2016.

[15] W3C. Sparql 1.1 overview. `https://www.w3.org/TR/sparql11-overview/`, 2013. Accessed: 2018-04-01.

[16] W3C. Rdf 1.1 semantics. `https://www.w3.org/TR/rdf11-mt/`, 2014. Accessed: 2018-04-01.

[17] W3C. Rdf schema 1.1. `https://www.w3.org/TR/2014/REC-rdf-schema-20140225/`, 2014. Accessed: 2018-04-01.

[18] K. Zeng, J. Yang, H. Wang, B. Shao, and Z. Wang. A distributed graph engine for web scale rdf data. In *Proceedings of the 39th international conference on Very Large Data Bases*, PVLDB'13, pages 265–276. VLDB Endowment, 2013.

[19] X. Zhang, L. Chen, Y. Tong, and M. Wang. Eagre: Towards scalable i/o efficient sparql query evaluation on the cloud. In *29th International Conference on Data Engineering*. IEEE, Apr. 2013.

[20] L. Zou, M. T. Özsu, L. Chen, X. Shen, R. Huang, and D. Zhao. gstore: A graph-based sparql query engine. *The VLDB Journal*, 23(4):565–590, Aug. 2014.

doi: 10.2313/NET-2018-11-1_11

An overview over Capsule Networks

Luca Alessandro Dombetzki
Advisor: Marton Kajo
Seminar Innovative Internet Technologies and Mobile Communications
Chair of Network Architectures and Services
Department of Informatics, Technical University of Munich
Email: luca.dombetzki@tum.de

ABSTRACT

Hinton et. al recently published the paper "Dynamic Routing Between Capsules" [20], proposing a novel neural network architecture. This Capsule Network (CapsNet) outperforms state-of-the-art Convolutional Neural Networks on simple challenges like MNIST [13], MultiMNIST [20] or smallNORB [6]. In this paper, we describe multiple aspects of the current research in Capsule Networks. This includes explaining the shortcomings of CNNs, the idea and architecture of Capsule Networks and the evaluation on multiple challenges. Furthermore, we give an overview of current research, improvements and real world applications, as well as advantages and disadvantages of the CapsNet.

Keywords

capsule networks, overview, computer vision, convolutional neural networks, pooling

1. INTRODUCTION

In 1986 Geoffrey E. Hinton revolutionized artificial neural networks with the use of the backpropagation algorithm. Since then artificial intelligence has made a big leap forwards, especially in computer vision. Deep Learning gained in popularity, when deep convolutional networks performed extraordinarily well on the ImageNet challenge in 2012 [10]. It has since been a very active field of research, with companies like Google and Microsoft being dedicated to it. This brought forth ideas like Inception [23] and Residual blocks [5], boosting the performance of CNNs. However, all of these advancements build upon the basic structure of a CNN.

Based on his research in human vision, Geoffrey E. Hinton stated that there is something fundamentally wrong with CNNs [7]. By trying to replicate the human visual cortex, he came up with the idea of a capsule as a group of neurons. In "Dynamic routing between capsules" [20] Hinton et. al developed a first working implementation, proving this theory. In computer graphics, a scene is built by putting known parts into relation, forming a more complex object. Inverse graphics does the exact opposite, the scene is deconstructed into parts and their relationships. The main goal of the Capsule Network is to be capable of performing inverse graphics [7]. To achieve this, Hinton proposed to encode the idea of an entity inside a neural network, a capsule [7].

In the following, we first explain the basic structure of a Convolutional Neural Network and its possible shortcomings in Section 2. In Section 3.1 and 3.2, we describe the architecture of a Capsule Network by comparing it to a general CNN. Furthermore, we explain the idea and implementation of routing-by-agreement algorithm, as well as the loss functions and the training of the network (Section 3.3-3.4). Section 3.5 shows the network's performance on multiple challenges, including an overview over novel matrix capsules in Section 3.6. After summarizing the main advantages and disadvantages in Section 3.7 and 3.8, we look at improvements to the CapsNet (Section 3.9). Before drawing a conclusion in Section 4, we outline some real world use cases of capsule networks in Section 3.10.

2. CONVOLUTIONAL NEURAL NETWORKS

2.1 Basic CNN architecture

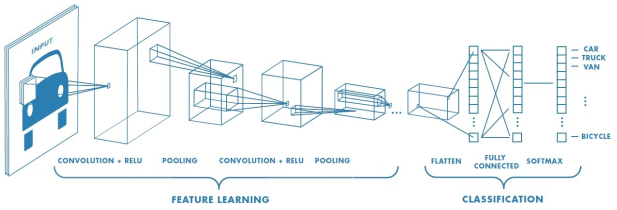


Figure 1: Basic architecture of a CNN; figure from [15]

Fig. 1 shows the basic architecture of a Convolutional Neural Network. The network first extracts learned features, which are then fed through a fully connected neural network, that produces a classification. The network can learn features by chaining together convolutional blocks. Such a block consists of a convolutional layer, an activation function and a pooling layer. The convolutional layer learns multiple simple features, also called kernels. To learn more complex, non-linear, problems, the output is fed through a non-linear activation function (e.g. ReLU). To connect the blocks together, the

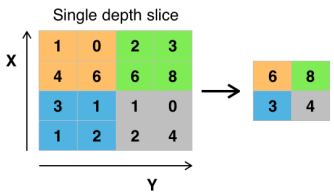


Figure 2: Max pooling example; figure from [2]

outputs of the previous block need to be routed to the next block's inputs. The most commonly used routing algorithm

doi: 10.2313/NET-2018-11-1_12

is pooling. Max pooling can be seen in Fig. 2 To improve the classification significantly, it discards unimportant activations and only propagates the most prominent ones. This allows the next convolutional layer to work only on "important" data and makes the classifier robust against small transformations in the input data.

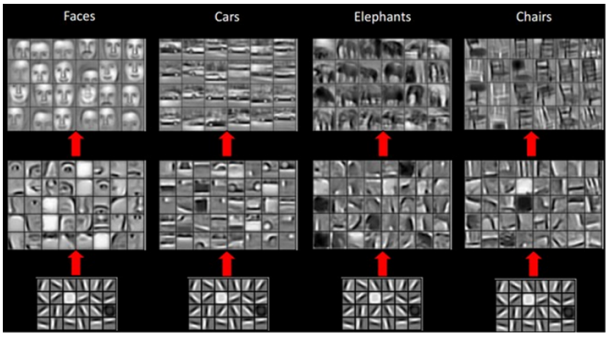


Figure 3: Feature detections of a CNN from [14]

Adding more blocks allows later blocks to extract features from results of the previous block. Thereby the network can learn more and more complex features. Like in Fig. 3 this means that the first block of a CNN might learn edges, the next block learns parts of an object parts and ultimately the last block learns complete objects like a dog, a plane, etc. With enough pooling layers, the network can become location invariant. Hence a car in the top left corner of the image is detected as well as one in the lower right.

2.2 Problems with pooling for routing

In his talk "What is wrong with convolutional nets?"[7], Hinton stresses his belief in convolution, however he brought forth four main arguments against using pooling for routing. They are explained in the following.

2.2.1 *Pooling is unnatural*

Hinton states that pooling is a "bad fit to the psychology of shape perception". Human vision detects the object instantaneously. Based on the information this object holds, we route it to the area in the brain that best handles that information. Max pooling on the other hand routes the most active information to all subsequent neurons. This is unnatural and prevents the network from learning small details.

2.2.2 *Invariance vs. equivariance*

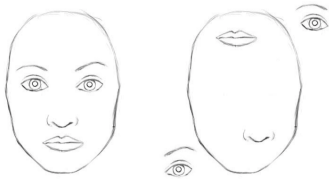


Figure 4: Both images are being classified as "face" by a CNN; figure from [9]

CNNs try to make the neural activities invariant to small changes in the viewpoint, pooling them together [7]. This is helpful for classification tasks, since the label should be the same, no matter where the object is (spacial invariance). However changes in viewpoint should ideally lead to changes in neural activities [7] (spacial equivariance). This is especially important for segmentation and detection tasks, but is also needed in classification. Fig. 4 shows that CNNs can only detect features, but cannot relate them. This means that, in an extreme case, both images are perfect representations of a face for a CNN.

2.2.3 *Not using the linear structure of vision*

A single transformation, such as rotation, can change a significant number of pixels in the image. Thereby the viewpoint is the largest source of variance in images [7]. In computer graphics, composing the scene of multiple parts is a linear problem, since we know all the relations between the parts. Without using this linear structure in computer vision, the problem of detecting an object is not linear anymore, but far more complex. As a result, a normal CNN has to be exponentially increased in size and trained with a similarly exponential amount of data [20].

2.2.4 *Dynamic instead of static routing*

Pooling collects the most prominent activations, but transports them to the same neurons of the following layer. This is like broadcasting an important information. Thereby, if the input image is translated, a completely different set of neurons is responsible for handling different kinds of activations. Our human vision however is smart enough to understand that the image is translated and activates the same neurons. This happens at runtime, hence is dynamic, and not statically preconnected like pooling. This is like letting the neurons from the next layer choose, what is most interesting to them. In a nutshell, pooling chooses the best input, while dynamic routing chooses the best worker neuron.

Nonetheless, pooling still works, as long as the network is big enough and trained with enough data, that neurons of the following layers can handle every input type. As a result, CNNs perform very well, when classifying images with only one kind of object. But especially in detection and segmentation, they lack in performance, since the information of multiple objects has to stay separated.

2.3 Solution

To solve these shortcomings, Hinton proposes to propagate not only the probability of a feature's existence, but also the spacial relationships, i.e. the pose of the feature[7]. By adopting biological research he tries to tackle all of the problems stated above. His implementation of this solution is known as Capsule Networks.

3. CAPSULE NETWORKS

Hinton's basic idea was to create a neural network capable of inverse graphics. In other words the network should be able to deconstruct a scene into co-related parts. To achieve this, the architecture of a neural network needs to be changed to reflect the idea of an entity. Every entity gets its own part of the network, encapsulating a number of neurons. This entity is called a capsule.

3.1 The capsule

A normal layer of neurons will be divided into many capsules, which in turn contain the neurons [20] (see Sec. 3.2).

doi: 10.2313/NET-2018-11-1_12

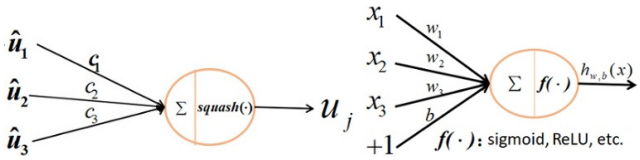


Figure 5: An capsule and neuron in comparison [8]

Therefore a capsule is a wrapper around a dedicated group of neurons. Fig. 5 shows a simplified comparison between a capsule and a neuron. A neuron computes a scalar value from a list of scalar values. Since a capsule essentially wraps a group of neurons, it computes a vector from a list of input vectors. It is now able to encode entity parameters like location, skew, etc. [20]. However this also means, that it does not represent the probability for the existence of a feature anymore. Instead the length of the vector can be used as the probability for feature existence, while not losing the important pose information. Furthermore this also enables the network to learn the parameters by itself, removing the need for crafting them by hand. This means that a n-dimensional (nD) Capsule can learn n parameters and outputs a n-dimensional vector.

For the output vector to model a probability, it's length has to stay between 0 and 1. Normal activation functions like ReLU only work on scalar values, hence a novel non-linear squashing function Eq. 1 was introduced.

$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1 + ||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||} \tag{1}$$

To understand how the inputs of the capsule are combined to s_j, we will now look into the architecture of the Capsule Network presented in [20].

3.2 Architecture

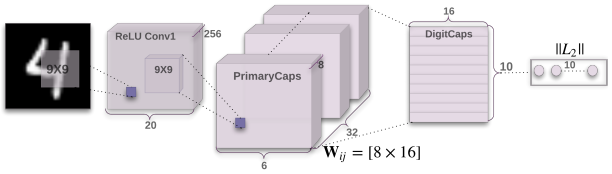


Figure 6: Capsule Network Architecture as described in [20]

The architecture in Fig. 6 shows a Capsule Network for classifying images from the MNIST dataset [13]. An input image is transformed into 10 scalar values, representing the probability for each number 0-9.

3.2.1 Conv1

The first layer applies a normal convolution with a $9 \times 9 \times 1$ kernel to the $28 \times 28 \times 1$ image over 256 channels.

3.2.2 PrimaryCaps

The next layer consists of 32 channels, each channel a 6×6 grid of so called primary capsules. They serve as a transition between the scalar values of the convolution to 8D vector outputs. The primary capsules can be seen as another convolutional layer with a $9 \times 9 \times 256$ kernel, just with squashing as their activation function. This means that the weights, i.e. the kernel, are shared between all capsules in each 6×6 grid.

3.2.3 DigitCaps

Following is the DigitCaps layer, fully connected to the primary capsules. These are now pure 16D capsules getting their inputs from the previous primary capsules. The weight matrix W_{ij} transforms the 8D output of primary capsule i to a 16D vector as input for digit capsule j ($\hat{u}_{j|i}$) Eq. 2.

$$\mathbf{s}_j = \sum_i c_{ij} \hat{\mathbf{u}}_{j|i} \ , \qquad \hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij} \mathbf{u}_i \tag{2}$$

Therefore each digit capsule has a weighted sum of $32 \times 6 \times 6$ 8D vectors as input (s_j). Instead of using pooling, the new technique of routing-by-agreement is used to focus on the most important inputs. This is discussed in section 3.3.

3.2.4 Class predictions

The 10 16D vectors correspond to the numbers 0-9 (10 classes). Because of the squashing function, the length of each of the vectors can be directly used as a probability for each class. Hence there are no more fully connected layers needed for classification (compare to Fig. 1).

3.3 Routing-by-agreement

Routing-by-agreement is a novel dynamic routing technique. In contrast to pooling, the routing happens at runtime. The goal of this technique is to redirect previous capsule outputs to a following capsule where it agrees with other inputs. In the scope of inverse graphics this can be compared to routing a detected nose to the face-capsule and not the car capsule. A detected nose, eye and mouth agree together in the face-capsule, while the nose would not agree with a wheel and door in the car capsule.

This works because of "coincidence filtering". In a high dimensional space - in this case the parameter dimension - it is very unlikely for agreements to lie close to another. So a cluster of agreements can not be, in a probabilistic way, a coincidence.

As an implementation, Hinton et. al chose an iterative clustering algorithm, see Alg. 1.

Algorithm 1 Routing algorithm. (from [20])

1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}$, r, l)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$
6: for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$
7: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
return $\mathbf{v}_j$

In simple terms, the algorithm finds the mean vector of the cluster (s_j), and weighs all inputs based on their distance to this mean (b_{ij}) and normalizes the weights with the "routing softmax" (c_i) 3.

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{3}$$

doi: 10.2313/NET-2018-11-1_12

The number of iterations r is a hyperparameter for the network and is empirically found to produce the best results around 3 to 5 iterations. When the routing is finished the input vectors now have an associated weight c_{ij} [20].

3.4 Training

A Capsule Network produces vectors as outputs. Hinton et. al proposed multiple loss functions to be used at the same time for training [20]. Reconstruction loss is used to train the capsule parameters, while margin loss is used to optimize digit classification. Both are explained below.

3.4.1 Reconstruction loss

This loss is normally used to train a neural network unsupervised. It is mostly used in autoencoders [7] to force the network to find a different representation of the data. Therefore it makes sense to train Capsule Networks the same way, to force the capsules to find adequate pose parameters.

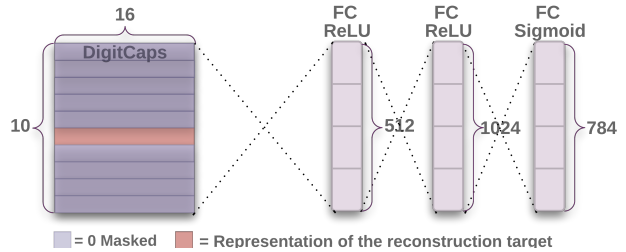


Figure 7: Decoder network, building on top of the DigitCaps layer, as described in [20]

To implement this loss, the previous architecture of Fig. 6 is extended with a decoder network (Fig. 7) to form the typical architecture of an autoencoder. The decoder network consists of two fully connected layers with ReLU activation and one sigmoid activated layer. The output of 784 values represents the pixel intensities of a 28×28 image, the same size as the images from the MNIST dataset.

As the actual reconstruction loss Hinton et. al [20] used the euclidean distance between the actual image and the sigmoid layer output. Additionally, the DigitCaps layer is masked to exactly one capsule as input for the decoder. The masked capsule corresponds to the ground thruth label, e.g. the numbers 0 to 9 as in [20]. This forces the DigitCaps to encode the actual digits [20].

3.4.2 Margin loss

$$L_k = T_k \ \max(0, m^+ - ||\mathbf{v}_k||)^2 + \lambda\,(1 - T_k)\ \max(0, ||\mathbf{v}_k|| - m^-)^2 \quad (4)$$

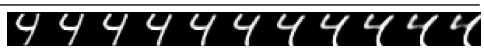
Eq. 4 shows the mathematical definition of the margin loss. In [20], the constants were chosen as $m^+ = 0.9$ and $m^- = 0.1$. T_k acts as a switch between two cases. $T_k = 1$ if a digit of class k is present $T_k = 0$ if not. This loss ensures that the output vectors of the digit capsules are at least m^+ long, when the class is detected, and at most m^- long when that class is not detected. $\lambda = 0.5$ is used to prevent the loss from shrinking all vectors in the initial learning phase.

This loss function is applied to each digit capsule individually. The total loss is simply the sum of the losses of all digit capsules [20].

Table 1: CapsNet classification accuracy. The MNIST average and standard deviation results are reported from 3 trials. Methods used were B=Baseline=CNN and C=CapsNet. [20]

Method	Routing Iterations	Rec. Loss	MNIST (%)	MultiMNIST (%)
B	-	-	0.39	8.1
C	1	no	$0.34_{\pm 0.032}$	-
C	1	yes	$0.29_{\pm 0.011}$	7.5
C	3	no	$0.35_{\pm 0.036}$	-
C	3	yes	$\mathbf{0.25}_{\pm 0.005}$	**5.2**

Figure 8: Resulting reconstructions if one of the 16D in a digit capsule is alterered marginally [20].

Scale and thickness	
Localized part	
Stroke thickness	
Localized skew	
Width and translation	
Localized part	

3.4.3 Hyperparameters

As stated in Sec. 3.4.2, the margin loss parameters were defined in [20] as $m^+ = 0.9$, $m^- = 0.1$ and $\lambda = 0.5$.

The total loss to be optimized is a weighted combination of both losses. To prevent the reconstruction loss from dominating the margin loss, the reconstruction loss is scaled down by 0.0005 [20].

Hinton et. al [20] experimented with the number of iterations in the routing algorithm 1. They empirically found 3 iterations, combined with the reconstruction loss, to produce the best results. This can be seen in table 1.

3.5 Evaluation

For evaluation, Hinton et. al generated a new dataset called MultiMNIST. For each sample two digits from the MNIST dataset were overlapped by 80% [20].

The proposed network has been tested on the MNIST and MultiMNIST dataset. The results are depicted in table 1. This shows that Capsule Networks are able to outperform the baseline CNN in both challenges, achieving significantly better results in the MultiMNIST dataset.

3.5.1 Representation of the pose parameters

To prove their goal of encoding transformation parameters in the capsule, Hinton et. al fed a capsule prediction to the decoder network (see 3.4.1. Fig. 8 displays how small changes to some of the 16D parameters affect the reconstructions. The results suggest that their goal has been reached.

When decoding the two predicted capsules, this leads to very accurate reconstructions, see Fig. 9.

doi: 10.2313/NET-2018-11-1_12

Figure 9: Correct reconstructions (lower image) of the CapsNet on MultiMNIST test dataset (upper image). L:(l_1, l_2) represents the label for the two digits in the image. R:(r_1, r_2) represents the two digits used for reconstruction. [20]

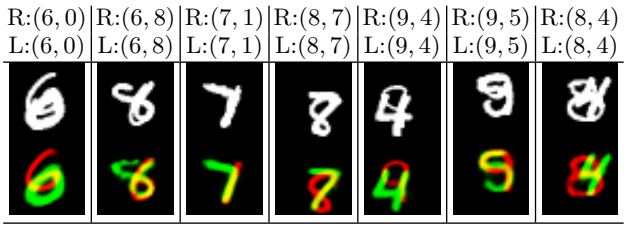
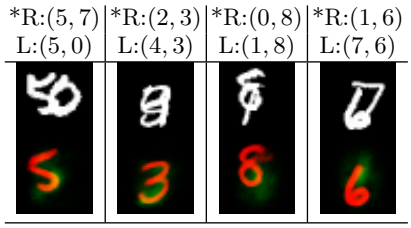


Figure 10: Capsule Network forced to reconstruct on false lables (marked with *) on the MultiMNIST dataset. [20].



