jamk.fi

Practical Method for Understanding Resilience and Quality of Data Integrity in Embedded Systems

Author Tuuli Siiskonen

Master's thesis
June 2019
School of Technology, communication and transport information technology
Degree Programme in Information Technology
Cyber Security

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Description

Author(s)	Type of publication	Date
Siiskonen, Tuuli	Master's thesis	June 2019
		Language of publication: English
	Number of pages 85	Permission for web publication: yes

Title of publication

Practical Method for Understanding Resilience and Quality of Data Integrity in Embedded Systems

Degree programme

Master's Degree Programme in Information Technology, Cyber Security

Supervisor(s)

Rantonen Mika, Karjalainen Mika

Assigned by

Vatanen Marko

Abstract

Background for this work is to understand practical solutions to manage security threats imposed on data integrity in systems where automated decisions are taken. The task was to study, implement and establish a proposal for practical solutions to evaluate data integrity where the receiving party of data cannot trust the origin party of data, common data model is absent and automated decisions must be made reasonably fast. This is currently very common setup in the critical infrastructure of the society. The work review current practical computer system based solutions to manage data integrity, review current scientific research to find possible theoretical background for the solution and evaluate a solution based on machine learning and neural networks capable of classification and link prediction on graph structured data and producing graph embeddings. The solution aims to find practical use of graph embeddings in data integrity management. The solution was implemented based on unsupervised learning with Graph Convolutional Networks (GCN) and Variational Autoencoders (VAE). Adversarial data was simulated by creating malicious graph data using reinforcement learning. Data quality was tested using semi-supervised learning in Graph Convolutional Networks (GCN).

The proposed solution was able to detect data manipulation attacks against the generated data model. Graph convolutional network can be used in creating new data model through classification and link prediction for datasets from various data sources without common data model. Convolutional networks and sampling approaches used in machine learning can be used with analysis of cyber-physical processes that produce relational data. Due to its programmatic nature and high ability to integrate, unsupervised learning fits well in embedded system ecosystems control functions where automated decisions are taken.

Keywords/tags

Cyber security, integrity, machine learning, neural network, information security, data model, distributed computing, embedded systems

Miscellaneous

jamk.fi

Kuvailulehti

Tekijät	Julkaisun laji	Päivämäärä
Siiskonen, Tuuli	Opinnäytetyö ylempi AMK	Kesäkuu 2019
		Julkaisun kieli Englanti
	Sivumäärä 85	Verkkojulkaisulupa myönnetty: x

Työn nimi

Käytännöllinen menetelmä integriteetin hallitsemiseksi sulautetuissa järjestelmissä

Tutkinto-ohjelma

Insinööri (YAMK), Kyberturvallisuus

Työn ohaaja(t)

Rantonen Mika, Karjalainen Mika

Toimeksiantaja(t)

Vatanen Marko

Tiivistelmä

Lähtökohtana oli tarve ymmärtää paremmin tiedon muuttumattomuuteen kohdistuvien uhkien hallintaa käytännössä, erityisesti automaattiseen päätöksentekoon nojaavissa järjestelmissä. Työssä ehdotettiin ja toteutettiin käytännöllinen ratkaisu tiedon muuttumattomuuden arvioimiseen silloin, kun tiedon vastaanottava taho ei luota lähettävään tahoon, yhteistä tietomallia ei ole ja automaattisen päätöksenteon on tarve tapahtua suhteellisen nopeasti. Tätä tapahtuu yhteiskunnan kriittisessä infrastruktuurissa koko ajan.

Opinnäytetyössä tarkastellaan nykyisin käytettävissä olevien tietojenkäsittelyn keinojen soveltuvuutta integriteetin hallintaan. Opinnäytetyö arvioi viimeaikaista tieteellistä tutkimusta ratkaisun soveltuvuuden todentamiseksi. Tuloksena oli relaatiotiedon sekä todennäköisyyksien käsittelyyn pystyviin neuroverkkoihin ja syväoppimiseen perustuva ratkaisu tiedon luokitteluun ja linkkien ennustamiseen tietomallin tuottamiseksi sekä tiedon muuttumattomuuteen kohdistuvien hyökkäysten havaitsemiseksi. Ratkaisussa käytettiin konvoluutioneuroverkkoja puoliohjattuun ja ohjaamattomaan oppimiseen.