3.5.2 *Smart reconstructions*

As another experiment they forced the decoder network to reconstruct non-predicted capsules (Fig. 10). It can be observed, that the CapsNet only reconstructed digits that it also detected. Hinton et. al proposed [20], that the model is not just finding the best fit for all the digits in the image. Instead it also includes the ones that do not exist. Hinton et. al suggest, that "in case of $(8, 1)$ the loop of 8 has not triggered 0 because it is already accounted for by 8. Therefore it will not assign one pixel to two digits if one of them does not have any other support"[20].

3.6 Matrix Capsules with EM routing

Hinton et. al published another paper, currently under open review, called "Matrix Capsules with EM routing" [6]. They propose to use a EM [6] algorithm instead of the current routing algorithm 1 from [20]. Additionally they changed the capsules to use a 4×4 pose matrix instead of a vector. Such a matrix is used in computer graphics to compute the scene, like it would be seen through a virtual camera. This is called the viewport. Since the network is able to learn this matrix, it is able to become viewport invariant [6].

They tested this new network on smallNORB dataset (Fig. 11) and outperformed the current state of the art CNN by 45%, reducing the error percentage from 2.56% to 1.4% [6]. Furthermore, they conducted an experiment, training the network only on specific viewpoints and testing it on unseen viewpoints [6]. Both networks were trained to the same error of 3.7% on seen viewpoints. While the baseline CNN's error increased to 20% on unseen viewpoints, the Capsule Network still achieved 13.5%. Based on these results, CapsNets seem to be able to generalize better than CNNs, being able to adapt to 3D viewports in 2D images.

Figure 11: Example images from the smallNORB dataset [6]. Multiple object classes in different viewports.

3.7 Advantages of Capsule Networks

Capsule Networks show multiple advantages compared to classic Convolutional Neural Networks. The following list with explanations is adapted and extended from [4].

3.7.1 *Viewpoint invariance*

The use of parameter vectors, or pose matrices [6], allows Capsule Networks to recognize objects regardless of the viewpoint from which they are viewed. Furthermore Capsule Networks are moderately robust to small affine transformations of the data [20].

3.7.2 *Fewer parameters*

The connections between layers require fewer parameters, since only neuron groups are fully connected, not the neurons themselves. The CNN trained for MultiMNIST consisted of 24.56M parameters, while the CapsNet only needed 11.36M parameters [20]. This is close to half as many as before. Matrix capsules with EM-routing required even less [6]. This also means that the model can generalize better.

3.7.3 *Better generalization to new viewpoints*

CNNs memorize, that an object can be viewed from different viewpoints. This requires the network to "see" all different transformations possible. Capsule Networks however generalize better to new viewpoints, because parameter information of a capsule can capture these viewpoints as mere linear transformations [7]. Therefore CapsNets are not as prone to misclassification of unseen data, as shown in Sec. 3.6.

3.7.4 *Defense against white-box adversarial attacks*

Common attacks on CNNs use the Fast Gradient Sign Method. It evaluates the gradient of each pixel against the loss of the network. The pixels are then changed marginally to maximize the loss without distorting the original image. This method can drop the accuracy of CNNs to below 20%. Capsule Networks however maintain an accuracy over 70% [4].

3.7.5 *Validatable*

A problem for industry usage of CNNs is their black box behaviour. It is neither predictable how a CNN will perform on new data, nor can its performance be properly analyzed and understood. Because Capsule Networks build upon the concept of inverse graphics, the network's reasoning can be explained considerably better than CNNs. Shahroudnejad

doi: 10.2313/NET-2018-11-1_12

et. al [21] proposed an explainability method building naturally upon capsules and their structure. This suggests, that Capsule Networks are superior to CNNs in validatability.

3.7.6 Less amount of training data

Through unsupervised learning and the dynamic routing procedure, Capsule Networks converge in fewer iterations than CNNs. Furthermore, CNNs need exponentially more training data to understand affine transformations [20].

3.8 Challenges for Capsule Networks

The CapsNet's early development stage, brings not only common problems with it, but also reveals unique challenges. In the following both kinds are listed in more detail.

3.8.1 Scalability to complex data

Hinton et. al [20] evaluated the network experimentally on the CIFAR10 dataset, failing to perform as good as current CNNs. The results were comparable to the first CNNs tackling the challenge. Matrix capsules and other discussed approaches in section 3.9 try to tackle this problem but are still far from performing on the ImageNet challenge.

3.8.2 Capsules need to model everything

As described in [20], capsules share this problem with generative models. The network tries to account for everything in the image. This also means that it performs better, if it can model the clutter like background noise, instead of having an extra "not-classifiable" category. LaLonde et. al [12] try to solve this issue by reconstructing not the whole image, but only the segmentation. This removes the need for the network to model the background and allows it to concentrate only on the active class. For solving their challenge of segmenting medical images, this approach shows promising results.

3.8.3 Structure forcing representation of entities

The concept of entities was introduced to Capsule Networks to aid in computer vision and perform inverse graphics. This network architecture could therefore prevent the network from being applied to non-vision areas. However this has not yet been investigated thoroughly.

3.8.4 Loss functions

Since the network produces vector or matrix outputs, existing loss functions cannot be simply reused. However, they can often be adapted and sometimes leverage the additional data, as can be seen in the reconstruction loss. Still, using the CapsNet on a new dataset will often require a new loss function as well.

3.8.5 Crowding

Human Vision suffers from the "crowding" problem [16]. We cannot distinguish objects, when they are very close together. This can also be observed in Capsule Networks [20], since this concept was used to model the capsule in the first place [20]. Capsules are based upon the idea, that in each location in the image is at most one instance of the type of entity that the capsule represents [20]. While this enables capsules to efficiently encode the representation of the entity [20], this could also present itself as a problem for specific use cases.

3.8.6 Unoptimized implementation

The original, but not sole, implementation of the Capsule Network can be found at [22]. It shows, that Capsule Networks present multiple challenges for current deep learning frameworks. Neural networks are often represented as a graph. Therefore, the number of routing iterations must be defined empirically [20], allowing for the $for\ loop$ to be unfolded beforehand and chained r times together in the graph. Another problem for training neural networks is that the routing algorithm is dynamic and not easy to parallelize, preventing GPUs from leveraging their full computing power. Nevertheless it is very likely that these deep learning frameworks will adapt with time.

3.9 Further improvements

The performance and scalability of the original Capsule Network has been analyzed by Xi et. al in [26]. They came to the conclusion that, apart from minor improvements, the Capsule Network does not work very well on complex data. Capsule Networks initially showed a lot of problems similar to CNNs before 2012. These were problems like unoptimized algorithms, vanishing gradients, etc. This inspired further research in this area, producing multiple different approaches for a new routing algorithm, new losses, or even complete restructuring of the architecture. Some of these publications are presented below.

Phaye et. al [17] investigated using DenseNet-like skip-connections to increase the performance of the Capsule Network. The resulting DCNet was furthermore restructured in a hierarchical manner (DCNet++), outperforming the original CapsNet.

Rawlinson et. al [18] also proposed a change in the Capsule Net architecture. By removing the margin loss (section 3.4.2) and introducing sparsity to the capsules, the network achieved similar results and generalized very well. Furthermore the network could now be trained completely unsupervised.

Bahadori et. al [3] developed a novel routing procedure. The algorithm is based on the eigen-decomposition of the votes and converges faster than EM-routing, described in [6]. The connection between the proposed S-Capsules and EM-Capsules is analogous to the connection between Gaussian Mixture Models and Principal Component Analysis. This analogy suggests why S-Capsules are more robust during the training [3].

Wang et. al [24] optimized the routing algorithm by leveraging Kullback-Leibler divergence [11] in regularization. Their proposed routing outperforms the original routing procedure in accuracy 1.

3.10 Use in real world applications

Capsule Networks are currently evaluated on challenging tasks and in difficult environment, where typical CNNs fail to produce acceptable results. In the following, three example cases are presented.

Afshar et. al [1] use Capsule Networks for brain tumor type classification. Capsule Networks require less training data

doi: 10.2313/NET-2018-11-1_12

than CNNs. This is very important in the medical environment, resulting in CapsNets being the superior network for this task.

Wang et. al [25] employ the capsule concept in a Recurrent Neural Network. Thereby they achieved state-of-the-art performance in sentiment analysis tasks.

LaLonde et. al [12] designed a new network architecture similar to U-Nets [19]. Apart from outperforming the baseline U-Net model on large 512×512 images, they reduced the parameters needed by 95.4%.

4. CONCLUSION

Hinton et. al propose with Capsule Networks a completely new way for Convolutional Neural Networks to analyze data. The routing-by-agreement algorithm 1 tackles the problems of pooling (2.2) [7]. Together with the concept of capsules, this enables networks to be viewpoint invariant and robust against affine transformations. This eliminates the main reason for huge amounts of data being needed in CNN training. Overall the new architecture holds many advantages over the typical CNN. Current research shows that, with some alterations, Capsule Networks are able to perform even in complex scenarios [1]. However, Capsule Networks have to be developed further to outperform, or even replace CNNs in real world scenarios, especially when data is not a problem.

5. REFERENCES

[1] P. Afshar, A. Mohammadi, and K. N. Plataniotis. Brain tumor type classification via capsule networks. *CoRR*, abs/1802.10200, 2018.

[2] Aphex34. Convolutional neural network - max pooling. `https://en.wikipedia.org/wiki/Convolutional_neural_network#Max_pooling_shape`; last accessed on 2018/06/14.

[3] M. T. Bahadori. Spectral capsule networks. 2018.

[4] S. Garg. Demystifying "matrix capsules with em routing.". `https://towardsdatascience.com/demystifying-matrix-capsules-with-em-routing-part-1-overview-2126133a8457`; last accessed on 2018/06/14.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] G. Hinton, S. Sabour, and N. Frosst. Matrix capsules with em routing. 2018.

[7] G. E. Hinton. What is wrong with convolutional neural nets? Talk recorded on youtube, `https://youtu.be/rTawFwUvnLE`; last accessed on 2018/06/14.

[8] K.-Y. Ho. Capsules: Alternative to pooling. `https://datawarrior.wordpress.com/2017/11/14/capsules-alternative-to-pooling/`; last accessed on 2018/08/18.

[9] T. Kothari. Uncovering the intuition behind capsule networks and inverse graphics. `https://hackernoon.com/uncovering-the-intuition-behind-capsule-networks-and-inverse-graphics-part-i-7412d121798d`; last accessed on 2018/06/14.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[12] R. LaLonde and U. Bagci. Capsules for Object Segmentation. *ArXiv e-prints*, Apr. 2018.

[13] Y. LeCun, C. Cortes, and C. Burges. Mnist dataset, 1998.

[14] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[15] Mathworks. Convolutional neural network. `https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html`; last accessed on 2018/06/14.

[16] D. G. Pelli. Crowding: A cortical constraint on object recognition. *Current opinion in neurobiology*, 18(4):445–451, 2008.

[17] S. S. R. Phaye, A. Sikka, A. Dhall, and D. Bathula. Dense and diverse capsule networks: Making the capsules learn better. *arXiv preprint arXiv:1805.04001*, 2018.

[18] D. Rawlinson, A. Ahmed, and G. Kowadlo. Sparse unsupervised capsules generalize better. *CoRR*, abs/1804.06094, 2018.

[19] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[20] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869, 2017.

[21] A. Shahroudnejad, A. Mohammadi, and K. N. Plataniotis. Improved explainability of capsule networks: Relevance path by agreement. *CoRR*, abs/1802.10204, 2018.

[22] soskek. Capsnets tensorflow implementation. `https://github.com/soskek/dynamic_routing_between_capsules`; last accessed on 2018/06/14.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[24] D. Wang and Q. Liu. An optimization view on dynamic routing between capsules. 2018.

[25] Y. Wang, A. Sun, J. Han, Y. Liu, and X. Zhu. Sentiment analysis by capsules. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1165–1174. International World Wide Web Conferences Steering Committee, 2018.

[26] E. Xi, S. Bing, and Y. Jin. Capsule Network Performance on Complex Data. *ArXiv e-prints*, Dec. 2017.

doi: 10.2313/NET-2018-11-1_12

Attack-Defense-Trees and other Security Modeling Tools

Benjamin Löhner
Advisor: Heiko Niedermayer
Seminar Future Internet SS2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: b.loehner@tum.de

ABSTRACT

A recent study [14] shows that US companies took an average of 206 days to detect a data breach. With increasingly complex computer systems and networks, mitigating such attacks quickly becomes a major task for modern organizations. Taking into account the small margin of error, security modeling and risk management tools present a viable solution to cope with this issue. In this paper, we therefore give an overview over some existing tools on security modeling. After an in-depth view of attack defense trees, we cover alternative approaches using UML as well as practical implementations in the form of the risk management software Verinice. These tools are then evaluated and compared to each other regarding various situations and questions from a practical viewpoint. Finally, each tools strengths and weaknesses are summarized.

Keywords

IT-Security, Security Modeling, Attack Defense Trees, Unified Modeling Language, Misuse Cases, Mal-activity Diagrams, Extended Statechart Diagrams, Verinice

1. INTRODUCTION

Information is one of the main assets of many organizations. In most modern businesses it is stored digitally, distributed via networking infrastructure and processed on several endpoints. If this data is accessed by unauthorized actors, the consequences can be devastating, ranging from business crises to threads with nation wide impact. A rather extreme example are the ransomware attacks on Germany in 2017, which temporarily affected public infrastructure like the national railway and hospitals internal databases [11]. It is known that its way of infection was based on the EthernalBlue exploit, which was published by WikiLeaks after a data breach at the CIA [28]. To keep the chance of an attack as low as possible, a system must be sufficiently protected. To aid this process, security models and risk management tools have been introduced. With the above risks in mind, the demand on such tools is high and manifold: First, it should not only enable the user to build a good model of the real components to be protected, but also help in finding possible attack vectors. After the surfaces are identified, it is desirable for a model to support the addition of possible defenses. Depending on the situation, a user might also want to evaluate the resulting model mathematically, generating metrics like probabilities to measure the risk of a successful attack. Depending on the situation, further, more specific questions may surface: How do we model dynamic defenses? Can the tool deal with a defense failing? In this paper, we present three tools for security modeling. Then, we compare them by a number of features, including the requirements and questions above. Finally, we evaluate their performance and suitability in practical scenarios.

2. SECURITY MODELING TOOLS

2.1 Attack defense trees

2.1.1 Concept and features of ADTs

Similar to attack trees, ADTs are trees with labeled nodes. These are split into two categories: Attacks an attacker might launch against a systems component (*attack nodes*) and countermeasures a defender employs to ensure the systems protection (*defense nodes*). While the former are drawn as circles, defense nodes are depicted as rounded rectangles. Edges represent causal relationships. By default, edges are *disjunctive*, meaning that the parent nodes attack is considered successful, if at least one of its children's condition is true. On the contrary, edges adjacent to a parent can be marked *conjunctive* by grouping them together with connecting arcs. Conjunctive edges simulate the behavior of ATs by requiring every nodes condition in the group to be fulfilled for the parents action to be successful. The distinction between attack and defense nodes induces two kinds of relationships: Edges can connect two nodes of the same category, thus decomposing them into components or different possibilities (*refinements*). Edges connecting attacks and defenses are called *countermeasures* and are highlighted by dotted lines. To simplify the structure, a node can only have one child of the opposite type, which is usually drawn as the rightmost child. Apart from these guidelines, the basic ADT can be modeled freely, including parents with edges and connected nodes of mixed kinds.

2.1.2 An example

Figure 1 shows an example ADT based on the example in Kordy's 2014 paper [18]. As noted on the trees root, the main goal is the protection of "Data Confidentiality". Indicated by the two conjunctive edges leading to the defense nodes "Network Security" and "Physical Security", a breach in either of these fields would result in data being exposed. The latter is only directly affected by the possibility of a "Break In", which is then further decomposed into various possible entry points. As any entry would result in a successful break in, disjunctive edges are used. Network security on the other hand is more multilayered, requiring multiple defense mechanisms to be active simultaneously in order to protect the companies information. Namely in the

doi: 10.2313/NET-2018-11-1_13

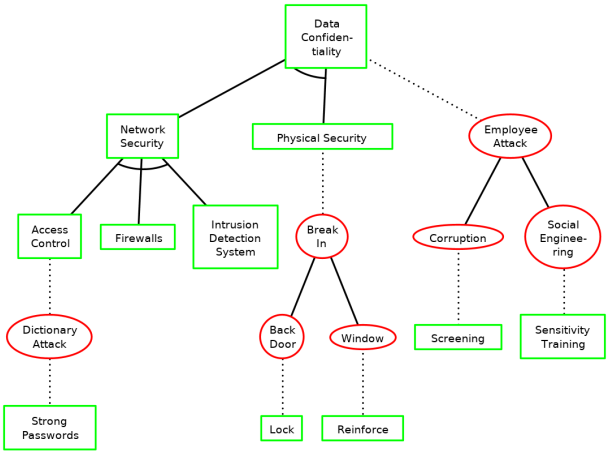


Figure 1: Example attack defense tree

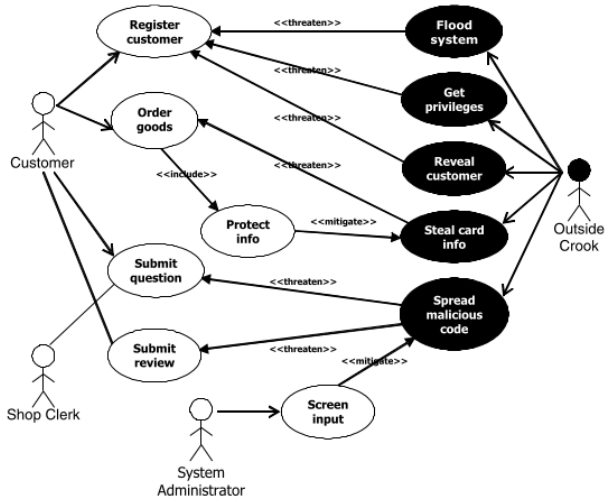


Figure 2: Example misuse case diagram

example, ”Access Control”, ”Firewalls” and ”Intrusion Detection Systems” are listed and further refined. Additionally, there could be an attack on the companies employees. This possibility is further refined into ”Corruption” and ”Social Engineering”, which respectively is protected against by ”Screening” during the recruitment process and ”Sensitivity Training” of employees.

2.2 Extended UML

Unified Modeling Language (UML) is a general purpose tool for visualizing various views and components of computer and software systems. As of 2018 it is one of the foundational modeling tools in software engineering and taught in almost every kind of software related education. Consequently, this extensive use led to various extension to make UML applicable to security modeling. Therefore in this section, we present UML-based tools for security modeling. For each tool, we first explain its design and extensions over standard UML, followed by an example and finally provide some information on the main scope and the models use cases.

2.2.1 Misuse case diagrams

Misuse case diagrams (MCDs) [26] are an extension to common UML use case diagrams (UCDs). As opposed to UCDs, which only feature neutral *actors* to interact with the system, MCDs differentiate between non-evil *actors* and *misusers*, commonly distinguished by different colored symbols. Similarily, *uses cases* are accompanied by their counterpart *misuse cases*, using the same color code. These components form nodes in a directed unweighted graph. For edges, ordinary use case relationships like *extend*, *generalize* or *include* are supported, while the security specific interactions are expressed by the new tags *threaten* and *mitigate*. A use case which has a directed edge of latter type to a misuse case is called *security use case*.

The example in figure 2 taken from Sindres paper [26] depicts use and misuse cases for an e-commerce store. In addition to the normal use cases of a system, security and misuse cases are added. For instance, the actor ”Customer” can ”Order goods”. One possible abuse would be to either intercept traffic or use other means to capture sensitive data while the user interacts with the ordering website. This is depicted by the misuser ”Outside Crook”’s use case of ”Stealing card info”, which has a threatening relationship to the ”Order goods” case. In this example, ”Order goods” includes a ”Protect info” node, which could use measures such as transport level encryption to protect against the attack, indicated by the mitigate relationship to the ”Steal card info” node. Likewise, other use cases partly involving different actors are modeled.

By design, MCDs are most suitable for modeling threads at the system boundary, providing a good overview of its attack surfaces.

2.2.2 Mal-activity diagrams

Mal-activity diagrams (MADs) [25] use the same base syntax as activity diagrams, while providing similar extensions as the misuse case models. Namely, they support *malicious actors* and *malicious activities,* distinguished by a different color. In addition, we can use *malicious decision boxes* to model decisions with malicious intent.

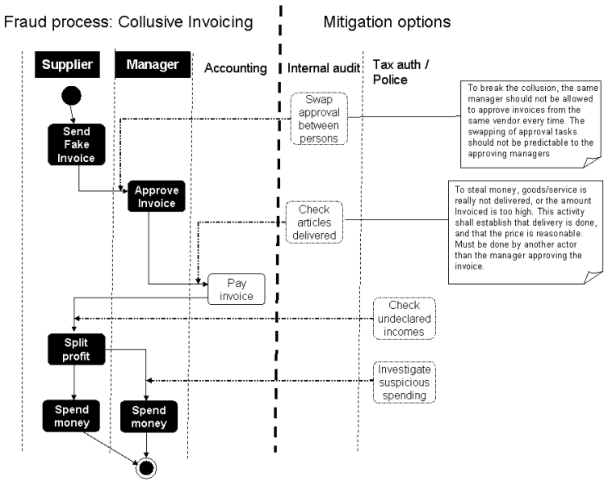


Figure 3: Example mal-activity diagram

doi: 10.2313/NET-2018-11-1_13

In the example taken from Sindres paper [25] in figure 3 on the left side, a supplier and a manager collude to get fraudulent invoices paid and spend the profit themselves. A mal-activity in the "Supplier" column labeled "Send Fake Invoice" and one edge directed to a "Manager" node "Approve invoice" model the malicious collaboration between supplier and manager. After the next neutral activity "Pay invoice" is completed by the "Accounting" department, a similar structure of mal-activities is used to describe the splitting and spending of the profit. Similarly, the defense mechanisms are depicted on the right. There, the node "Swap approval between persons" and "Check articles delivered" add new conditions to the completion of "Approve Invoice" and "Pay invoice" respectively. Their exact function and behavior is described in the constraint boxes on the very right. Likewise, "Split profit" and "Spend money" is extended by "Check undeclared incomes" and "Investigate suspicious spending" activities by the "Tax auth / Police".

MADs are more suitable for modeling attacks either inside or outside of a system with a focus on involved actors.

2.2.3 *Extended statechart notation*

To avoid symbol ambiguity, extended statechart models (ESMs) [8] introduce six new types of states. Theses are given the descriptive names *threatened state*, *vulnerable state*, *defensive state*, *compromised state*, *quarantine state* and *recovery state*, which are also used to label their depictions (e.G. <<`vulnerable`>>). In addition, they can optionally be distinguished by different color and symbols. Furthermore, the extended notation provides *thread-* and *countermeasure events* to model offensive or defensive relationships as well as *inital thread-* and *final compromised nodes* to indicate special start or final states. Like extended states, they can be colored differently. Optionally, thread event lines can be drawn dashed.

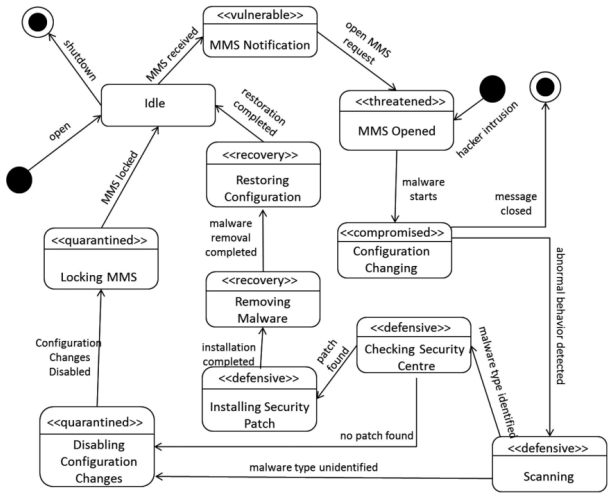


Figure 4: Example extended statechart diagram

Extensive use of the different kinds of states can be seen in the example in figure 4 taken from El-Attars paper [8]. It shows how Android phones are attacked by the "AndroidOs.FakePlayer" malware and what defense mechanisms can be used to treat such an attack. It first starts by describing the way of infection, starting at the initial node on the left. After the phone receives an MMS, it transitions from "Idle" to an vulnerable state, as it display the "MMS Notification". As of this time, the system does not get infected as long as the user leaves the notification unused. If it is opened however, we transition the edge "open MMS request" to the threatened state "MMS Opened". This is the point where the hacker actually gains access to the device, indicated by a new initial node "hacker intrusion". From there on, the "malware starts", leaving the device in a compromised state, after its "Configuration Changed". From there on, different security measures are modeled, following the same principles as above. If abnormal behavior is detected, the device launches a defensive security scan. If it can identify the malware, it transitions through multiple defensive states and finally reach recovery states where it removes the virus and restores the devices original state. In the case of no successful identification, it continues to transition through quarantined states to disable to misbehaving components. Either way, the thread is mitigated and phone goes into "Idle" again.

ESMs are particularly useful to model one single process in great detail.

2.3 Models in practical risk management tools - Verinice

In the industry, the models presented are most of the time not used in their theoretical forms. Instead, their concepts are incorporated into risk management tools, which include common scenarios and defenses, as well as providing ways to automate and simplify some of the proof- and risk probability analysis tasks.

In the following section, we show the parallels between theory and practice by describing some key aspects of the application Verinice [10].

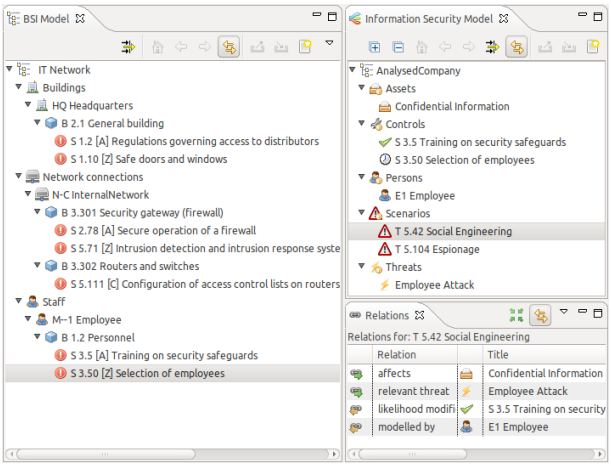


Figure 5: Example of models created with Verinice

Even though Verinice internal data structures are not visible, we can distinguish two modes of operation. In the "BSI Model" we can map scenarios from the German IT baseline catalog (BSI) [9] to our business structure. This model is especially useful to find defenses that are commonly used. The insights gained in this mode can then be used to generate

doi: 10.2313/NET-2018-11-1_13

”to do” lists to aid the implementation of those measures.

In figure 5 on the left hand side, we can see how that model looks in practice. The company is decomposed into common attack vectors like ”Buildings”, ”Network Connections” and ”Staff”. Each of that categories is then refined into individual instances like ”Headquarters”, which contain measures to be taken to ensure their safety. In this case, the entries ”5.1.2 [A] Regulations governing access to distributors” and ”5.1.10 [Z] Safe doors and windows” from the BSI catalog are chosen. Optionally, own company specific defense could be added. Similarly, ”Network Connections” and ”Staff” is further refined with specific instances and mitigation strategies.

The second mode of operation, the ”Information Security Model” (ISM) is used to model relationships between business entities like assets or employees, attack scenarios and defenses. It can then be used to calculate detailed reports about the business security state, including risk possibilities and costs in case of attacks. To create reports close to reality, Verinice supports a wide selection of attributes for each of the entities, as well as many different kinds of relationships.

To get an idea of the ISM, figure 5 shows an example on the right hand side. The ”AnalysedCompany” contains the asset ”Confidential Information” and the employee ”E1 Employee”. In addition, it implements the controls ”T 5.3.5 Training on security safeguard” and ”T 5.3.30 Selection of employees” from the BSI catalog. Furthermore, the scenarios ”T 5.42 Social Engineering” and ”T 5.104 Espionage” are possible. These entities are connected by using relationships. One example are the relationships related to the social engineering scenario, as shown in the bottom right edge of figure 5. There, the malicious social engineering scenario is shown to be one member of the relevant thread ”employee attack”. It is modeled by the person ”E1 Employee” and directly affects the asset ”Confidential Information”. Besides the attack side, there also is a defense. As shown in the relationships panel the likelihood of this kind of attack is reduced by the control measure ”Training on security safeguards”. Similarly, the scenario espionage is modeled.

3. COMPARISON

After we briefly introduced common security modeling tools, this section compares and evaluates their usefulness in answering practical questions.

In general, Verinice is a practical tool build for use in industry, while the other models are developed from a more academic perspective. This becomes even more obvious considering the use cases: On one hand, Verinice is well-suited for aiding a companies general decision making in instances like asset management, where the value of an asset and the cost of its protection need to be compared to each other. On the other hand it can actually suggest measures to ensure the security of systems and devices. In many situations, this can be done by employees with limited training, as many steps of the vulnerability and defense finding process are automated by external tools or simplified by Verinice's databases.

On the contrary, the UML-based modeling tools and ADTs are better-suited to gain insight into the attack vectors of possibly new systems. They provide the means to structurally examine systems on different levels and visualize their components and their weaknesses. The resulting abstract models can then be used to find possible defenses or evaluate the efficiency of known countermeasures. It is therefore mainly used in academia.

To provide more detailed insight, the following subsections compare the tools in their capability to solve practical problems.

3.1 Discovering attack vectors and defenses

One of the main use cases of security modeling tools is to map a real word situation and its defenses in order to find uncovered or not sufficiently protected components.

ADTs are very suitable for this purpose, as they enable a user to start modeling the system at a very generic degree, which can then be refined to an arbitrary level of detail. Each of the resulting subcomponents can then be evaluated by their current state of protection, leading to quite a complete analysis of the system.

UML on the other hand does not offer multiple levels of detail in one model. Instead, a user first needs to select the scope of his analysis. If he wants to keep an overview over the full system and possible loopholes on the system boundary, MCD are most suited. Close views of single components or processes can be modeled with MADs and ESMs. While the former is more useful to model interactive processes involving multiple actors, the latter provides stronger focus on the process itself. Even though MADs and ESMs could be used to discover exploitable components, they are more suited to develop defenses, once the possible attacks are known. To fully analyze a system, a combination of multiple of above types is often needed, making the approach less structured than using ADTs.

Verinice faces the problem that it provides almost no visual representation of relationships between attacks, defenses and system components. As the tree like structure and a word-based search are the only ways of navigating the provided data, manual attack discovery is complex and important links are easily overlooked. The main strengths of Verinice in that area are therefore its features as a practical software tool. For instance, it enables a user to directly import data automatically generated by vulnerability scans on the network or the connected hosts. Besides the ability to find loopholes by itself, Verinice offers the functionality of importing databases of common scenarios. The data can either be used to automatically add suitable defenses and relationships to the vulnerabilities discovered before, or can lead the user through a catalog of commonly found attack vectors, who can then add them and their countermeasures to his security model. The databases are based on the German baseline security catalog [9] or the ISO 27000 standards [15], adding the general advantages of standardized procedures. Besides that, especially for non-standard systems, non-IT components or very process-based situations, above features can not be utilized. Therefore, Verinice is only partly suited for attack discovery.

3.2 Dealing with failing defenses

Due to their static nature, one central question is how well the presented tools can model measures after defenses fail.

With ADTs, the closest such feature are multiple disjunctive child defense nodes. If one of these nodes fail, the parent system is still protected by the other remaining defense nodes. Dynamic processes of the form "if defense A fails, activate measure B" can not be expressed, thus significantly limiting ADTs capabilities for that kind of situation. One proposal for an extension of attack trees to include this feature are Boolean logic Driven Marcov Processes [22].

As opposed to ADTs single-model approach, UMLs multi-layered modeling is able to better capture such situations. Like ADTs, MCDs static nature make them unsuitable for that task. Instead, MADs and ESMs can be used to add more detail to selected processes. While activity-diagrams and the presented extensions support conditional nodes to choose different paths, ESMs can be used to express parting ways by drawing multiple outgoing edges for an activity.

Verinice on the other hand does not support any kind of event sequences, rendering its model more limited than UML. It makes up for that shortcoming with its feature-richness regarding different relationships and accurate metrics. While the user can specify an attack's impact on availability, confidentiality, integrity and overall value of an asset, defenses can be modeled to reduce the weight of such consequences. Alternatively, they can be altered to decrease the likelihood of a specific attack's success. As a result, Verinice can approximate an attack's probability and simulate the business state after its success, including the expected usefulness of other relevant defenses. It is therefore still superior to ADTs in evaluating consequences of a failing defense.

3.3 Prioritizing defenses and dependencies

In practice it is often desirable to rank priorities. This is useful if decisions regarding the implementation of a defense need to be made and factors like cost, usefulness or impact on business operation are compared.

ADTs do not provide direct means of ranking of any kind. The only way to model an order is by using the refinement feature, which can be used to express the dependency of a parent node on the success of its child nodes. Priorities between direct children of a node are not possible. For attack trees, Baca and Peterson proposed a notion using integer annotations to rank the attacks- and countermeasures mitigation impact [3], which might be transferable to ADTs.

While the presented extension to UML do not directly support ranking, the UML base notation does. By using UML constraints, any kind of dependency can be informally annotated, including notes on prioritization with any of the above factors. As these notations do not carry any formal meaning, their use is limited to aid a discussion of an attack's or defense's suitability, but are useless for any kind of automatic mathematical processing or proofs.

As already mentioned in "Dealing with failing Defenses", Verinice supports weighting of assets, relationships, attacks and defenses. Thus, limited prioritization within the boundaries of Verinices supported metrics is possible. In this case, automatic processing and report generation from those values is supported, making it partly superior to UML.

Table 1: Strengths and weaknesses of the presented tools

Model /Tool	Strenghts	Weaknesses
Attack defense tree	• Different levels of detail in one model • Mathematically thoroughly defined	• Limited to static processes • Unprioritized defenses
UML	• Widely used and easily learnable • Allows for dynamic process modeling • Multiple models for different levels of detail	• Often requires multiple models for complete analysis • Mostly useful for visualization, no mathematical foundation • Only informally prioritized defenses
Verinice	• Automatic analysis via external tools • Access to standardized scenario catalogs • Report generation including accurate metrics • Prioritizing of defenses possible	• Steep learning curve • Time consuming for non-standard situations • Almost no visualization of relationships

3.4 Strengths and weaknesses

To complete the comparison, a brief overview of the presented tools is shown in table 1. The row labeled UML contains information on the collective use of misuse cases, mal-activity diagrams and extended statechart diagrams.

4. RELATED WORK AND FURTHER READING

ADTs base their ideas on other concepts, including fault trees [29], threat trees [2, 30] and attack trees [24]. The latter was then extended by various researchers as summarized by Piètre-Cambacédès and Bouissou [22]. Many approaches extend the features and semantics of ADTs. In a theoretical context, the relation between ADTs and game theory has been analyzed [17]. ADTs computational aspects have been studied in [19], where semantics based on De Morgan lattices were used. Furthermore, ADTs quantity capabilities were evaluated in a practical case study [4].

In addition to the UML models presented in this paper, other approaches were proposed to enable secure software engineering, including UMLSec [16] and SecureUML [21]. As a contrast to MADs, there was an attempt to directly model secure business processes, concentrating more on modeling a process after security problems and countermeasures have been found [23]. Similar to MCDs and MADs, abuse frames are another example of an approach to system

doi: 10.2313/NET-2018-11-1_13

exploitation from the perspective of an attacker [20]. Misuse cases have also received a number of extensions, like specialization and generalization support [27]. Beside the theoretical perspective on UML, there has also been various tests on UML in realistic scenarios [1, 7, 12].

As Verinice is a practical tool, academic literature beyond its short mention is limited. Consequently, information can be found concerning its data sources ISO-27000 and the BSI baseline catalog [5, 6, 13].

5. CONCLUSION

In general, there is no tool to fit every situation. Verinice has strengths in its automatization capabilities and the high accuracy of its generated output, but it can only be used to model certain standard scenarios. UML on the other hand provides a high degree of freedom and flexibility while being easy to learn, but lacks a rigid formal foundation. ADTs form a compromise: While unifying multiple levels of accuracy in one model, they still provide a formal structure to work with. In practice, a combination of above tools may be employed. ADTs could be used as an overview to find attack vectors in top-level components, while detailed vulnerability discovery on process level might be done with MCDs or MADs. The resulting information could then be fed into Verinice to calculate actual attack probabilities and cost impact. Either way the presented models form a good base to thoroughly analyze a system to identify and later evaluate possible defenses, making risk management easier and general system security better.

6. REFERENCES

[1] I. Alexander. Initial industrial experience of misuse cases in trade-off analysis. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pages 61–68. IEEE, 2002.

[2] E. G. Amoroso. *Fundamentals of computer security technology*. PTR Prentice Hall New Jersy, 1994.

[3] D. Baca and K. Petersen. Prioritizing countermeasures through the countermeasure method for software security (cm-sec). In *International Conference on Product Focused Software Process Improvement*, pages 176–190. Springer, 2010.

[4] A. Bagnato, B. Kordy, P. H. Meland, and P. Schweitzer. Attribute decoration of attack–defense trees. *International Journal of Secure Software Engineering (IJSSE)*, 3(2):1–35, 2012.

[5] K. Beckers, H. Schmidt, J.-C. Kuster, and S. Faßbender. Pattern-based support for context establishment and asset identification of the iso 27000 in the field of cloud computing. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, pages 327–333. IEEE, 2011.

[6] I. BSI. Baseline protection manual. 2000.

[7] F. den Braber, T. Dimitrakos, B. A. Gran, K. Stølen, and J. Ø. Aagedal. Model-based risk management using uml and up. *Issues and Trends of Information Technology Management in Contemporary Organizations*, pages 515–543, 2002.

[8] M. El-Attar, H. Luqman, P. Kárpáti, G. Sindre, and A. L. Opdahl. Extending the uml statecharts notation to model security aspects. *IEEE Transactions on Software Engineering*, 41(7):661–690, July 2015.

[9] B. für Sicherheit in der Informationstechnik. Bsi-grundschutz katalog. Available at `http://www.bsi.de/gshb/deutsch/index.htm` (Aug. 2018), 1996.

[10] S. GmbH. Verinice website. Available at `https://verinice.com/` (Aug. 2018).

[11] Heise. Ransomware wannacry befällt rechner der deutschen bahn. Available at `https://www.heise.de/newsticker/meldung/Ransomware-WannaCry-befaellt-Rechner-der-Deutschen-Bahn-3713426.html` (Jun. 2018).

[12] S. H. Houmb, F. Den Braber, M. S. Lund, and K. Stølen. Towards a uml profile for model-based risk assessment. In *Critical systems development with UML-Proceedings of the UML'02 workshop*, pages 79–91, 2002.

[13] E. Humphreys. *Implementing the ISO/IEC 27001 information security management system standard*. Artech House, Inc., 2007.

[14] P. Institute. 2017 cost of data breach study: Global overview. Available at `https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=SEL03130WWEN` (Aug. 2018).

[15] Information technology – Security techniques – Information security management systems – Overview and vocabulary. Standard, International Organization for Standardization, Geneva, CH, Feb. 2018.

[16] J. Jürjens. *Secure systems development with UML*. Springer Science & Business Media, 2005.

[17] B. Kordy, S. Mauw, M. Melissen, and P. Schweitzer. Attack–defense trees and two-player binary zero-sum extensive form games are equivalent. In *International Conference on Decision and Game Theory for Security*, pages 245–256. Springer, 2010.

[18] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer. Attack–defense trees. *Journal of Logic and Computation*, 24(1):55–87, 2014.