Ehdotettu ratkaisu pystyy havaitsemaan tiedon pahantahtoiseen muokkaamiseen perustuvat hyökkäykset. Konvoluutioneuroverkkoja voidaan käyttää puuttuvan yhteisen tietomallin muodostamiseen hyökkäysten havaitsemisen mahdollistamiseksi. Helpon yhdistettävyytensä ja ohjelmallisen luonteensa vuoksi algoritmit ja ohjaamaton oppiminen soveltuvat hyvin sulautettuihin ympäristöihin sekä ympäristöihin, joissa automatisoitua päätöksentekoa tapahtuu.

Avainsanat

Kyberturvalisuus, muuttumattomuus, koneoppiminen, neuroverkko, tietoturvallisuus, tietomalli, hajautettu tietojenkäsittely, sulautetut järjestelmät

Muut tiedot

Contents

Figures

Tables

1 Introduction

The connected society relies on speed and information availability. It relies on data, measurements, observations, analysis and notifications. Decision making is likely to be increasingly automated.

Because of need for speed or rather fear of latencies, data is no longer only collected in centralized manner but such services e.g. cloud services may suplement fog and edge technologies. Data is likely to reside both scattered, available close to devices, and centralized, managed and available for added value services in business transactions. The future is likely to hold much more. The data is being provided by or produced with e.g. embedded systems.

Data integrity and non-repudiation are important because this data is supporting decision making within the critical infrastructure and on top of it. If integrity of data is lost and decision making takes place based on that information, the consequences of decisions may be severe or even cause physical damage.

2 Structure of Study

At first, this study defines basic concepts. In the problem statement section this study describes the most important aspects or factors having influence in integrity in connected IoT, embedded and cloud ecosystems. Related works and Research Methods describe theoretical background of suggested approaches. Knowledge domains section introduce data engineering and prior 3rd party implementations used. The implementation plan section describes how the technical setup was built and how the tests required to carry out this study were conducted. Results, Conclusions and Discussion sections discuss the results of tests and concludes recommendation for practical method to maintain situational awareness on integrity.

3 Integrity and Other Important Concepts

3.1 Internet of Things and Cloud

Concept of Internet of Things (IoT) originates from Auto-ID Lab in Massachusetts Institute of Technology (MIT) and since 1999 and means everyday objects as well as the critical infrastructure, i.e. embedded systems being enabled with connectivity to Internet. Internet of Things research begun with finding ways to practically giving things digital identities that could attributed with location and many other business attributes that would in turn benefit business processes. The unintelligent things got identity and real-world visibility in computer systems through e.g. RFID technologies. The concept of IoT has emerged to cover almost any device that can be reached through Internet or local area networks remotely and more importantly those devices that run some automation or independent process that the industry, consumer or the society depends on.

Embedded systems are computational units with its own, typically real-time operating system and programmatic logic that define their role and participation in larger ecosystem of embedded devices.

Cloud can be understood through Cloud computing technologies or through an idea of storing vast amounts of data in data storage infrastructure that is both capable of being online and capable of making data globally available without affecting the user experience due to geographic location of a user of that data. Cloud computing in this case means running full manageable virtual infrastructure with application programme interfaces (APIs) to invoke and revoke resources and even automated invocation and revocation of server and network resources according to need and load.

Cloud ecosystem means offering and using data as a service through productized APIs in enterprise context and where business processes can be made rely on data provided or collected through those APIs. The idea of Internet of Things in enterprise

context has been presented for example by Haller, Karnouskos and Schroth (Haller 2008). In their article Haller et al. (Haller 2008) state that there are two main paradigms from which the business value can be derived. The other one is real-world visibility and the other one is business process decomposition. In practice, real-world visibility means bringing the data close to consuming enterprises who in turn are capable of creating unconventional business models. Real-world visibility of a device brings the data at hand and the business models can be enabled through use of cloud computing infrastructure.