[19] B. Kordy, M. Pouly, and P. Schweitzer. Computational aspects of attack–defense trees. In *Security and Intelligent Information Systems*, pages 103–116. Springer, 2012.

[20] L. Lin, B. Nuseibeh, D. Ince, and M. Jackson. Using abuse frames to bound the scope of security problems. In *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, pages 354–355. IEEE, 2004.

[21] T. Lodderstedt, D. Basin, and J. Doser. Secureuml: A uml-based modeling language for model-driven security. In *International Conference on the Unified Modeling Language*, pages 426–441. Springer, 2002.

[22] L. Pièvre-Cambacédès and M. Bouissou. Beyond attack trees: Dynamic security modeling with boolean logic driven markov processes (bdmp). In *2010 European Dependable Computing Conference*, pages 199–208, April 2010.

[23] A. Rodríguez, E. Fernández-Medina, and M. Piattini. Capturing security requirements in business processes through a uml 2.0 activity diagrams profile. In *International Conference on Conceptual Modeling*, pages 32–42. Springer, 2006.

[24] B. Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[25] G. Sindre. Mal-activity diagrams for capturing attacks on business processes. In P. Sawyer, B. Paech, and P. Heymans, editors, *Requirements Engineering: Foundation for Software Quality*, pages 355–366, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[26] G. Sindre and A. L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1):34–44, Jan 2005.

[27] G. Sindre, A. L. Opdahl, and G. F. Brevik. Generalization/specialization as a structuring mechanism for misuse cases. In *Proceedings of the 2nd symposium on requirements engineering for information security (SREIS'02), Raleigh, North Carolina*, 2002.

[28] F. times. Leaked cia cyber tricks may make us wannacry some more. Available at `https://www.ft.com/content/a7a6c91c-3a35-11e7-ac89-b01cc67cfeec` (Jun. 2018).

[29] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. Fault tree handbook. Technical report, Nuclear Regulatory Commission Washington DC, 1981.

[30] J. D. Weiss. A system security engineering process. In *Proceedings of the 14th National Computer Security Conference*, volume 249, pages 572–581, 1991.

doi: 10.2313/NET-2018-11-1_13

Identity Management on the Blockchain

Julian Roos
Advisor: Heiko Niedermayer
Seminar Innovative Internet Technologies and Mobile Communications
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: ga58moy@mytum.de

ABSTRACT
This paper gives an overview over six of the most promising identity management systems that use the blockchain: Sovrin, Jolocom, uPort, ShoCard, Blockstack and Namecoin. It shortly introduces them, briefly explains their design, evaluates them, assesses in how far they fulfill the promises they make and lastly lists their current work and future goals. In the end, all systems are compared with each other regarding some of their properties.

Keywords
Blockchain, Identity Management, Identity Management on the Blockchain.

1. INTRODUCTION
Identity management can be defined as a system to identify, authenticate and authorize individuals or - more generally - identities. It has been an important topic for a long time on and off the internet and with the future of Internet of Things (IoT) devices it seems to become more important. There are a few predictions on how the identity management of the future may look like. The most promising ones rely on the blockchain technology, introduced by Satoshi Nakamoto in 2008 [12]. With the use of the blockchain technology it becomes possible for identity management systems to disprove Zooko's triangle [27]. Zooko's triangle is a theory by Zooko Wilcox-O'Hearn, published in October of 2001, in which he states that three desirable properties for names of participants in a network protocol cannot all be fulfilled at the same time. Those properties are Human meaningfulness, meaning the names are human readable and memorable, security and lastly decentralization.
This is due to the fact that the blockchain technology offers a different approach to storing and managing (digital) identities, mainly because it enables decentralized identity storage without a central authentication authority while preventing tampering with the identities and the data stored. While some proposals use their own blockchain to implement their identity management others rely on already existing ones. Nevertheless, most of those new identity management techniques offer a variety of use-cases and can be used for governments, managing accounts on websites or a new domain name system. For example, they can be used to verify a user's age on multiple websites, without the user having to send each website a photo of his passport.

In this paper we explain, evaluate and compare six identity management systems. The structure of this paper is as follows: Each identity management system is described in its own section and is introduced by a short overview and some of its properties. Then we explain its design in the next subsection. The last subsection begins with an evaluation whether the technology fulfills its promises if it listed some, followed by problems and advantages the technology has and lastly lists their current work as well as their future goals (again only if they are listed). At the end of this paper we then compare the identity management systems with each other over some predefined properties.

2. SOVRIN
Sovrin [21] is a public open source identity network that is built on permissioned distributed ledger technology (DLT). Because it is public everyone can use it and due to it being a permissioned DLT nodes that form the consensus have to be authorized by the Sovrin Foundation. Sovrin enables its users to have a self-sovereign identity [22], which means that the users have lifelong full ownership of their identity and do not rely on any central authority to store it Additionally, the identity is private which means they can manage it themselves and can choose to whom they reveal what information (and to whom not) [3] [1].

2.1 Design
A Sovrin identity uses decentralized identifiers (DID) [6] to enable the identity and binds them to a user by using asymmetric cryptography. The DIDs require no central authority to be issued and enable users to create identifiers that are permanent, globally unique and cryptographically verified while the identity owner maintains full control over those identifiers. Those DIDs are then put onto a blockchain together with a DID document object (DDO) comprising the identity owner's respective public key (for this DID), other information the identity owner wants to reveal as well as the network addresses needed for interaction. The identity owner owns the DDO by having the respective private key. Therefore, everyone with access to the internet can verify his control of the private key and consequently of the DID. Additionally, there is no limit of the number of DIDs a user can have, so one can create as many as needed to ensure privacy.

2.2 Evaluation
On their website the Sovrin Foundation claims that "Sovrin identities will lower transaction costs, protect people's personal information, limit opportunity for cybercrime, and

doi: 10.2313/NET-2018-11-1_14

simplify identity challenges in fields from healthcare to banking to IoT to voter fraud" [22].

The argument for lower transaction cost is that since the seller can have more information about a customer he can reduce his risk and therefore his price. An example of this could be a car renting firm, where the customer can show his driving record i.e. the record that shows all received punishments that are driving related if he has it attached to his identity and wants to show it. The car renting firm's employee could now check whether the customer has any major driving violations or if (supposedly) is a safe driver. If the customer appears to be a safe driver, the risk for the car renting firm reduces and therefore it can offer a better price. This only works if the verifier trusts the authenticity of the information, otherwise the price will stay the same. The protection of people's personal information is also given, since people themselves can choose what to share. If someone does not want to share anything, one is not forced to do so and therefore one's privacy is not infringed. However, it is possible that for example sellers will not sell anything if a user does not expose her name. In this case the user is not entirely forced to reveal it, but in order for her to get the item she needs to do that.

The claim that it limits the opportunities for cybercrime can be linked to the fact that Sovrin improves the identity and access management. This results in less accounts being stolen and used for cybercrime. Furthermore, Sovrin enables a seller to only sell to buyers that have shown him their identity and whose identities are validated by someone the seller can trust. In that case, the chance of a buyer committing fraud is mitigated since he can be identified easily. Therefore, also this claim is valid.

Lastly, the Sovrin foundation say that Sovrin will simplify identity challenges in various areas. This mainly depends on the amount of people that use Sovrin identities. If it surpasses a critical mass it becomes suitable in those areas to include Sovrin identities to face their challenges. If it only has a relatively low number of users it may not be useful to include it. Nonetheless, one can say that the possibility to store identity attributes that are hidden until their owner chooses to reveal them offer some great possibilities in those areas.

In conclusion, Sovrin does not promise false claims and most of them can be backed, however, the last one seems to be a bit exaggerated considering that it predominantly depends on the number of users Sovrin has.

3. JOLOCOM

Jolocom [10] is an open source project that is currently developing an identity management system that uses hierarchical deterministic keys (HD keys) to offer decentralized identity to its users. Jolocom identities are also self-sovereign.

3.1 Design

The HD keys are generated from a known seed and controlled by the users [10] and enable the user to generate further child keys from the parent key [17], which can be recovered later if the seed is known. This enables users to generate multiple "personas" which are basically sub identities that can offer anonymity in some interactions. In addition, the child keys make it possible to have ownership of IoT devices mapped onto the Jolocom identity. However, an implementation of an identity management system that just uses HD keys has usability issues [10], therefore Jolocom combines HD keys with DIDs. A DID is derived from the user's public key and the corresponding DDO is stored on an InterPlanetary File System (IPFS) which is a decentralized distributed file storing system. Then the mapping of the DID to the hash of the DDO's IPFS will be stored in a smart contract on the Ethereum blockchain.

To store identity attributes Jolocom uses so-called "verified credentials" that are files that contain a statement and signature by the verifier identity. An example used in the Jolocom whitepaper for verified credentials is: "This user is employed by Company X" as a statement and a signature containing metadata about that statement by the verifier identity - in this case Company X.
For storing statements, the authors distinguish between private and public verified credentials as well as the need for constant availability. Private verified credentials usually require access control and are stored on self-hosted servers when they should be constantly available and on the user's device when they do not have to be constantly available. Public verified credentials with constant availability should be stored on distributed storage, for example IPFS. In the case of non-constant availability, the authors list no storage proposal because there is "no apparent use case for this configuration" [10].

3.2 Evaluation

Besides the claim for self-sovereign identity Jolocom does not promise any additional benefits of its technology - neither in the paper nor on its website. Since Jolocom already has a working application it therefore fulfills its claims.
However, there is one thing that Jolocom does not do (and does not claim to do) that is not entirely clear: In their conclusion the authors write: "We offer a truly self-sovereign decentralised digital identity solution" [10]. This could make someone believe that Jolocom eliminates all need for trusted third parties (TTPs). But this is not the case since Jolocom does not remove the need for a TTP when it comes to authenticating the attributes and getting verified credentials. An attribute itself is worthless unless it got verified by someone who is trusted - so a TTP. The easiest example is that of age: Anyone could make a Jolocom identity and claim to be 18 years or older to be able to buy things restricted by age. The claim itself is worthless, unless a government agency, bank or other TTP signs it, thus making it a verifiable credential.

Their future goals comprise providing a simple, global and self-sovereign identity for everyone while continuing to use the existing technical standards regarding decentralized digital identities and keeping open source releases. Furthermore, the implementation of the storing of private verifiable credentials that should be constantly available is not a priority right now, and also a future goal.
Currently Jolocom is focusing on the storing on private verified credentials that are saved on the user's phone and the public verified credentials.

doi: 10.2313/NET-2018-11-1_14

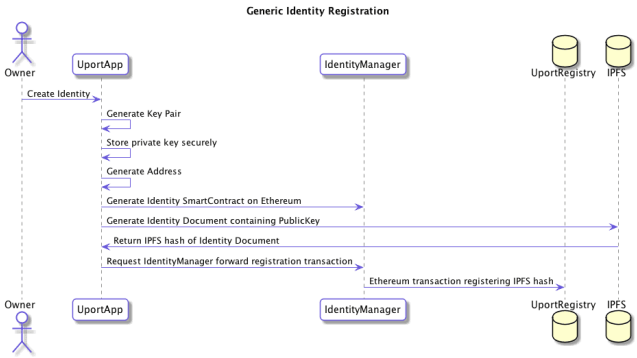


Figure 1: Identity creation process. Retrieved from [24]

4. UPORT

uPort is an open source identity management system that is used on the decentralized internet [26]. It claims to enable users to have a self-sovereign identity registered on the Ethereum blockchain which can then be used for some applications like the usage of credentials or managing of data. Currently, uPort exists only as a mobile application [25].

4.1 Design

To implement its self-sovereign identities uPort uses smart contracts on the Ethereum blockchain. Smart contracts are stored on the Ethereum blockchain and are a piece of code that can move ether (Ethereum's cryptocurrency) as well as data when invoked [8] [7]. To be executed they have to be called by their unique 160-bit hexadecimal identifier [7]. uPort uses two smart contracts on the Ethereum blockchain that are called controller and proxy.
The whole identity creation process is depicted in figure 1. To create an identity, an asymmetric key pair is created and then an instantiation of the controller with a link to the just created public key is made. After that a new proxy is created with a link to that controller instance and only that controller instance is able to invoke the proxy's functions. This can be seen in figure 2.The uPort identifier (uPortID) is now the address of the newly created proxy [7]. There is no limit to the number of uPortIDs a user can have except the number of possible 160-bit identifiers of the proxy smart contract, which obviously is 2^{160}. Furthermore, all created uPortIDs for a user are unlinkable if one does not link them oneself. To store identity attributes a global smart contract called registry is used. The registry stores a hash of the JSON attribute structure together with a uPortID. To access the attributes, which are stored outside of the blockchain on an IPFS, the stored hash is used.

Because the smart contract is global and every hash for every uPortID is stored there everyone can access the attributes of a uPortID. However, only the owner of the uPortID can change its own stored attributes. This system enables uPort to also support verified credentials.

Because the private key is only stored on the user's mobile phone which can get lost, uPort has a system that enables its users to regain control of their identity with a different key. The system works as follows: The user nominates a group of

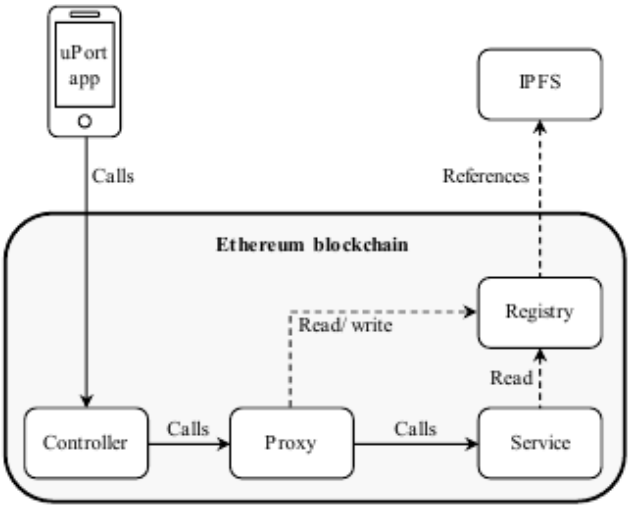


Figure 2: Interactions between smart contracts. Retrieved from [7]

```
{

   "@context": "http://schema.org",
   "@type": "Person",
   "publicKey": "0x044c31ed1499dce76ee7711c7238...",
   "publicEncKey": "Py+NXzHgacNMTzj9Ufe4S2KPuzR...",
   "name": "First Last"

 }
```

Figure 3: Example of a uPortID stored as a JSON attribute structure. Retrieved from [24]

trustees (with their uPortID) in advance. The trustees can then vote to change the associated public key of the user's uPortID with a new one and the key will be changed once a quorum is reached. Note, that there is no way for the system to identify whether the private key was truly lost. Therefore, it is possible for malicious trustees to take over a uPortID if they get quorum.

4.2 Evaluation

The system uPort offers is working since it is based on the already working Ethereum smart contracts. One is able to create an identity, manage it together with its respective attributes and can get control back in the case of loss due to the trustee system. However, the trustee system can be exploited if the uPortID has too many malicious trustees. Nevertheless, the system is working since there is no other way to regain control of a uPortID and it is the user's fault when he nominates too many malicious trustees. If uPort had not introduced the possibility to regain control of a lost uPortID, then it would not be a good identity management system and possibly be flooded by "dead" accounts which no one has control over anymore. Although, this is still possible, if a user loses his private key and has not named any trustees before.
Additionally, like Jolocom, uPort also uses verified credentials, therefore, a user is still depended on TTP. Because

doi: 10.2313/NET-2018-11-1_14

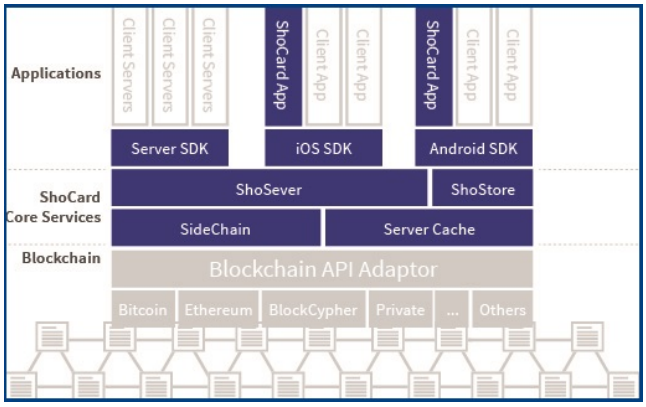


Figure 4: ShoCard's architecture. Retrieved from [19]

uPort has no promises other than a working uPort system, uPort meets all its requirements.
Currently uPort is working on the development of a mobile SDK, so that uPort accounts can be implemented in every app.

5. SHOCARD

ShoCard [20] [19] combines identity management on the blockchain with already trusted credentials like passports or driver licenses. The identities can then be extended with further attributes. Furthermore, is ShoCard the only identity management system in this paper that uses its own server to store relevant information for the identification of users (although ShoCard has no access to those because they are encrypted). An advantage of ShoCard is, that it is able to use multiple blockchains at the same time and can add new ones if needed [19].

5.1 Design

ShoCard's architecture can be seen in figure 4. To create a ShoCard identity or ShoCardID a user needs the ShoCard mobile application. The app generates an asymmetric key pair and then the user has to take a picture of the user's passport as well as a picture of himself [18]. Afterwards the selfie and the passport data are encrypted together and then stored locally on the user's device. Furthermore, the passport data and selfie are each hashed, signed (with the just generated private key) and then put together into a seal record. The seal record is then put onto the blockchain and can then be retrieved through its unique address on the blockchain. This unique address is at the same time the user's ShoCardID.

Now comes the certification part in which the information of the user gets certified. In the context of paper [18] this is done in the context of air traveling by an airline agent or automated kiosk. The user now has to show that he has control over the seal stored on the blockchain as well as the key that signed it. For this the user has to show the original data that were used to create the seal and sign a challenge to show ownership of the private key [7] [18]. Moreover, the user has to present his identification (passport, driver license, etc.) to an agent who will check that the official information of the identification matches the ones of the ShoCardID. The agent will further check if the selfie shows the user. If everything is right, the agent will request a certification of the user that certifies the ShoCardID and selfie with the airline's private key. Note that if the checking is done by an automated kiosk and without a human agent, it will look a bit different. Nevertheless, the same things will be checked and the user will also be certified at the end. Afterwards, a message containing a reference to the ShoCardID, the certification records and some further information is created, signed with the user's private key, encrypted with a new symmetric key and then stored on the ShoStore servers. This is called enveloping and the stored data can be accessed with an EnvelopeID. During this process ShoCard never learns the symmetric key and therefore only the user is able to share the data.

If the user now needs to be verified, he can generate a QR-code containing the EnvelopeID as well as the symmetric key. The agent now scans the QR-code and the envelope is downloaded from the ShoStore server and then decrypted when the download is finished. The now following verification consists of i) comparing the private key used for signature on the envelope as well as the seal record, ii) comparing the passport information and selfie versus the seal record and lastly iii) validating the signing of the certification records [18].

5.2 Evaluation

On its websites ShoCard states in the about ShoCard section "It's the one identity verification system that works the way consumers and businesses need it to for security, privacy, and always-on fraud protection" [20]. Even though the ShoCard system is working, it relies on many trusted authorities to be enabled. For example, in the use case of air travel one needs to trust every airline since every airline can certify information for any user. This can become a big problem when there is no knowledge about the authenticity of an airline or if it actually exists. Because in the latter case attackers could fabricate an airline and then sign their own information while they could bribe smaller but real airlines in the former one. This problem can be avoided if one does not trust any other airlines or diminished when airlines just trust big airlines. However, the first case would make the technology completely useless while the latter one would shrink its use by a bit, depending on the number of passengers that fly with one of the non-trusted airlines.

Another aspect is the storage of the envelope on the ShoStore servers. If the servers are down for whatever reason, the validation of a ShoCardID cannot be done which basically renders ShoCardIDs useless during that time. This becomes extremely problematic in the case of a permanent shutdown of the servers (e.g. if the company stops existing) because then ShoCardIDs are straight up useless.

A huge advantage of ShoCard is the fact that it can use multiple blockchains and change to new ones if it needs to. This is due to most other systems relying on one (specific) blockchain and would cease to exist if that blockchain would stop existing as well or if it somehow got rendered useless by a new attack.

doi: 10.2313/NET-2018-11-1_14

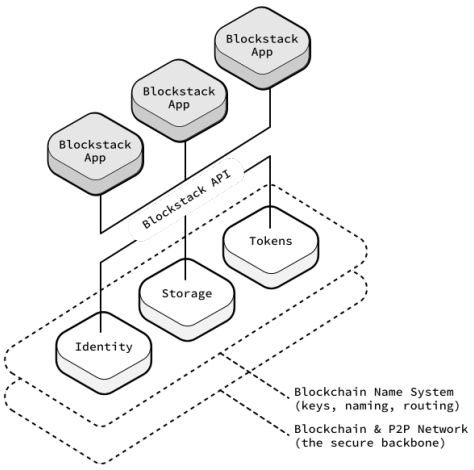


Figure 5: Blockstack's architecture. Retrieved from [5]

Concluding, one can say that while ShoCard is working, there are other identity management system out there that can also verify a user's information. The problem whether one can trust the verifying instance remains the same in all those identity management systems, including ShoCard. With the central server needed for ShoCard and the resulting availability problems it does not look all too promising for ShoCard. On the other hand one has to admit that ShoCard is focusing on the consumer business interaction while the other systems have a more general approach for identity storage. In the end it is up to the businesses if they want to risk having an unusable system in the case of ShoStore being unavailable for the advantage that ShoCard is specifically being developed for that area as well as being independent from one specific blockchain.

6. BLOCKSTACK

The main goal of Blockstack is a decentralized internet where users do not need to trust remote servers and can run decentralized applications which can be seen in figure 5. To implement this Blockstack claims to have a solution to replace some of today's core internet infrastructure like DNS, public key infrastructure and storage backends. Blockstack is open source, can work on any blockchain and in addition, also offers identity management on its blockchain called blockstack identities [4].

6.1 Design

The blockstack identities consist of profile and globally unique names and can be registered for people, companies, websites, software packages and more [4]. Each profile can contain private as well as public information, both of which are put in by the identity owner. Other peers and select authorities can then validate this information.

Blockstack also implements a direct use for decentralized identities based on the blockchain with Gaia, its decentralized storage system [2]. The storage of data on Gaia has comparable speed to today's cloud services. That is the case because Gaia uses the infrastructure of current cloud providers (like Dropbox, Amazon S3 and Google Drive) to store encrypted and/or signed data on them. The stored data are tamper resistant because the data hashes are stored on the blockchain. On Gaia a user can use a decentralized identity based on the blockchain to log in to apps and services, which enables him to save data generated by those on storage backends under the user's control.

In order to replace DNS, Blockstack proposes a new system based on the Blockchain called Blockchain Name System (BNS). BNS provides similar functionality to DNS but does not have any central root servers. The naming system in BNS relies on names being owned by cryptographic addresses of the blockchain and their respective private keys. In order to be able to register a website, one has to first preorder it. This is done to prevent a race attack, in which the attacker tries to register the same name while the transaction is still unconfirmed. A website belongs to the user who was the first one to successfully write both, a preorder and register transaction. The names in the BNS are structured into namespaces whose information are stored on the root blockchain. Namespaces define the costs for names as well as their renewal rates and are therefore similar to the top-level domains (TLDs) in DNS. The resolvement process is as follows: The user that wants to resolve a name, makes a query to the BNS server that runs in her trust zone. Then the BNS server looks at the relevant blockchain, searches the relevant record and retrieves the respective zone file from the linked but untrusted external source.

6.2 Evaluation

The authors state a variety of different desirable properties in the paper [2]: The first thing is that "End-users should be able to (a) register and use human-readable names and (b) discover network resources mapped to human-readable names without trusting any remote parties". The first aspect gets enabled through the blockstack identity while the other aspect is being made possible by the BNS system.
Further they state that "End-users should be able to use decentralized storage systems where they can store their data without revealing it to any remote parties". This gets enabled through Gaia, where encrypted data gets stored in multiple clouds.
Lastly they want that "The end-to-end performance of the new architecture (including name/resource lookups, storage access, etc.) should be comparable to the traditional internet with centralized services". As already stated, this is true for Gaia, the decentralized storing system. There is no time comparison included with the BNS, though it needs less requests when resolving a name. However, it is still possible that it takes a longer time, even with less requests.

The first property (a), that names should be human-readable, is also the reason why Blockstack (and Namecoin) solve Zooko's triangle in contrast to Sovrin, Jolocom, uPort and ShoCard. This is due to the fact that in the latter ones the names are generally not human-readable. So, while all identity management systems in this paper are secure and decentralized (the two other properties of Zooko's triangle), only Blockstack and Namecoin also enable human-readable names. For example, in uPort the uPortID is the address of its corresponding proxy smart contract on Ethereum's blockchain which is a 160-bit number that is typically repre-

doi: 10.2313/NET-2018-11-1_14

sented as a 40-character hexadecimal string. In contrast, in Blockstack a name can be chosen by the user and the only requirement is its uniqueness.

The fact that Blockstack is able to use any blockchain is very important for further success. Because due to it, Blockstack is able to migrate from one blockchain to another, in case a blockchain becomes insecure in any way e.g. if not enough hash rate is in the system which enables malicious actors to do a 51% attack. However, it is not as secure as ShoCard's approach that supports multiple blockchains at the same time.
Another big advantage is that it already has a lot of registered identities. Currently Blockstack has 80,000 registered identities [4] and therefore is one of the biggest systems with identity management on the blockchain.

7. NAMECOIN

Namecoin is an open source technology that aims at improving decentralization (and therefore resistance of censorship), security and privacy [14]. Additionally, it also tries to improve certain structural aspects of the internet like the domain name system (DNS) and identities. The use cases listed on their website include censorship resistance, attaching attributes to an identity, decentralized TLS certificate validation based on the blockchain and the use of the .bit top-level domain (TLD) [14]. Namecoin is the first fork of the cryptocurrency Bitcoin, which was the first application of a blockchain, and consists for a great part of the Bitcoin codebase with some added functionality.

7.1 Design

Namecoin enables the user to register names and then store associated JSON values (up to 520 bytes) on their blockchain. The Namecoin software can then be used to search the blockchain for the name and retrieve the respective data (values). An identity can consist of a name, email address, photo (in the form of an URL due to the limited storing space), crypto address (like Bitcoin and Namecoin address), fingerprints of cryptographic keys and other things [15]. Unlike most other identity management systems, identities (or rather the names under which they are stored) have to be renewed at least every 35,999 blocks which is approximately every 200 - 250 days [13].

A Namecoin identity has a name and normally a connected email address and Namecoin address. Additionally, there is also a fingerprint of some cryptographic key stored. This data can then easily be fetched (if one knows the id) which can be seen in figure 6. Furthermore, Namecoin is the basis of NameID, a service that enables one to use his Namecoin identity to create an OpenID which can then be used to log into OpenID enabled websites.

7.2 Evaluation

Namecoin is already working and is able to fulfill most of its promises. Especially in the range of identity management, Namecoin does what it is supposed to do. Like Blockstack, Namecoin also solves Zooko's triangle, because a user can choose his name himself. Although, its attribute storage space for identities is only 520 bytes at maximum, it enables the use of URLs that can then contain almost infinite storage

```
$ namecoind name_show "id/khal"
{
  "email":    "khal@dot-bit.org",
  "bitcoin":  "1J3EKMfboca3SESWGrQKESsG1MA9yK6vN4",
  "namecoin": "N2pGWAh65TWpWmEFrFssRQkQubbczJSKi9"
}
```

Figure 6: Fetching data from the Namecoin identity khal. Example retrieved from [15]

space. But the information stored on the site of the URL still need to be present (e.g. as a hash) in the Namecoin identity, to disable tampering of them.
However, Namecoin is not widely used, hence, the .bit TLD is not really common. Even though Namecoin wants to decentralize the DNS system, their next goal is to get most nameservers to implement the .bit domain because otherwise there is no real growth for websites in the .bit domain. This is due to the fact that one currently needs to install Namecoin and then download the whole blockchain to visit those sites. Another possible solution that is currently explored and does not depend on the nameservers to implement the .bit TLD is the inclusion in a browser or OS. Furthermore, the registration of the .bit TLD as a special use domain, like TOR's .onion TLD, is tried.

8. COMPARISON

In this section we will - if possible - compare the different systems with each other by looking at their properties. Those properties are:
1. Usage of a permissioned blockchain?
2. Do they offer self-sovereign identities?
3. Built-in incentives for the nodes that run the blockchain to stay honest?
4. Who can use the identities and who can verify them?
5. Do they offer more than identity management?
6. Their current development status

The first property (1) is the use of a permissioned blockchain i.e. a blockchain for which nodes need permission to work on. A system that has a permissioned blockchain is Sovrin, where the nodes need to be certified to work on that blockchain. Jolocom and uPort use the Ethereum blockchain, which is not permissioned. ShoCard and Blockstack are not bound to one specific blockchain and therefore do not need to run on permissioned blockchains. Namecoin is running on its own fork of the Bitcoin blockchain and because the Bitcoin blockchain is not permissioned neither is the one of Namecoin.

The next property (2) is the one of self-sovereign identities. Sovrin, Jolocom and uPort claim to offer self-sovereign identity, and Sovrin as well as Jolocom fulfill all criteria for it. But uPort has a problem with the user's attributes, because anyone with the uPortID can see them. Because one has no choice over whom to reveal what attributes to, it does not satisfy all our requirements for self-sovereign identity. In contrast, ShoCard, Blockstack and Namecoin do not claim to offer self-sovereign identity. ShoCard does not enable its users to have a self-sovereign identity be-

cause it has a central authority (the ShoStore servers) used to store data which contradicts the conditions for a self-sovereign identity. Blockstack seems to meet all criteria needed for self-sovereign identity. However, since identity management is not the focus of Blockstack its implementation is not described. Therefore, it could be the case that it does not satisfy some criteria when it comes to privacy. Namecoin's problem for self-sovereign identity is the same as uPort's. The identity is stored as unencrypted JSON values on Namecoin's blockchain and thus anyone can access the data. Therefore, Namecoin does also not fulfill this criteria for self-sovereign identity.

Aspect (3) is about built-in incentives for the involved actors (mostly nodes running the blockchain) to stay honest which is important for a flawlessly working long term system. Sovrin does not offer incentives to help sustain the network, however, due to the nodes running the blockchain being trusted the risk of nodes being malicious shrink significantly. Both Jolocom and uPort use the Ethereum blockchain which offers mining rewards in the form of ether to its nodes [9]. ShoCard and Blockstack can work on different blockchains and therefore have the possibility to switch to one with rewards for miners. In Namecoin the miners also get rewards in its cryptocurrency NMC [13].

The following aspect (4) considers who can use the identities and who can verify them. In Sovrin anyone can create identities and anyone with access to the internet can verify them. Jolocom enables anyone to create an identity and any claim by a Jolocom identity can be verified using the Jolocom's user interface. With uPort everyone can create an identity and everyone can verify the claims that identity made by verifying the signature of that claim. ShoCard also makes it possible for everyone to create an identity, but the verification is done through some verifiers depending on the use case. In its most prominent example the verifiers are airline agents or automated machines of the airline. In Blockstack everyone can have an identity, but it may happen that the respective name is already taken and one has to take a different one. Blockstack claims that other peers and select authorities can then verify this information, but do not describe a specific process. Namecoin, like Blockstack, also enables anyone to have an identity but the names of the Namecoin identities are unique. Namecoin does not directly offer a method to verify a name identity. However, if the other person included the fingerprints of some cryptographic keys one could check if the other person is in control of those keys.

In order to classify the identity management system, aspect (5) is about whether the system offers more than just identity management. The only systems that provide more than identity management are Blockstack and Namecoin with their approach to a decentralized internet / DNS system. The other systems provide only identity management.

The last aspect (6) is the current development status of the systems i.e. what they currently offer. This is important to assess the likelihood of a system having widespread support. It is notable that most systems are developed as open source projects. For Sovrin we found no possibility to use it yet and create an identity at the current time, so it still seems to be in early development. Jolocom released the alpha version of its SmartWallet in the Google Play Store at the end of February [11]. With the app users are able to create a decentralized identity with verified credentials, just like Jolocom proposes. uPort already launched its uPort ID mobile app alpha in January 2017 and is working with the city Zug (Switzerland) to officially register their citizens [23]. The current working area is the mobile SDK to enable other apps to use uPortIDs and get a widespread support for them. ShoCard currently offers a demo version on its website, that one has to request. Furthermore, ShoCard Inc. offers two apps in the Google Play Store, ShoCard and ShoBadge. ShoBadge is a digital identity card for mobile application. Blockstack is already working and has (by its own claim) 80.000 registered identities. Similarly, Namecoin is also working and has implementations like NameID [16] that enables users to turn their Namecoin identities into an OpenID.

9. CONCLUSION

We presented six identity management systems in this paper. They vary in terms of different design and sometimes a different target group. Sovrin has its permissioned blockchain, Jolocom its HD-Keys to offer sub-identities, uPort utilizes smart contracts and ShoCard does not rely on one Blockchain and is further focusing on the consumer business interaction. Blockstack and Namecoin differ from the rest because they want to decentralize the internet and from each other due to Blockstack's wider approach in decentralizing the internet with decentralized DNS, storage and applications while Namecoin only proposes a decentralized DNS.
There are quite a few advantages that the identity management systems presented in this paper have over the current ones. Most notably those are: more control over one's identity and in some systems self-sovereign identity, a decentralized identity due to blockchain and easier verification to multiple entities. However, the systems still rely on TTP's to verify identity attribute claims, otherwise those claims are mostly useless.
Concluding, one can say that all described identity management systems have their special and unique properties and it remains to be seen whether identity management on the blockchain will be the identity management of the future and if any of the systems presented in this paper will succeed.

10. REFERENCES

[1] A. Abraham. Whitepaper about the concept of self-sovereign identity including its potential. October 2017. https://www.egiz.gv.at/files/download/Self-Sovereign-Identity-Whitepaper.pdf.

[2] M. Ali, R. Shea, J. Nelson, and M. J. Freedman. Blockstack: A new internet for decentralized applications. October 2017. https://blockstack.org/whitepaper.pdf.

[3] C. Allen. The path to self-sovereign identity, April 2008. http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

[4] Blockstack - Blockchian Identity. https://blockstack.org/posts/blockchain-identity.

[5] Blockusign. https://blockusign.co/signup.html.

[6] Decentralized Identifiers (DIDs) v0.10. https://w3c-ccg.github.io/did-spec/.

doi: 10.2313/NET-2018-11-1_14

[7] P. Dunphy and F. A. P. Petitcolas. A first look at identity management schemes on the blockchain. https://arxiv.org/ftp/arxiv/papers/1801/1801.03294.pdf.
[8] Ethereum-Whitepaper. A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper.
[9] Ethereum Wiki, Mining Rewards . https://github.com/ethereum/wiki/wiki/Mining#mining-rewards.
[10] C. Fei, J. Lohkamp, E. Rusu, K. Szawan, K. Wagner, and N. Wittenberg. Jolocom: Decentralization by design. February 2018. https://jolocom.com/whitepaper/.
[11] Jolocom - 2018 Where we have been so far. https://jolocom.com/2018-where-weve-been-so-far/.
[12] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf.
[13] Namecoin FAQ. https://namecoin.org/docs/faq/.
[14] Namecoin Website. https://namecoin.org.
[15] Namecoin Wiki. https://wiki.namecoin.org/index.php?title=Welcome.
[16] NameID Website . https://nameid.org/.
[17] K. Robles. Hierarchical deterministic keys. https://www.w3.org/2016/04/blockchain-workshop/interest/robles.html.
[18] ShoCard. Travel identity of the future. 2016. https://shocard.com/wp-content/uploads/2016/11/travel-identity-of-the-future.pdf.
[19] ShoCard. Shocard whitepaper - identity management verified using the blockchain. 2017. https://shocard.com/wp-content/uploads/2018/01/ShoCard-Whitepaper-Dec13-2.pdf.
[20] ShoCard Website. https://shocard.com/.
[21] Sovrin-Foundation. A protocol and token for self-sovereign identity and decentralized trust. January 2018. https://sovrin.org/wp-content/uploads/Sovrin-Protocol-and-Token-White-Paper.pdf.
[22] Sovrin Website. https://sovrin.org/.
[23] uPort - 2017 Recap, 2018 Outlook . https://medium.com/uport/uport-year-in-review-whats-to-come-in-2018-15ccb9214439.
[24] uPort Developer Website. https://developer.uport.me/.
[25] uPort Specs. https://github.com/uport-project/specs/.
[26] uPort Website. https://www.uport.me/.
[27] Z. Wilcox-O'Hearn. Names: Decentralized, secure, human-meaningful: Choose two. October 2001.

doi: 10.2313/NET-2018-11-1_14

Data Management in Distributed Systems

Simon Schäffner
Advisor: Stefan Liebald
Seminar Innovative Internet Technologies and Mobile Communications SS2018
Chair of Network Architectures and Services
Department of Informatics, Technical University of Munich
Email: simon.schaeffner@tum.de

ABSTRACT
Distributed Systems allow a number of nodes to collaborate on a problem. They can be of help when a single server can no longer fulfill the requirements or when a larger number of low performance nodes would like to cooperate. This paper gives a broad overview over different strategies for data management in different kinds of distributed systems. It focuses on the issues of scalability, performance, consistency, redundancy, overhead and attack resistance. Five systems from three categories are reviewed: peer-to-peer file sharing (BitTorrent and Kademlia), content delivery networks (Akamai) and distributed databases (Zatara and CouchDB).

Keywords
Distributed Systems, Data Management, Peer-To-Peer, Content Delivery Network, Scalability, Performance, Consistency, Redundancy, Overhead

1. INTRODUCTION
Since the beginning of computing, there have been two important developments: the advancement from single-core to multi-core processing and the establishment of local and global high-speed networks. This allows for a variety of different kinds of computers, ranging from small credit-card sized computers or even smartphones to supercomputers, to connect together and form a distributed system [19]. This system can be characterized as "a collection of autonomous computing elements that appears to its users as a single coherent system" [19]. "autonomous" refers to the "computing elements", called nodes, all being independent of each other and not being controlled by a central instance. "appear[ing] as a single coherent system" means that the system has a larger goal and therefor nodes have to collaborate to achieve the goal. In addition to these two characteristics, it allows for the nodes of a distributed system to be geographically dispersed.

Most tasks either require some data input, an ability to output/save data or even both. This is also the case in distributed systems, but data management is usually a lot more complex than just reading and/or writing data to a single disk.

This paper aims to compare data management strategies in different distributed systems of different categories. Mostly, popular systems with interesting and/or unique features were chosen. The goal is not to give an in-depth review of the strategies, but rather to give a broad overview over strategies that are actively used at the time of writing.

In the following, first, different attributes of data management systems are defined, and then the systems are compared based on these metrics. In section 2 we define different attributes that data management in distributed systems can have. In section 3 we take a look at 5 concrete distributed systems and compare their data management by the attributes defined in section 2. In the summary we compare all the systems and give an overview of them in table 1.

2. ATTRIBUTES
We now describe the six attributes the distributed systems are then compared by. The attributes were chosen as they are relevant for most systems building on top of them.

2.1 Scalability
Because of the openness of the internet, a service provider never knows when a spike in the popularity of their service might happen. Therefor they would like to be able to scale their applications indefinitely, as long as it makes sense economically. With the whole system, the data management system has to be able to scale as well.

2.2 Performance
Performance in this context describes how fast a requested datum can be accessed. This includes both the initial delay to find where the datum is stored and how fast it can then be transferred to the requesting node.

2.3 Consistency
For redundancy a datum may be stored on more than one node. This implies a challenge whenever a datum is changed as the change has to be distributed through the system. Consistency describes how well this is handled and if there is the chance of receiving stale data.

2.4 Redundancy
As described above, a system can store data in more than one place to ensure its availability or to increase performance. Higher redundancy is better.

2.5 Overhead
Because of the nature of distributed systems, nodes inside them have to communicate one way or another. One extreme is that all nodes have all data saved in their local storage,

doi: 10.2313/NET-2018-11-1_15

but then any change in data has to be broadcast. The other extreme is that the data is completely distributed between the nodes, but then the nodes have to request data from each other. Independent of the strategy for redundancy, this communication obviously needs additional resources called overhead over a single computer directly accessing data on its internal disk. Less overhead is better.