3.2 Integrity, confidentiality and availability

Integrity and repudiation or non-repudiation are close relatives. Non-repudiation is about a system's ability to e.g. prevent end-user from denying his or her prior action ever taking place. Integrity is the 'I' in the CIA triad. Integrity is a quality of constancy of information or even a binary. It means that information or the bits are consistent and in their intended accuracy and form are as those were meant by the original creator. If system's information about user's prior activities cannot be trusted anymore, this falls into the category of repudiation. In light of this study, even though not officially part of the CIA triads, consistency seems to be important concept for managing situational awareness (Tipton 2007).

Consistency means condition to adhering together or ability to be asserted together without contradiction. This is familiar concept from hand-shake principles applied in data packet exchange in computer networks and therefore defining concept for all messaging protocols ever implemented. Even though Internet of Things or Smart Grid Devices may not be conventional personal computers communicating over TCP/IP, CAN bus of a car or programmable logic controllers used to orchestrate nuclear power plants very often communicate over some kind of messaging protocol.

In this study, availability plays an important role. Availability is the 'A' member of the CIA triad. Availability means that information is available when it is needed but as

opposite to disclosure, availability means that information or service is available for those who have need to use them. Unavailability may be result of a security risk that materialize as a threat, e.g. in form of denial-of-service attack (Tipton 2007). Within recent years, development of cloud technologies, availability is still there but the meaning of it has gotten new flavours. It used to running cold or hot spare system but now new strategies of resilience have entered the domain of availability. Availability may be reached through deliberately dying compute nodes, not fixing the problem but rather discarding the trouble maker and continuous integration automation to ensure steady flux of adequate amount of resources. Availability or resilience are not conceptually close to integrity but it might be possible that with cloud environments allocating resources dynamically the resilience strategy may have at least indirect effects on our perception about current state of integrity. Incorrect or falsified data may also be considered as factor to unavailability of correct situational awareness. Taking out the ill-behaving nodes may in some cases improve the availability of correct situational awareness.

Confidentiality is the ‘C’ of the CIA triad. For the purpose of this study it might be useful to look at the CIA triad the cryptographic academics are using, where the ‘A’ stands for authenticity rather than availability (Ferguson 2003). In this case authenticity does not mean any of the practical implementations to make software, devices or users to authenticate themselves against set of technologies or access management in general. But as Ferguson and Schneier (Ferguson 2003) suggest, it rather means idea of separating cryptographic authentication key from the key that is being used to seal the message, the encryption key. This key is called MAC or message authentication code.

3.3 Encryption and signing

Encryption is about the confidentiality and encryption does not stop the adversary from changing the message and hence tampering with the message integrity.

Digital signatures can be used to prove that a certain message was sent by certain source. In practice the digital signature is computed by a device that cannot be trusted and hence the value of integrity proof can be questioned (Ferguson et al. 2003). Digital Signatures are pieces of data that have been encrypted. Encryption is based on ciphering algorithms or ciphers that together with proper key or key pair produce a cipher text, an encrypted clear text. There are asymmetric and symmetric ciphers.

Confidentiality can sometimes be achieved by using cryptographic means to protect the data either at rest on in transit. According to Kerckhoff's Principle, cryptographic cipher algorithms are public. This means that anyone can check the soundness of the algorithm without loosing their secrecy (Ferguson 2003). Keys are the key to the secrecy, not the algorithm. Asymmetric cryptographic cipher can be used when there is a need for the sender to conceal or encrypt the message being sent using the cipher algorithm and for the receiver of the message to be able to read it. Symmetric-key algorithms are used as well where there is no need to be able to re-reveal the cipher text, the encrypted message. It is used e.g. to create electronic signatures. Asymmetric ciphers require key-pair. Asymmetric key pair consists of private and public key. The senders may encrypt the message with their private key and the receiver of the message has obtained their public key and is thus able to decrypt the message and read it clear text (Ferguson 2003).