2.6 Attack Resistance

The open nature of the internet implies that distributed systems may be attacked. In the case of data management, an attack could mean not being able to deliver data to the rest of the system anymore or delivering manipulated data. Attack resistance describes how many attack vectors there are and how significant each is.

3. COMPARISON

In the following we are going to compare different distributed systems, that have a focus on data management. We chose two protocols relevant for peer-to-peer filesharing, BitTorrent and Kademlia, the content delivery network Akamai and two distributed databases, Zatara and CouchDB.

3.1 Peer To Peer Filesharing

While peer to peer filesharing first emerged with the development of Napster[1] and gnutella[2] for legally controversial uses, it has advantages over traditional server-client structure downloading, that makes it interesting for completely legal use, as well [18]. It uses shared resources to split the load between many participants and has a higher reliability as there are fewer single points of failure.

3.1.1 BitTorrent

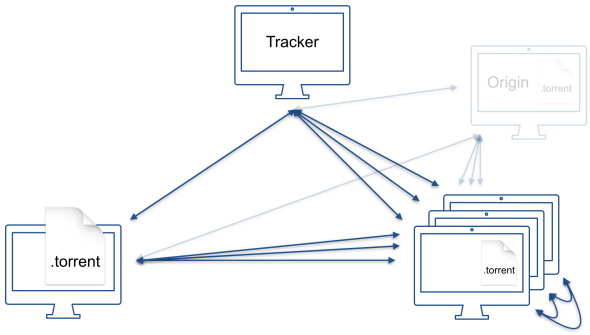


Figure 1: BitTorrent network with offline origin

BitTorrent is a peer-to-peer file sharing protocol specified in [10]. It is still under active development, and is still the most popular peer-to-peer file sharing protocol on the internet, but its usage falls with streaming services, such as Netflix[3], gaining more popularity. [7]

A download is started by first downloading a .torrent file containing all the meta information needed. This includes the URI of a so-called tracker, which coordinates all the peers downloading the same file. A peer, also called a leecher when it has not yet completed the download, connecting to a tracker receives a list of other peers, which have parts of the file available that the leecher still needs. As long as the network as a whole has the complete file available, every peer can download the needed parts from the other peers. Initially one peer, the 'origin', has to have the whole file available at once.

The intention of a BitTorrent network is to distribute one or more files to anyone or any node that can access the .torrent file and connect to the tracker. So the goal of this distributed system is data management itself. On a higher level, metadata also has to be managed. Especially the .torrent file has to be managed externally, so it is not considered here.

In the following BitTorrent is analyzed according to the attributes described in section 2.

Scalability: According to a two-week experiment described in [16], the download speed scales with the number of peers. The trackers are no bottleneck as they only have to distribute metadata. The more problematic issue is fairness. An investigation[21] shows that peers with a high download speed from the origin become "Super Peers" that have a very high share ratio (upload to download ratio). In all five repetitions of the experiment the group of "Super Peers" consisted of the same peers independent of the time they joined the download. This is unfair because a small number of peers has to provide a larger amount of resources than the other peers. Independent of fairness, scalability is still good.

Performance: BitTorrents upload utilization is very good as reported in [6]. In their experiments in a homogeneous environment (same bandwidth for all leechers) the upload utilization was around 95%, independent of the number of peers. The download utilization was lower because it was bounded by the upload speed of the leechers, so the bottleneck was the upload bandwidth, not the protocol being incapable. Performance of BitTorrent is very good because of the upload utilization.

Consistency: As the content of a torrent never changes and the user first downloads the metadata file containing checksums of all pieces of the file(s), consistency is not an issue with BitTorrent. In the worst case scenario of downloading a corrupted piece, the piece is just discarded and downloaded again. In the case of downloading a corrupted metadata file, the BitTorrent client will notice a difference in the checksum from the metadata file and the checksum provided by the tracker and will stop the download [10].

Redundancy: As with all file-sharing platforms, a higher redundancy is better, because it infers a higher availability. There is also no disadvantage of high disk space utilization as every peer who downloads the file also wants to use it, so they want to have it available locally to them. With BitTorrent, a higher redundancy also implies a higher download speed for new leechers.

Overhead: The obvious overhead of BitTorrent, compared to downloading a file from a single source, is the metadata

[1] `https://us.napster.com`
[2] `http://www.gnutellaforums.com`
[3] `http://netflix.com`

doi: 10.2313/NET-2018-11-1_15

file, the connection to the tracker and the meta information sent when connecting to other peers. The connection to the tracker makes up for a thousandth of all traffic, which makes it negligible [9]. The already mentioned overhead of replicating the file to every node in the system and thereby using a lot of disk space is not negative as this is the goal of the system.

Attack Resistance: While there are few attack vectors on BitTorrent that actually result in the successful download of a corrupt file, there are methods to hinder a download. These are called poisoning attacks and are executed by anti-piracy-agencies and malicious users. Poisoning attacks include uploading a large amount of fake files and/or malware, so that users cannot find the file they are looking for with a search engine [11]. Another possible attack is flooding all peers that offer parts of the file with download requests so that it cannot be downloaded by others. This and other methods are even offered as a service by companies like MediaDefender [5].

All in all, BitTorrent is a peer-to-peer file sharing protocol with high performance, a good strategy for consistency, little overhead and a small number of attack vectors.

3.1.2 Kademlia

Kademlia [14] is a peer-to-peer distributed hash table (DHT). Even though Kademlia was developed as a research project, it has found its way into practical usage with it being integrated into BitTorrent [13] and the cryptocurrency Ethereum [20].

The identifier space is used for both keys and node IDs. Keys are stored on nodes whose IDs are "close". Because the whole network is interpreted as a binary tree, the magnitude of the distance in a fully populated tree is determined by the height of the smallest subtree the two nodes are part of. The path from the root of the network to the root of the subtree sets the prefix for all identifiers in the subtree.

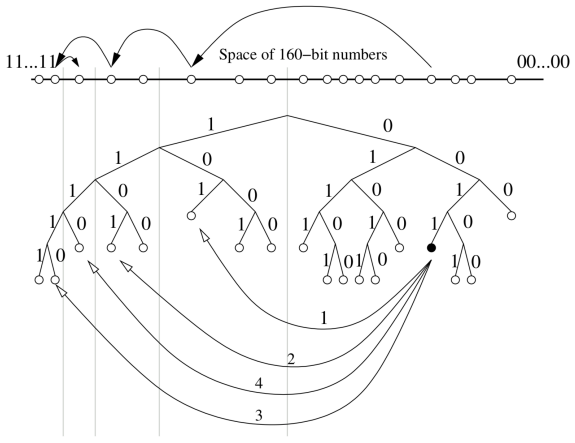


Figure 2: Finding a key in Kademlia [14]

To ensure that a node can find every other node in the network, it has to know at least one node from every subtree it is not part of. To look up the value for a given key, the node sends parallel requests to the k nodes it has contact with closest to the key. Each of the nodes returns the k nodes *it* has contact with closest to the key. The requesting node then sorts the nodes by distance to the key and contacts the k closest nodes. This is done recursively until one of the contacted nodes returns the value.

To store a <key,value> pair, a node searches for the k closest nodes to the key and sends them a store command. When a new node joins the network, it introduces itself to the system and stores all <key,value> pairs it is one of the k closest nodes to.

Kademlia can be used as a way to remove nearly all central components from a peer-to-peer network. In this context it provides a way to find data and information on how to access it, so in the case of BitTorrent it replaces the central trackers [13].

Scalability: The amount of storage scales linearly with the amount of nodes as the redundancy-factor k is global and not dependent on the amount of nodes in the network. Of course, with an increasing amount of nodes in the network, the amount of nodes that have to be contacted to find the value of a key increases, but because of some optimizations in the tree lookup system this scales even better than $O(log_2(n))$ [14] which is good.

Performance: Kademlia uses parallel, redundant requests to decrease performance-loss by broken nodes or nodes that have left the network. When another node does not reply within a certain time, the node is removed from the requesting node's contact list and is no longer contacted until it reintroduces itself. The system also uses caching to decrease the likelihood of creating a hotspot on a single node when a certain <key,value> pair is often requested. Whenever a node requests a value for a given key, it sends a store command to the node closest to the key that did not have the value stored before. Because of the unidirectional topology, requests from other nodes for the same key are highly likely to hit the caching nodes. With parallel requests and optimized caching Kademlia's performance is very good.

Consistency: The original publisher of a <key,value> pair has to republish it every 24 hours to limit stale information in the system. Each of the k nodes the <key,value> pair is stored on primarily has to republish it every hour. To decrease the amount of cached stale information, the time to live in the cache is exponentially inversely proportional to the distance between the node the <key,value> pair is primarily stored on and the node it is cached on. Since stale information can live a long time in Kademlia, its consistency is not outstanding.

Redundancy: <key,value> pairs are stored at least k times in the system as they are stored on the k closest nodes to the key, so whenever a node leaves the network (or dies) at least $k-1$ nodes should still have the information available. Due to republishing, after a maximum of one hour, the information should be available on k nodes again. When all k nodes that store the same <key,value> pair leave the network in a single hour, the information should be available again after 24 hours at last. With k chosen to have a small likelihood

doi: 10.2313/NET-2018-11-1_15

of k nodes leaving the network in one hour, Kademlia's redundancy is good.

Overhead: As already mentioned, <key,value> pairs have to be republished every hour. This process is optimized by a node not republishing for the next hour when a republish was received, as it is assumed that the other nodes have received the republish as well. Caching also introduces some overhead but that is limited to short time periods and to <key,value> pairs that are requested often. It also decreases the amount of hops needed to receive a value for a given key, so it reduces network traffic which is the more precious resource in a peer-to-peer system that runs on end users' home computers.

Attack Resistance: One attack vector for open distributed systems is to overtake a large part of the system and by that being able to control the whole network. This is difficult with Kademlia as the existing nodes do not replace nodes in their contact list that have been longer known to them with newer ones. One reason for this is that statistically, nodes that have been in the network for a longer time have a smaller risk of leaving the network. Another reason is that a malicious attacker cannot overtake the network by flooding it with new nodes.

Overall Kademlia has a good scalability, a great performance, but falls short on consistency and overhead as keys have to be actively republished but cannot be actively removed.

3.2 Content Delivery Networks

Websites usually start out running on a single server, but when they grow large, even a single cluster of servers is no longer enough. One possibility is to use proxies in front of the content generating servers in order to cache static content. But now that a large amount of content on the internet is generated dynamically, hit rates of proxies are low (25-40%) [12] and more elaborate systems called Content Delivery Networks (CDNs), were developed. These systems consist of a large amount of servers distributed both geographically and regarding network topology. Whenever a user wants to access a website, the user actually connects to one of the CDN's servers instead of the actual website's servers. The CDN only connects to the content generating server if it has to.

3.2.1 Akamai

Akamai is one of the world's largest content delivery networks (CDN) with well known customers such as Facebook, Adobe and Airbnb. They have "more than 240,000 servers in over 130 countries and within more than 1,700 networks around the world"[4].

When a large amount of users hits a single website at once (a flashcrowd), Akamai allocate more of their servers to the website that needs them at the moment and less to others. They also try to serve users from a server nearby, so that latency is low and packet-loss is small [12].

For customers with large amounts of data, Akamai use tiered distribution within their own network: A set of "parent" clusters (see figure 3) is used to cache the data within the network, and when another cluster does not have that data available, it retrieves it from the parent cluster instead of the origin. This can result in offload of over 90% [15].

Akamai also handle livestreams through their network [15], but this is out of the scope of this paper.

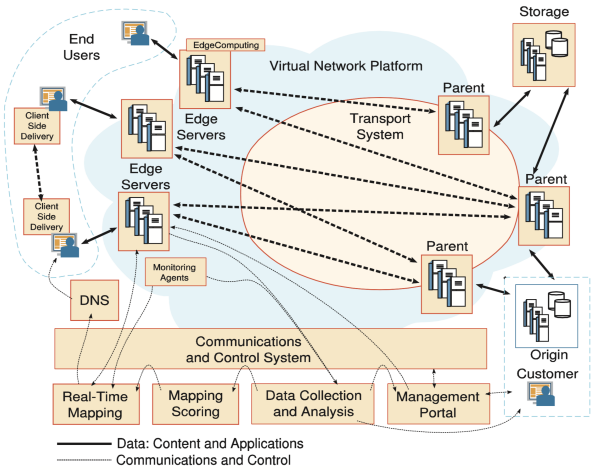


Figure 3: Overview of the Akamai network [15]

Scalability: Akamai was designed with scalability in mind [15]. Tiered distribution allows for a large amount of data being available within the network and therefor being available at high speed to the whole overlay network.

Performance: Performance was another goal Akamai was designed for [15]. They have developed their own overlay network with a number of improvements over using standard internet. One advantage is path optimization, as paths defined by the Border Gateway Protocol are not always optimal and sending traffic through an intermediate server on the Akamai network can be faster. This can show improvements in speed of 30-50% [17]. On top of that it can increase reliability by offering alternate paths. Another advantage of path optimization is reduced packet loss. For applications requiring low latency, a packet can be sent via multiple ways simultaneously, which decreases the chance of a packet not reaching its destination at all. This is also combined with forward error correction techniques to decrease the amount of times packets have to be discarded because of transmission errors. Additionally, Akamai uses transport protocol optimizations to mitigate the overhead of protocols like TCP. Further, application level optimization is used when possible. This includes content compression or even the implementation of application logic on edge servers [15].

Consistency: Data management for cacheable objects is mostly based on standard techniques such as assigning time-to-live values to each object or using different URLs for different versions of the same object. But as Akamai only serve content for their customers, they expect to retain control over their data [15].

Redundancy: How often a datum is stored in the Akamai network depends on the customer and their needs. As mentioned above, for a customer Akamai is an extension of their

doi: 10.2313/NET-2018-11-1_15

own network and the amount of redundancy is completely dependent on their application. With tiered distribution Akamai tries to strike a balance between redundancy and fast availability within their own network.

Overhead: As Akamai's overlay network is proprietary, the protocol overhead is not known, but they claim it to have improvements over standard TCP as mentioned above (see performance). This is done by reducing the amount of times a connection has to be setup and torn down, as they can leave up connections within their network for a longer time.

Attack Resistance: As Akamai is a closed network, there are no attackers within the network that could try to destroy data or deliver wrong data. Attacks from the outside are still possible, but Akamai was engineered with a high failure rate of equipment and connections in mind. Because of that, a lot of effort was put into recovery from all kinds of failure scenarios [15].

All in all, the highly distributed nature is fundamental to Akamai's high performance. The entire communication within the overlay network is optimized and the two small hops on either end are meant to be short enough that they do not matter. This means great scalability, performance and redundancy.

3.3 Distributed Databases

A distributed database can partition data over many servers, for OLTP use-cases (transactional processing: a large amount of simple queries) each query can be handled by a different node or for OLAP use-cases (analytical processing: a small amount of complicated queries) the nodes can work together on a single query. With replication, they can also make large amounts of data available to geographically dispersed systems with a low speed link in-between.

Zatara was chosen as it was one of the first general use-case NoSQL databases. CouchDB was selected for its ability to gracefully resynchronize a client that was offline for some time.

3.3.1 Zatara

Zatara is an eventually consistent distributed database built to satisfy the needs of modern cloud applications [8]. Opposite to relational databases like MySQL or Postgresql, Zatara is a NoSQL database. NoSQL databases allow for other data models than tables and support other query systems than SQL. Zatara, in particular, is more similar to a key-value-storage.

Each key can be of the type cache only or persistent storage. Cache only keys are stored on a single node in memory and are not replicated at all. This implies that they are lost on node failure and are non persistent. Persistent keys are eventually consistent and are stored on disk. They are also replicated to all other nodes in the group.

Each key to be stored is mapped to a single node by a hashing algorithm and the client connects directly to that node to store the key. For cache only keys this is the one node that stores the key and for persistent keys this is the primary node the key is stored on and the node that handles replication for this key.

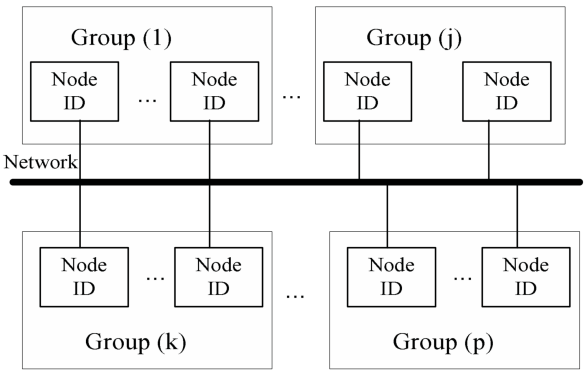


Figure 4: Zatara nodes organized in groups [8]

When a new node joins the network, there are two options: The first option is that the node joins an existing group and replicates the keys that are stored in the group. The other option is that a group is split. For this, the keys of the old group have to be rehashed and are thereby divided between the two new groups.

Every node is configured with a unique node id and authentication information. To connect to the network, each node is given a list of other nodes in the network.

Scalability: Data is replicated regionally only, whereby replication data only has to be sent to a small amount of nodes, instead of the whole network. Data is also replicated asynchronously after it has been stored on at least a second node, which brings a good scalability.

Performance: One of the performance improving design aspects of Zatara is that keys are cached in memory and the more often they are requested, the higher a TTL they are being assigned. This decreases the chance of the key being deleted in case the node runs out of memory. In more practical terms, the authors not only describe Zatara in [8], but also evaluate its performance using 196 Amazon EC2 Large Instances as nodes [8]. They come to the conclusion that performance scales near linearly without any replication and is only 10% slower with replication for reading a key from the database. In [8] they note that they were "able to reach more than 20 million operations per second in a pay-as-you-grow cloud infrastructure that costs less than 100USD per hour."

Consistency: Zatara offers eventual consistency[4], so it guarantees that the results for a changed entry will be consistent after the consistency window has closed. This mechanism is implemented by the node that primarily stores the key actively distributing the new value for the key to the other nodes in the group. As soon as at least one other node of the group has acknowledged the update request, the node acknowledges the request to the client. By this, Zatara

[4]consistency is guaranteed only a given timespan after the data update is completed, not directly, as it is with traditional database systems

doi: 10.2313/NET-2018-11-1_15

can also guarantee a persistent key to be always stored on at least two nodes. If a node cannot be contacted, the sending node will increase a soft fail counter for that node. If the soft fail counter increases above a certain threshold, the node's status will be set to HardFail and other nodes will be informed.

Redundancy: Persistent keys in Zatara are guaranteed to be stored on at least two nodes. When the primary node for the key is not available, the key will be read from another node in the group. As long as the groups are large enough, redundancy in Zatara is good.

Overhead: Asynchronous replication is used to reduce overhead compared to synchronous replication, but it is still a problem in large networks with many nodes. To solve this problem, nodes are organized in groups with recommended group sizes of 2-4 nodes per group. With the recommended group size, Zatara can strike a balance between redundancy and overhead.

Attack Resistance: Zatara employs an internal key to store username and passwords and another internal key to store which users are allowed access to which databases. Every client and every node first has to authenticate against the internal key, so in theory only trusted clients and nodes should have access to the network.

All in all, Zatara is a distributed database with very good scalability because of asynchronous data replication, good performance because of in-memory caching and eventual consistency. Overhead and attack resistance are neither outstandingly good nor bad.

3.3.2 CouchDB

CouchDB [3] is a document storage NoSQL database aimed at web applications. Documents are the main unit of data, consisting of a number of fields and attachments [1].

One of the primary use cases of CouchDB is to have multiple offline nodes and then synchronize the data upon reconnect. CouchDB is designed to handle partitioning of the system, especially when a single node disconnects and reconnects to the network, gracefully [1].

Internally, documents are versioned and when a new version of a document is saved, it is appended to the end of the database file. Occasionally, all current versions are copied into a new file and the old file is deleted when no clients use it anymore [1].

Scalability: CouchDB restricts data lookup to keys, which allows for data to be partitioned between many nodes without loosing the ability to query each node individually [1]. This brings good scalability.

Performance: Locally (on a node) data is stored in B-trees, so that data lookup by key efficient ($O(logN)$). Additionally, CouchDB uses multiversion concurrency control (MVCC) instead of locks [1]. With MVCC, queries see the state of the database at the beginning of the query for their whole lifetime. Because of this, queries can be run fully in parallel as long as they do not write to the same data set. As already mentioned, data can be partitioned between nodes to handle large amounts of data and large queries. Because of efficient data lookup and parallel requests, performance of CouchDB is good.