Another important feature is the randomness of the key used to create the digital signature or a cipher text. The randomness depends on how random number the Random Number Generator or RNG can create or how high the entropy is. It is about probability of a certain value to exist in randomly generated set of values. There are only some processes in computing devices that produce good entropy and as often as computing systems use Random Number Generators or RGNs, those utilize Pseudorandom Number Generators or PRGNs. Pseudo-random means that the data is not random. A deterministic algorithm produces pseudo-random data. The reason is that pseudo-random data is easier to produce and it is computationally secure. It is

secure as long as an attacker is not able to break the PRNG. Even if the attacker might see some amount of data produced by the PRNG, she should not be able to guess what some of the following values produced by the same deterministic algorithm are (Ferguson et al. 2003).

Key distribution is a process where public keys or symmetric-keys are being delivered to the recipient using varying methods. This is sometimes considered to be the weak point of encryption systems. How to deliver the key so that cannot be caught by a middle-man while in transit? The other challenge to solve is to deliver the key to the recipient in the first place. Loosing a public key is not such a bad event after all because public key is meant to be publicly available. Loosing the secret key is much worse but in the public-key cryptography the idea is to distribute the public keys rather than the private ones. Symmetric-key cryptography does not offer this protection, however. Currently most computer systems rely on public key infrastructure (PKI). PKI in turn relies on public-key encryption and it is a formal agreement and a system to manage public key cryptography (PKC) in practice.

3.4 Security models

Basic security concepts are weaved into security models that should be used as part of the architectural design of a device or larger system. A security model is either technical or process based formal statement of the security features to be provided by the system, describing required, allowed and prohibited relationships between objects. The security models take into account respective security clearance and classification of the object. The security models of industrial device networks can often be similar to those of corporate networks; however, additionally the security models of industrial systems comprise of software architecture and network protocols and device composition that is not typical of corporate networks. Security models from different worlds from different era meet.

According to NIST, a threat is "any circumstance or event with the potential to adversely impact organizational operations”. It is realized potential for security incident to take place. Incident is "An occurrence that actually or potentially jeopardizes the confidentiality, integrity, or availability of an information system or the information the system processes, stores, or transmits or that constitutes a violation or imminent threat of violation of security policies, security procedures, or acceptable use policies” (NIST 2008).

3.5 Indicators of compromise

Indicator of compromise (IOC) is a trace or footprint that indicates a system is being under compromise attempt or has been compromised, i.e. a security incident has taken place. Intrusion detection and intrusion prevention systems typically search for traces or footprints from the network traffic and the footprint information is usually publicly available. The footprints can be specific to unusual behaviour of systems or users and can be traced both from networks and the systems monitoring and logging. IOC is different from malfunction or error as malfunction or error is basically identified substandard behaviour within parameters defined for the system or systems. Sometimes identified erroneous behaviour can be a sign of compromise e.g. when the erroneous behaviour is result of unusual use of the system. Security models can be built to take into account attack surface. In addition to understanding what needs to be protected from harm, the security model can be built to slow down the potential attacker or make the attacks so resource consuming that the attacker withdraws through controlled attack surfaces. The security model can also contain ways to return the system onto its initial state despite attack attempt, ways to monitor security events and even handle the attack itself.

3.6 Situational awareness, machine learning

Situational awareness in this study means understanding the goodness of data integrity. This is more than situational awareness on goodness of security measures against planned security model or information security management system requirements or ability to detect security incidents. Goodness of data integrity is neither measure of data quality but in essence the measure of how confident one can be about if and how much can one trust on data gathered. Situational awareness is often based on identifying and understanding significant patterns that seem to depict current state and evaluate possible future options. Here, pattern detection in cyber-physcal systems means analysis of spatio-temporal data streams to find significant patterns (Spezzano 2015). In order the decision making to be effective, change detection in the patterns is required and the spatio-temporal data streams very often do not come with labels and as Vallim et.al. suggest, detection should take place in unsupervised manner (Vallim 2012).

Machine learning is ”the application and science of algorithms that make sense of data” (Raschka 2016). Deep learning means applying algorithms that are able to indepedently discover hidden structures within the data set in order to make predictions. There are three kinds of machine learning: supervised, unsupervised and reinforcement learning. Supervised learning attempts to perform based on already known answers by developing a model based on those. In reinforcement learning, a reward signal makes the learner system to improve its performance and the goal is to create measure of reward and system that changes on its performance based on maximum reward, by trial and error. Unsupervised learning, the data set is unknown or its structure is unknown and it is possible to find answers without consulting a pre-existing model or a feedback loop that gives hints on the right answers. Classification of the raw data is one of the tools used in supervised learning but classification methods fall in both supervised and unsupervised categories and in unsuervised learning such a toolset is called clustering (Raschka 2016).

Deep learning is often based on neural networks. Neural network is a multilayered set of neurons where one layer consists of one or several neurons or learners. Neurons may or may not be fully connected to next layer's neurons (Raschka 2016).

Graph convolutional networks are essentially convolutional networks (CNN). Convolutional networks are feed forward neural networks where not all neurons connect to the neurons on the next layer (Ketkar 2017). A convolutional network aims to locate the value that is shared a cross a data point or a domain in data set and organizes these values, nodes, into multi-scale hierarchy or a pattern (Deferrard 2017).

Graph convolutional network aims to organize these nodes into a graph structure that depicts which nodes are connected within the pattern by finding the maximum modularity as the clustering method (Brandes 2008).

Auto-Encoders were developed for learning based on propabilistic models. An Auto-Encoder is a Stochastic Gradient Variational Bayesian (SGVB) estimator to optimize a recognition model in order to perform an approximation of a data point in the future using ancestral sampling (Kingma 2013).

Reinforcement Learning (RL) is a family of machine learning algorithms. One of them is Markov Decision Problem (MDP). These algorithms aim to learn optimal solution by asking feedback and learning from the mistakes until optimal solution has been reached. Reinforcement learning is good match for what to do or how to map decisions where the learner will deduce the best choices based on best reward (Sutton 2018).

4 Related work and contributions

4.1 Transaction Management

Transactions management and data integration are two topics that have long history of research within computer science. There is extensive research on this area since

consistency of data in modern computer systems has been rather relevant matter in solving practical corporate problems through computing. The field has evolved from relational databases to workflow management and data analysis systems. Some thought on integrity management has been given there as well. Research paper by Stefan Böttcher is a good starting point for setting the background for this study on integrity management in IoT cloud ecosystem and the sensors.

In his paper Concurrent Checking of Global Cross-Database Integrity Constraints (Böttcher 2002) Böttcher states that the problem of managing integrity in combining large seemingly similar to purpose but with differences in data models and business cases. He uses a case of corporate merger as background to his ideas. In such circumstances, it is of importance to merge data of some independently functioning local databases into one corporate data asset. Böttcher's study concerns management of data integrity in a large distributed data body through idea of lock management and of locking the integrity constraints itself.

Böttcher (Böttcher 2002) offers interesting model for lock management of the integrity constraints. The model supports transaction management strategies and places integrity management firmly in conjunction with transaction management. Transactin management aims to control transactions so that the internal integrity of a dataset is maintained. The distributed database system forms an integrity domain. This integrity management model does not cover countermeasures against tampering the data at the source, i.e. outside the data body and especially Böttcher's model and transaction management in general do not consider detecting signs of attempts to cover the tracks of an attack against the data sources before the data enters the integrity domain offered by the system of local and global databases (Böttcher 2002).

Transaction protocols can be classified according to e.g. synchronization strategy and synchronization granularity. Transaction management takes reconciliation approach inside the integrity domain. As an example, database entries may be written in order

and prevent concurrent database writes through locking and releasing locks after a transaction is complete. This idea of transaction domain is present with building the current Big Data Analytics as well as corporate workflow management systems (WFMSs) or with high-throughput messaging systems employed often in Big Data management solutions. This study does not aim to control integrity within the transaction domain but rather seeks to find evidence of state in integrity with existing data sources, i.e. sensors rather than the data stores.