Consistency: A single CouchDB node is fully ACID (availability, consistency, integrity, durability) compliant. As mentioned above, MVCC is used for reading queries. When a client tries to save a document that in the meantime has been edited by another client, an edit conflict is triggered. The client then is offered the option to resolve the conflict by reapplying the changes to the newer version of the document. A conflict is also triggered, when an offline node reconnects to the network and a document was changed on both the network and the node. Each node deterministically decides which version of a document wins in this situation (usually the newest one). The loosing versions are still stored in the database and can still be accessed. They are only purged on database compaction. Loosing versions, like any others, are replicated among the network. Because of that, every node in the network sees the conflict and has the option to resolve it either automatically or manually [1]. This is an elaborate system, similar to version control systems like git[5].

Redundancy: This is left for the user of the database system to decide upon. Databases can only run on a single node, can be fully replicated between many nodes, can be partitioned between many nodes or any configuration in between. All of these options have their own advantages and disadvantages resulting in higher or lower query speed and higher or lower availability [1].

Overhead: As mentioned above, data can be fully replicated between a number of nodes in the system. This obviously brings a large overhead in both storage and communication. When a node, that was offline for a while, reconnects to the network, all changes have to be transferred to it and all the changes on the reconnecting node have to be transferred to other nodes in the network. The amount of overhead depends on the level of redundancy, so it is not a negative.

Attack Resistance: For external security, CouchDB implements a simple authentication model by default, but also allows for a custom, more complex model. One can implement a JavaScript function for update validation that receives both the new state of the document and authentication of the client to decide on that basis if the update should be allowed [1].

All in all, CouchDB is a very flexible system being both useful with data partitioned over many servers and a single instance running on a smartphone. It offers very good performance and an elaborate system for consistency.

4. SUMMARY

In table 1 an overview of all systems and their particular advantages and disadvantages is given. All of the systems are marked at least "good" in performance, as every system is optimized for at least one use-case and performs well in it.

[5] `https://git-scm.com`

doi: 10.2313/NET-2018-11-1_15

Table 1: Overview of All Systems

System	Data update	Scalability	Performance	Consistency	Redundancy	Overhead	Attack Resistance
BitTorrent	-	++ dl. speed scales lin. with #nodes	+ upload util. very good	0 no data updates; corrupt data identified	+ high	0 high, but aligns with goals	- poisoning attacks possible
Kademlia	active republication	+ storage scales lin. with #nodes; lookup time scales with $O(log_2 n)$	++ parallel redundant requests, efficient caching	+/0 old information max. 24h in system	+ $< key, value >$ pairs stored on $>= k$ nodes	- optimized process republishing $< key, value >$ pairs every hour	+ cannot be overtaken by node flooding
Akamai	dep. on customer	++ >240.000 nodes	++ optimized transfer speed in network	0 based on TTL/version URLs, but dep. on customer	++ tiered distr. allows for any degree	0 improvements over standard TCP, dep. on redundancy	++ only attacks from outside possible; high recovery rate
Zatara	active	++ asynchr. data repl.; large #groups possible; $O(1)$ key lookup	++ in-memory caching (least recently used)	+ eventual consistency	+ keys are guaranteed to be stored on >= 2 nodes	0 regional replication	0 simple authentication
CouchDB	active	++ data lookup by key only; data can be partitioned over many nodes	++ key lookup locally $O(logN)$; MVCC for parallel reading queries	++ eventual consistency; graceful conflict handling	++ dep. on use-case	+ dep. on redundancy needed	+ simple auth. model by default, can be customized

There are also interesting differences between the systems. BitTorrent does not allow for the actual data to change and is the only system that uses passive data update (for metadata) by having the nodes poll from the tracker. All other systems use active updates for replication. With Kademlia data is republished every hour, Zatara actively updates the values during the replication window and CouchDB replicates data actively on change or on reconnection to the network. Akamai with tiered distribution can invalidate cached data actively, dependent on the customer's needs.

It makes sense that active replication was found the most as it decreases the time of the network being inconsistent. It brings more overhead if only a single node needs the data in the end, but that does not align with the goal of high availability that most systems have.

There are also different strategies for deciding on which nodes to store data. BitTorrent stores all data on all nodes. Kademlia stores a key on the k closest nodes to the key. With Zatara a key is stored on a primary node that then replicates it to the rest of the group. CouchDB is very flexible on this issue as one can use a custom filter function for every node to apply during replication. With Akamai using tiered distribution, data is stored on a few primary nodes first and is then cached by other nodes when it is needed. Which nodes are chosen as the primary ones is dependent on the customer's needs, e.g. where the customer's main userbase is from geographically.

There are also different ways for conflict handling. CouchDB is the only system that has an elaborate strategy. As mentioned above, it sets the key to a conflict state and then lets the clients handle it. All the other systems do not do conflict management. BitTorrent does not need it as data cannot be changed. Tiered distribution also does not create any conflicts as data comes from a single place and is propagated from the primary nodes. A node in a Kademlia network looses all data on disconnect and therefor cannot create a conflict on reconnect. With Zatara all writes are directed to the same node and are then replicated from there, so there is a single authority for every key in the system. Larger partitions breaking off the network and then rejoining it is not provided for.

5. CONCLUSION

All of the systems, with the exception of Zatara, are popular at the time of writing: BitTorrent is the most popular peer-to-peer filesharing protocol [7], Kademlia is implemented in BitTorrent [13], Akamai reaches more than 30 Terabits per second of traffic and CouchDB is used by many small companies and even large ones like Akamai [2].

A short overview of the different protocols used in the systems was given and their strategies for data management were compared. Each of them has their own advantages and disadvantages.

As mentioned, distributed systems could only be developed because of high-speed networks. The worlds largest high-speed network - the internet - is available and here to stay, so the amount of distributed systems will only ever increase. With more and more aspects of peoples lives moving online, systems will have to be employed that support a very large workload and can be accessed from all over the world. The only type of systems available to handle this, are distributed systems.

6. REFERENCES

doi: 10.2313/NET-2018-11-1_15

[1] 1. introduction. http://docs.couchdb.org/en/2.1.1/intro/index.html. last accessed: 02.06.2018 15:09 (revision f85b3421).

[2] Companies using couchdb. https://idatalabs.com/tech/products/couchdb. last accessed: 02.06.2018 15:11.

[3] Couchdb - relax. http://couchdb.apache.org. last accessed: 02.06.2018 15:10.

[4] Facts & figures. https://www.akamai.com/us/en/about/facts-figures.jsp. last accessed: 27.05.2018 20:47.

[5] N. Anderson. Peer-to-peer poisoners: A tour of mediadefender. https://arstechnica.com/tech-policy/2007/03/mediadefender, May 2017. last accessed: 27.05.2018 15:38.

[6] A. Bharambe, C. Herley, and V. N. Padmanabhan. Analyzing and improving bittorrent performance. 01 2006.

[7] K. Bode. Netflix now accounts for 36.5% of peak internet traffic. http://www.dslreports.com/shownews/Netflix-Now-Accounts-for-365-of-Peak-Internet-Traffic-133945, May 2015. last accessed: 02.06.2018 15:12.

[8] B. Carstoiu and D. Carstoiu. High performance eventually consistent distributed database zatara. In *INC2010: 6th International Conference on Networked Computing*, pages 1–6, May 2010.

[9] B. Cohen. Incentives build robustness in bittorrent. http://www.bittorrent.org/bittorrentecon.pdf, May 2003. last accessed: 27.05.2018 14:51.

[10] B. Cohen. The bittorrent protocol specification. http://www.bittorrent.org/beps/bep_0003.html, Feb 2017. last accessed: 18.05.2018 18:44.

[11] R. Cuevas, M. Kryczka, Á. Cuevas, S. Kaune, C. Guerrero, and R. Rejaie. Is content publishing in bittorrent altruistic or profit-driven? In *Proceedings of the 6th International Conference on emerging Networking EXperiments and Technologies (ACM CoNEXT 2010)*, http://hdl.handle.net/10016/10116, December 2010.

[12] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 6(5):50–58, Sep 2002.

[13] A. Loewenstern and A. Norberg. Dht protocol. http://www.bittorrent.org/beps/bep_0005.html, Jan 2008. last accessed: 02.06.2018 15:11.

[14] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In P. Druschel, F. Kaashoek, and A. Rowstron, editors, *Peer-to-Peer Systems*, pages 53–65, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[15] E. Nygren, R. K. Sitaraman, and J. Sun. The akamai network: A platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, Aug. 2010.

[16] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In M. Castro and R. van Renesse, editors, *Peer-to-Peer Systems IV*, pages 205–216, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[17] H. Rahul, M. Kasbekar, R. Sitaraman, and A. Berger. Towards realizing the performance and availability benefits of a global overlay network. *MIT CSAIL TR 2005-070*, Dec. 2005.

[18] R. Steinmetz and K. Wehrle. *2. What Is This "Peer-to-Peer" About?*, pages 9–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[19] M. van Steen and A. S. Tanenbaum. A brief introduction to distributed systems. *Computing*, 98(10):967–1009, Oct 2016.

[20] vbuterin and J. Ray. Kademlia peer selection. https://github.com/ethereum/wiki/wiki/Kademlia-Peer-Selection, Oct 2015. last accessed: 02.06.2018 15:13 (revision ea47c31).

[21] Z. Zhang, Y. Li, Y. Chen, P. Cao, B. Deng, and X. Li. Understand the unfairness of bittorrent. 12 2010.

Observing TCP Round Trip Times with FlowScope

Christian Wahl
Advisor: Simon Bauer, M. Sc.
Seminar Innovative Internet Technologies and Mobile Communications SS2018
Chair of Network Architectures and Services
Department of Informatics, Technical University of Munich
Email: wahlc@in.tum.de

ABSTRACT
Most of our network software implementations depend on the Transmission Control Protocol (TCP). Thus, the performance and reliability of these solutions depend on a reliable protocol. A fast connection requires a timely acknowledgement of a received data packet. We want to measure this timespan, between the sending of the packet by the source and the arrival of the acknowledge by the destination. This timespan is called round trip time (RTT). Therefore, we present a solution to observe this timespan by employing FlowScope to measure the round trip time with the help of a suited extension module. Furthermore, we evaluate with this proof-of-concept whether FlowScope [4] is a appropriate tool for this purpose and it can collect suitable data for further network focused research.

Keywords
TCP, FlowScope, RTT, Round Trip Time, Network Monitoring

1. INTRODUCTION
The Transmission Control Protocol (TCP) is used to transfer data over networks which could be deficient and as a consequence lose data packets. TCP [12] ensures a successful transmission by awaiting the confirmation before sending the next packet. TCP is the most used network transmission protocol, accounting for around 90 percent of the internet traffic [5, 8].Thus, weaknesses of the transport route will affect a majority of internet users. One of this weaknesses could be a link between the source and destination with a raised latency which leads to increased response times of the connection or in extreme cases, i.e. in case of a failure to timeouts of the session due to packet loss. To monitor the time between sending a data packet and the acknowledgement of the same in (near) real time, we present a solution based on FlowScope [4], extended by a module created for this purpose.

We structure the paper as follows: First, we introduce relevant terms and definitions in Section 2. Second, we discuss related work in Section 3. After that, we show the implementation in Sections 2 and 4. Then, we look at the results of the implementation in Section 5 and look at another way to mitigate the shortcomings found during the evaluation in Section 6. Furthermore, we propose possible improvements to the solution in Section 7. Before concluding the paper in Section 9, we encourage our readers to reproduce our research on different input data in Section 8.

2. TECHNICAL BACKGROUND
In this section we give an overview over employed tools, define commonly used terms and describe the investigated protocol.

2.1 Network Flow
A network *flow* is defined by its unique five-tuple. This tuple is formed by the *source ip address* and *port*, *destination ip address* and *port*, and by the used protocol (in our case TCP):

$$\text{flow tuple} = (\text{src}_{\text{IP}}, \text{src}_{\text{port}}, \text{dst}_{\text{IP}}, \text{dst}_{\text{port}}, \text{protocol})$$

2.2 Round Trip Time
This term defines the timespan between the point when the sender transmitted his packet and when it receives the acknowledgement that the packet has been received. This timespan is called round trip time (RTT). The authors of [7] discuss the different causes for the delay. The delay might be caused due to the transport medium (i.e., based on the physical properties of light or electrons), network (caused by queuing and congestion) or processing delay, etc.

Formally, we define it as the difference between the point in time where the packet with sequence number x arrives and the point in time when the packet arrives which acknowledges packet x (refer to Section 2.3 and to [12] for an explanation)

$$t_{\text{seq}} = \text{time stamp of packet with sequence number } x$$
$$t_{\text{ack}} = \text{time stamp of packet with acknowledge number } x + 1$$

$$RTT = t_{\text{seq}} - t_{\text{ack}}$$

2.3 Protocol Background
TCP is embedded into the protocol stack at Layer 4 of the OSI Model on host computers [12]. It assumes to be able to get data-segments from the layer below and to forward received data to the layer above. It is packet oriented and sends and receives the packets from the surrounding layers. TCP is a versatile protocol and has a number of optional header extensions. The protocol is defined in RFC 793 [12]. However, as we only want to analyse the round trip time we

doi: 10.2313/NET-2018-11-1_16

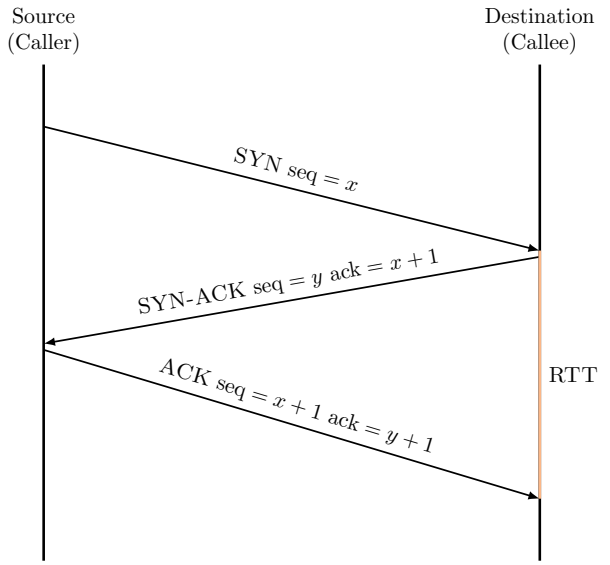


Figure 1: The TCP Three-Way Handshake [12] with Round Trip Time

focus on parts of the protocol that are important for this analysis.

A TCP packet includes the SYN-bit that is set (= 1) when the caller wants to initiate a new connection, the ACK-bit is set if the sender acknowledges a packet. TCP uses a procedure called "Forward Acknowledge", this means the acknowledge number is advanced by one compared to the acknowledged sequence: When a packet with acknowledge number x is received, then the packet with sequence number $x - 1$ is acknowledged. Furthermore, a TCP connection is initiated with the Three-Way Handshake depicted in Figure 1 and ended by the two parties sending a packet with the FIN-bit set and both acknowledging it.

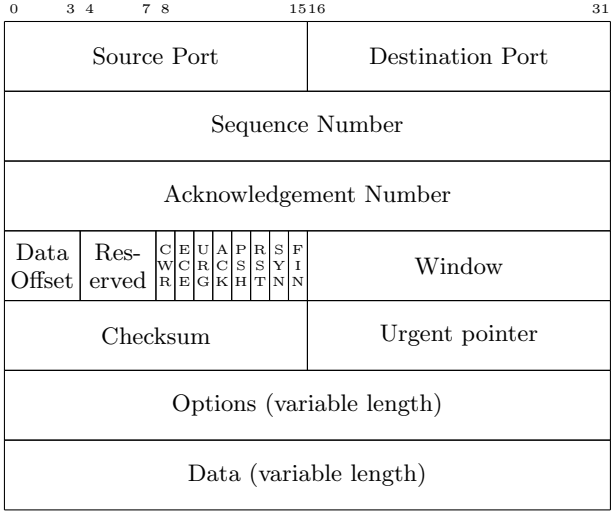


Figure 2: TCP header [12, 15]

2.4 Employed Software

During the development and test of the FlowScope module we used MoonGen [3] to replay a stored packet trace containing a SSL-scan and FlowScope [4] to capture and analyse the replayed traffic. Both tools are based on the *libmoon* [2] framework which encapsulates the *Data Plane Development Kit (DPDK)* [17]. The *libmoon* library extends the *luajit* [11] *just-in-time* compiler with interfaces to *DPDK* and a multi-threaded runtime model.

2.4.1 MoonGen

MoonGen is a customizable high-speed packet-generator [3]. It was used to generate test traffic that includes real TCP session interactions (including sequence numbers with the appropriate acknowledge numbers). This was accomplished by utilizing the included script to replay a captured *pcap* (refer to Section 8 for more information).

2.4.2 FlowScope

To capture and analyse the traffic generated by MoonGen, FlowScope was employed. It is designed for networks with a link rate exceeding 10 Gbit/s and according to the authors has been successfully tested with rates above 100 Gbit/s [4]. Most of the high-level functionality is written in *lua*. FlowScope is multi-threaded and due to the limitation of the *luajit* [11] each thread runs a separate instance of the *luajit* environment. With the means provided by *libmoon* it is possible to communicate over thread-boundaries. Each thread of FlowScope has a defined purpose [4, 14]:

Inserter This thread reads the packets from the NIC and stores it in a ring buffer.

Analyzer This thread runs the user module on a packet and stores the packet in a hash map according to the result of the user module. The required functions are shown in Section 4.1.

Checker The checker runs in configurable intervals and checks which flows are outdated and therefore can be removed or need to be preserved.

Dumper The dumper thread writes the captured packets to a *pcap*-file. The content of this file is determined by a filter expression written in the *packet-filter language* (similar to expressions used by `tcpdump`) and will be compiled to a *lua* function with pflua [20].

3. RELATED WORK

Observing the round trip time of TCP flows has been the topic of many research reports [5, 7, 16] and product of tool development [9, 21]. Two of these software pieces analyse the round trip time as an input value for further analysis of tcp flows. However, in this paper we focus on round trip time calculation. In [21] Zhang et al. present *T-RAT*, a tool to analyse the TCP flow rates. *T-RAT* takes traces of TCP connections and tries to infer the limiting factor on the flow rate. Integral to this analysis is the calculation of the round trip time (RTT): The RTT is calculated based on groups of packets forming a flight (as illustrated in [16]) based on

doi: 10.2313/NET-2018-11-1_16

the inter-arrival times. It then classifies the flight into one of the TCP phases (e.g., *slow start*, *congestion avoidance*, etc.). Shakkottai et al. report though, that this approach is controversial and not well defined [16]. Following the analysis of the TCP flow rates, Mellia et al. present in [9] another software to analyse and create different types of statistics on the IP and TCP level. The tool needs captured network traces (i.e., it only works offline) and will then extract the attributes and features of the flows needed for the statistic (on commodity hardware and retroactively). One of these attributes is the round trip time, where the authors gather the minimum, maximum and average RTT with respect to lost and retransmitted packets, also called the "Karn's algorithm" [6]. For the computation of the round trip time multiple models exist: Some only take the RTT during the Three-Way Handshake (illustrated in Figure 1) into account [5], while others evaluate if the RTTs are different during the phases of a flow [7]. Additionally, Fraleigh et al. point out in [5], that at the start of this century, it might have been more difficult to acquire a suitable dataset as it is now, due to limitations on storage space and available transmission capacities. More information about FlowScope can be found in [4], about MoonGen in [3].

4. IMPLEMENTATION

In this section we describe the requirements that FlowScope imposes on a user module, discuss the algorithm and environment we used to test our solution.

4.1 Required Attributes by FlowScope

FlowScope has the ability to be extended by a user module. This module needs to provide a minimum set of functions and attributes in order to use the flow tracking that FlowScope offers. See the Github repository [18] for the implementation. The lua module needs to export all attributes and functions that are listed below, only attributes marked with **optional** can be left out. In order to retain a clear structure, the required attributes and functions are sorted by the thread they belong to (this information was extracted from the example modules and by reading the source code in [14], especially the `exampleUserModule.lua` [13]). In order to implement a user module that is capable of measuring the RTT of TCP-flows, we implemented the following functions:

4.1.1 Flow Tracking

The *Analyzer* thread carries out the tracking of the flows. After a packet is inserted by an *Inserter*, the `flow key` needs to be extracted in the `extractFlowKey` function. Next, the `handlePacket` function is called with this packet.
C-structure types need to be defined with `ffi.cdef` function of the *luajit ffi* library [10] before they are known to the library. If a parameter needs to be set to a name of a *C-structure* the variable needs to include the whole name, i.e., "`struct ip`".

`mode` Either "`qq`" (to use the Queue of Queues data structure [4] / the ring buffer) or "`direct`" (to directly access the *NIC* without buffering)

`stateType` The type of the *C-structure* used to carry the state of this flow. Due to the implementation of the hash map in the background, this structure is limited to 128 B in size. With a change to the definition of the hash map in FlowScope this could also be enlarged.

`defaultState` This is an **optional** map datatype. The `stateType` will be initialized with the values defined here. Before applying the default values, the memory space designated for this structure is filled with zeros. Then, the default values are applied (see the description of `ffi.new` in [10] for an extensive description). If this map is undefined or empty, the values are set to 0.

`flowKeys` A `flowKey` is a *C-Structure* that defines how and on which attribute the packets should be associated. In case of a IP-flow, the five-tuple outlined in Section 2.1 constitutes a suitable flow key. The `flowKeys` attribute expects a map that includes all *C-Structure* definitions for possible flows. The size of a flow key is limited to 64 B.

`function extractFlowKey(buffer, flowKeyBuffer)` The function is used to extract the features of a flow from the `buffer` (which contains the whole packet including all headers from layer 2 and above) and store it in the `flowKeyBuffer`. Therefore the `flowKeyBuffer` is a pointer to a memory location that is large enough to store the largest `flow key`. The function returns **false** if the packet should be ignored and not stored at all or (**true**, i) if it should be stored, where i indicates which flow key was used (based on the `flowKeys` map).

`function handlePacket(flowKey, state, buffer, isFirstPacket)` The `handlePacket` function extracts the information from a packet (contained in `flowKey` and in the packet itself, in the `buffer`) and stores it into the flow `state`. The `state` variable is initialized with the contents of `defaultState` if `isFirstPacket` = **true** or it contains the state of the flow as it was changed by the last `handlePacket` invocation. If the first packet of the flow is handled, `isFirstPacket` is **true**. The function must return either **true** if this flow should be archived after it has expired (see Section 4.1.2) or **false** if this flow should not be archived.

4.1.2 Checker Thread Configuration

The *Checker* thread observes the Queue of Queues buffer for expired flows and marks flows that are expired (i.e., flows that are seen as finished) for the dumper thread (refer to the next section for a explanation). All of the attributes of the *Checker* thread are **optional**.

`checkInterval` This attribute defines the interval in which the *Checker* thread runs (in seconds). If the variable `checkInterval` is not defined, the *Checker* thread is disabled and will not run, in this case none of the functions prefixed with "`check`" are called.

`checkState` This *lua* map defines the internal state of this *Checker* thread.

`checkExpiry(flowKey, state, checkState)` This function checks whether the flow with `flowKey` and `state` is expired. It also has access to the *checker*'s state via the `checkState` variable. The `checkExpiry` function should return **false** if this flow is still active. If the flow

doi: 10.2313/NET-2018-11-1_16

is expired, the expiry-timestamp *ts* is smaller than the current time, the function should return a tuple with (**true**, *ts*) and is, as a consequence, removed from the hash table and from the rules of the dumper threads.

`checkInitializer(checkState)` The `checkInitializer` function is called after the state of the checker thread(s) has been initialized by FlowScope. It can be used to initialize the state of the *Checker* thread (`checkState`) or to initialize libraries once per thread.

`checkFinalizer(checkState, keptFlows, purgedFlows)` `checkFinalizer` is handled similarly to `checkInitializer`. However, it is invoked when the *Checker* thread is about to be terminated.

4.1.3 Dumper Thread Configuration

The *Dumper* thread writes the captured packets based on a Packet-Filter expression to a *pcap*-file.

`maxDumperRules` This variable limits the number of applied packet-filter rules. If set to 0 FlowScope is not able to store any dumping rules and thus, no packets are saved in a *pcap* file.

`buildPacketFilter(flowKey)` The `buildPacketFilter` function creates a packet filter based on the `flowKey`. To pass the filter to FlowScope it is required to return a string in the *packet filter language* [19].

4.2 Employed Algorithm

We use a SYN based approach as proposed in [16]. The authors found that, despite the simplicity of the algorithm, other tested algorithms might not offer big advantages over the SYN based approach. Whereas the SYN based approach is easy to implement and is comparatively less resource intensive, which makes it suitable for online RTT estimation. The larger challenge in this proof of concept was the integration into FlowScope. As outlined in Section 4.1.1, it is not possible to store an unbounded amount of data per flow in the hashmap managed by FlowScope. This implies that we are limited to a maximum of 128 B for the flow state. 34 B are used by counters for the minimum, average and maximum RTT, together with counters for the number of packets and the size observed. Our data structure to store the sequence number and the timestamp is 8 B in size (Figure 3), so we can store $(128\,\mathrm{B} - 34\,\mathrm{B}) : 8\,\mathrm{B} = 11.75$ sequence numbers with a timestamp. So we can store eleven sequence numbers together in a list. Despite the implementation of the `extractFlowKey` function, the main point of the implementation was the creation of the `handlePacket` function.

5. EVALUATION OF THE IMPLEMENTATION

Figure 4 shows the average round trip time during different flows between a pair of IP addresses. These numbers are gathered based on a replay of a SSL-scan stored in a *pcap* file with MoonGen [3]. We observed that all reported round trip times are 0. We conjecture that a replay of a stored SSL-scan might not be the most suited material to test the algorithm. One of the most probable causes for the result in Figure 4 could be, that most of the ports are closed and

```
struct timestamped_tuple {
    uint32_t sequence_number;
    uint32_t timestamp;
};
struct flow_state {
    uint32_t byte_counter;
    uint32_t packet_counter;
    uint64_t last_seen;
    double avg_rtt;
    uint32_t max_rtt;
    uint32_t min_rtt;
    uint8_t tracked;
    uint8_t rtt_index;
    struct timestamped_tuple rtts[11];
};
```

Figure 3: The Used State C-Structure

did not send a packet or terminated the connection shortly. Based on Figure 5, we further assume that the maximum of 11 sequence numbers are a too little subset to calculate the round trip time for this packet sample.

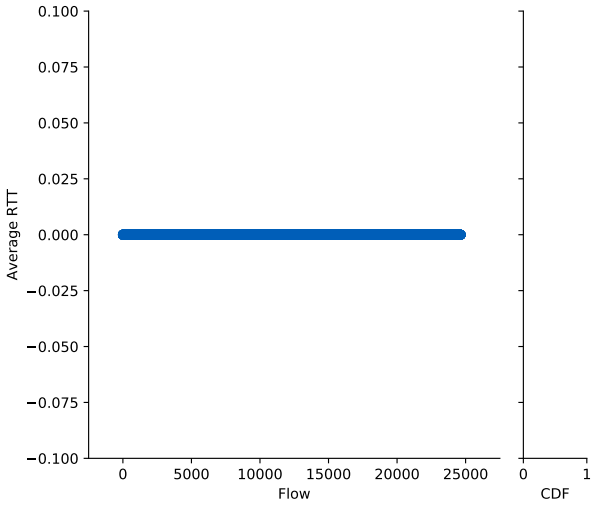


Figure 4: Round Trip Time per Flow Between Two Randomly selected IPs

6. VALIDATION

To further validate the aforementioned solution to evaluate the round trip time with FlowScope. And to confirm our assumptions about the number of sequence numbers we need to store, we created another approach that uses a dynamically expanding list of sequences.

6.1 Implementation of the New Approach

Therefore, we created another user module that is based on the previous approach but employs a dynamically expandable list to store the round trip times and sequence numbers. The captured samples of sequence numbers and round trip times are stored in a doubly linked list. The nodes of the list are *heap*-allocated and managed by the user module with `malloc` and `free`. Hence, we are not limited by the space constraints of the FlowScope hash map. As we are now able

doi: 10.2313/NET-2018-11-1_16

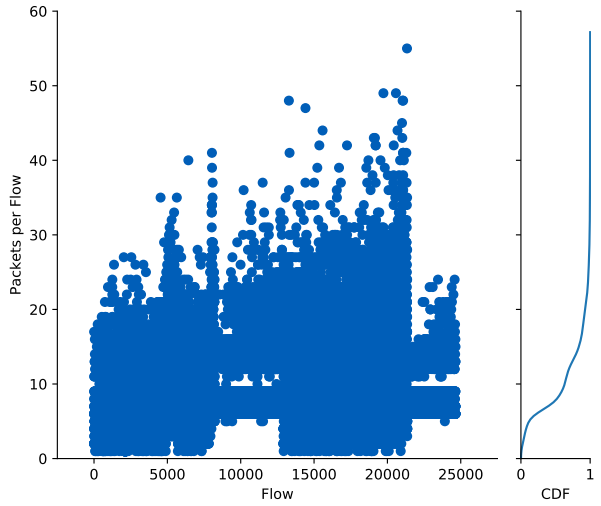


Figure 5: Packets per Flow Between Two Randomly selected IPs

to store every amount of sequence number that we capture, we are able to compute the average RTT after we finished collecting the sample. This opens the possibility to do further analysis on the round trip time. The improved flow state C-structure is depicted in Figure 6.

6.2 Evaluation of the New Approach

We tested the new approach again with an excerpt of a SSL-scan as before, albeit an different part of the SSL-scan. To reproduce this test with different data refer to Section 8. We found that the new approach yields results that we already expected from the implementation that used only a limited number of stored sequences. The data depicted in Figure 7 and Figure 8 was captured with the user module outlined above and flows with no RTT samples have been filtered out. We ascertain that the RTT is highly variant over the number of flows captured in this sample (Figure 7). To explore the distribution of the round trip times, the ten highest average round trip time are depicted in Figure 9. These respond initially promptly with an SYN-packet and then need apparently more time to process the next messages.

7. FUTURE WORK

Our original solution did not yield expected results. Conjectured cases for the shortcomings have been the properties of the SSL-scan or the previous calculation of the round trip times. However, these shortcomings have been caused by the limited amount of stored sequence numbers. As we found by employing a different approach in Section 6. To evolve this proof-of-concept into a better employable tool, it needs to be further tested with different packet traces that exhibit a different traffic pattern than a large amount of short lived connections or connection resets which might highlight further shortcomings of the two approaches.

Furthermore it needs to be evaluated, if FlowScope is the right tool for this task. Section 4.2 outlines the limits of FlowScope for the initial approach. The main limitation of FlowScope was the limited amount of storage available for

```
struct List_node {
  uint32_t sequence_number;
  uint32_t timestamp;
  struct List_node* next;
  struct List_node* prev;
};

struct RTT_List {
    uint32_t rtt;
    struct RTT_List* next;
    struct RTT_List* prev;
};

struct flow_state {
    uint32_t byte_counter;
    uint32_t packet_counter;
    uint64_t last_seen;
    uint32_t max_rtt;
    uint32_t min_rtt;
    uint8_t tracked;
    struct List_node* seq_list;
    struct List_node* ack_list;
    struct RTT_List* rtts;
};
```

Figure 6: The Improved C-Structure for the Flow State

each flow. Albeit, this could be mitigated in the second approach by using a heap allocated doubly-linked list which is not managed by FlowScope. However, manually allocating and disposing memory is prone to errors and an automatic solution to memory management is preferred here.

8. REPRODUCIBLE RESEARCH

We invite our readers to test our software and reproduce our findings with different input data. For this reason we provide the FlowScope modules at [18]. To run FlowScope with the user modules introduced herein, to follow the steps delineated below:

1. Server A and B need to be directly connected via network cards that are supported by the *libmoon* framework.
2. Install MoonGen [3] from [1] on server A.
3. Continue with installing FlowScope from [14] on server B.
4. Now, the repository with the FlowScope user module referenced by [18] needs to be cloned alongside the FlowScope installation.
5. Run `./libmoon/build/libmoon lua/flowscope.lua ../flowscope-tcp-rtt-analysis/src/TCPRTTTimeAnalysis_avg_w_seq.lua 0` on server B. This starts the collection of analysis data. To use the second approach from Section 6 substitute `TCPRTTTimeAnalysis_avg_w_seq.lua` with `TCPRTTTimeAnalysis_w_malloc.lua`
6. Now, to replay a captured *pcap* file, `./build/MoonGen examples/pcap/replay-pcap.lua -r 1 0 <path-of-pcap>`

doi: 10.2313/NET-2018-11-1_16

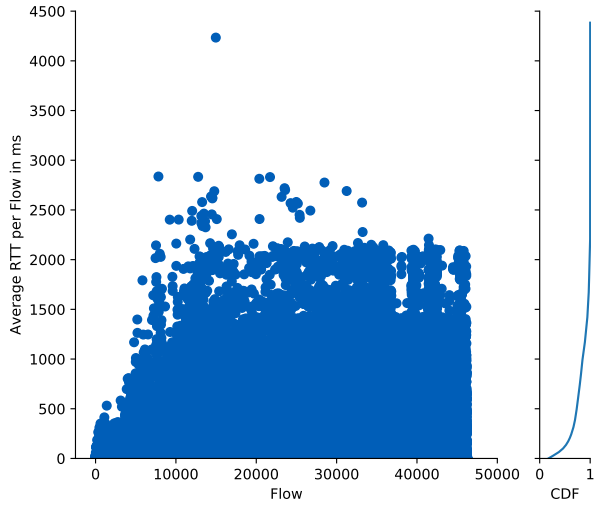


Figure 7: Average Round Trip Time Per Flow With the New Approach

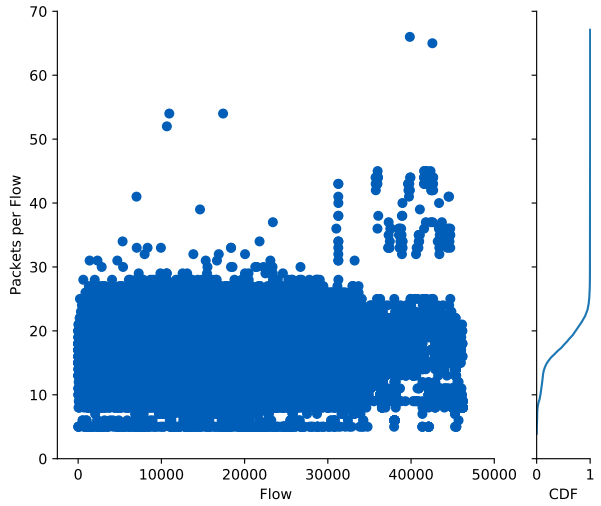


Figure 8: Packets Per Flow With the New Approach

needs to be run on server B within the MoonGen directory.

If these instructions deviate from the `Readme.md` of [18], please follow the instructions of the `Readme.md` as these instructions will provide up to date instructions.

9. CONCLUSION

With this paper, we contribute to the ongoing evolution of a toolset to measure and analyse high-bandwidth networks passively and during operation.

We started with a topic-focused introduction to the TCP protocol and the round trip time as a metric to measure the latency of networks. Then we continued to outline the requirements of FlowScope to a user module and the implementation of a module that collects data to measure the round trip time. During the implementation, FlowScope

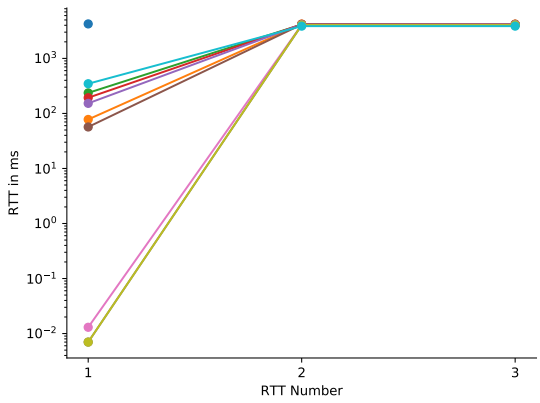


Figure 9: The Distinct RTT of the Ten Highest Average RTTs

showed some limitations. Mainly, the inability to store expanding flow status data, in our case, a list whose length is not known a priori. This is mitigated by the second approach outlined in Section 6. We further evaluated the possibilities to improve FlowScope to be suited for this matter and showed possibilities for further research.

10. REFERENCES

[1] P. Emmerich. emmericp/moongen: A fully scriptable high-speed packet generator built on dpdk and luajit. `https://github.com/emmericp/MoonGen`. Last visited 2018-06-14.

[2] P. Emmerich. libmoon: libmoon is a library for fast and flexible packet processing with dpdk and luajit. `https://github.com/libmoon/libmoon`. Last visited 2018-06-13.

[3] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle. MoonGen: A Scriptable High-Speed Packet Generator. *Proceedings of the 2015 ACM Conference on Internet Measurement Conference - IMC '15*, pages 275–287, Oct. 2014.

[4] P. Emmerich, M. Pudelko, S. Gallenmüller, and G. Carle. FlowScope: Efficient packet capture and storage in 100 Gbit/s networks. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 1–9, Stockholm, Sweden, June 2017. IEEE.

[5] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot. Packet-Level Traffic Measurements from the Sprint IP Backbone. *IEEE Network*, 17(6):6–16, 2003.

[6] P. Karn and C. Partridge. Improving round-trip time estimates in reliable transport protocols. *ACM Transactions on Computer Systems*, 9(4):364–373, 1991.

[7] H. S. Martin, A. J. McGregor, and J. G. Cleary. Analysis of internet delay times. In *Proceedings of passive and active measurement workshop in Auckland, New Zealand*, pages 1–8, 2000.

[8] S. McCreary and K. Claffy. Trends in Wide Area IP Traffic Patterns: A View from Ames Internet

Exchange. *13th ITC Specialist Seminar on Measurement and Modeling of IP*, (May 1999):1–11, 2000.

[9] M. Mellia, A. Carpani, and R. L. Cigno. TStat: TCP STatistic and Analysis Tool. *Quality of Service in Multiservice IP Networks*, 2601:145–157, 2003.

[10] M. Pall. ffi.* api functions. `http://luajit.org/ext_ffi_api.html`. Last visited 2018-06-14.

[11] M. Pall. The luajit project. `http://luajit.org/`. Last visited 2018-06-13.

[12] J. Postel. Transmission control protocol. RFC 793, RFC Editor, Sept. 1981.

[13] M. Pudelko and P. Emmerich. FlowScope/exampleUserModule.lua at 7beb980. `https://github.com/pudelkoM/FlowScope/blob/7beb980e2cb64284666ba2d62dda5727c7bfd499/examples/exampleUserModule.lua`. Last visited 2018-06-14.

[14] M. Pudelko and P. Emmerich. Flowscope. `https://github.com/pudelkoM/FlowScope`, 2018. See commit `7beb980` for the version used for this paper.

[15] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ecn) to IP. RFC 3168, RFC Editor, Sept. 2001.

[16] S. Shakkottai, R. Srikant, N. Brownlee, A. Broido, and K. Claffy. The RTT Distribution of TCP Flows in the Internet and its Impact on TCP-based Flow Control. *Cooperative Association for Internet Data Analysis (CAIDA)*, pages 1 – 15, 2004.

[17] The Linux Foundation. DPDK: Data plane development kit. `https://dpdk.org/`. Last visited 2018-06-13.

[18] C. Wahl. Tcp rtt analysis with flowscope. `https://github.com/sn0cr/flowscope-tcp-rtt-analysis`, 2018. See commit d73a208 for the version used for this paper.

[19] A. Wingo, J. Muñoz, and L. Gorrie. Pflang specification. `https://github.com/Igalia/pflua/blob/3c0b47078a24a0306e557af6063bc918e5897adb/doc/pflang.md`. Last visited 2018-06-14.

[20] A. Wingo, J. Muñoz, and L. Gorrie. pflua: Packet filtering in lua. https://github.com/Igalia/pflua, 2018. Last visited 2018-06-13.

[21] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the characteristics and origins of internet flow rates. *ACM SIGCOMM Computer Communication Review*, 32(4):309, Oct. 2002.

doi: 10.2313/NET-2018-11-1_16

ISBN 978-3-937201-63-4
DOI 10.2313/NET-2018-11-1
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)