4.2 Trust and Authentication or Certification

In related research, data integrity is quite often studied in conjunction with application architecture or network architecture, which lead to discussion about assurance and trust in between different components of given architecture. Interestingly, Irvine and Levin (Irvine 2002) discuss in their work A Cautionary Note Regarding the Data Integrity Capacity of Certain Secure Systems integrity as a capacity of a system but define data integrity again within one integrity domain. Their “premise is that builders and buyers of systems designed to provide high assurance enforcement of security policies should be aware of the impact of component and architectural choices on the integrity of data that users intend to protect.” (ibid.) Irvine and Levin are concerned about high assurance systems in the wake of commercial software and components entering the high assurance architectures. Although commonplace today, the observation meant that bringing 3rd party components in play made it necessary to consider application architecture from fenced integrity management point of view. Fenced means that the security model takes into account how less trusted components may or may not affect data integrity of a system that has both high assurance components and less trusted components present. Irvine et al. mention Biba as example of such security model to be used to address the integrity model as well (ibid.).

Biba was the first attempt at an integrity model. The purpose of the Biba model is to address the first goal of integrity: to prevent unauthorized users from making modifications to the information. Just as reading a lower level can result in the loss of confidentiality for the information, reading a lower level in the integrity model can result in the integrity of the higher level being reduced. Biba model sets integrity labels to all components. The Biba model starts with an idea that a subject may modify an object only if the subject's integrity label is stronger or dominates the object's integrity label (Tipton 2007).

Quite rightly, Irvine et al. criticize lack of integrity and confidentiality value and aim to study how the integrity value of data changes while it passes from one component to another. Irvine at al. point out that increasing the confidentiality widens the set of accessible data but increasing the integrity actually works out the other way around. They suggest that reasoning this is difficult and may lead to need to identify each component in respect to other components so that it's position and integrity label can be determined (Irvine 2002).

There would be need to identify and authorize a component witin system architecture. Access and authorization is mixed up with integrity management. Irvine et al. seem to develop their ideas around using integrity as measure of trust or assurance for programme code since methodical, high-assurance development process may produce high-integrity code and software with "this designation, the product may be deemed suitable for handling data within a certain confidentiality or integrity range." (Irvine 2002). It goes without saying that some integrity model needs to be implemented. Since embedded systems sometimes form busines driven ecosystems that aim to collect data about the state of a cyber-physical process and control it, integrity model introduced in business environments might be a fitting choice. An example of such integrity model is Clark-Wilson integrity model (Tipton 2007).

IoT or embedded devices may be so simple and low-footprint that it may not be possible to establish intra-device fencing needed to establish trusted computing zones to ensure component integrity nor prevent unauthorized modification of the data. Irvine et al state that “a computer system can only be trusted to manage modifiable data whose integrity is at or below that of its user interface and application components”. Following this analogy, in IoT cloud ecosystem this integrity model would lead to continuous mistrust where untrusted components would modify the data every time it reaches the IoT cloud ecosystem.

At the same time Irvine et al. refer to guidelines set by National Computer Security Center (1993) in Understanding Covert Channel Analysis of Trusted Systems. Accordingly, the observation about how being able to run untrusted applications on top of trusted systems without undue loss of security is a major tenet of trusted computer system (National Computer Security Center 1993) . It is possible that this is true within military type of trusted computer systems and even in corporate security domain where at least some parts of the systems could be considered trusted. However, within IoT sensor networks and data collection structures all parties and components must be considered untrusted and the security models in Internet of Things may require further assessment on integrity management.

In fact, the integrity capacity of the IoT or embedded systems’ cloud data is the same as that of a sensor sending its data to the cloud. Irvine et al. (Irvine 2002) conclude that if one part of the solution is on lower assurance level than some other part of the same solution, the solution is as good as its weakest link and that there are situations when corrupted data may prove to have significant consequences. These include e.g. high reliability embedded systems or systems where human life might be affected by improper execution of code. To push the limits further, this study aims to find solutions for understanding proactive or even dynamic ways to measure or test and affect the integrity capacity as both of these consequences are quite possible in the current day networked devices producing networked or relational data.

4.3 Integrity in Security Models

Access and authentication management as well as transaction management require somewhat less randomly changing environment. On ad-hoc networks, Stajano & Anderson (Stajano 2002) made observation that “Peer-to-peer and ubiquitous computing systems involve many principals, but their network connectivity is intermittent and not guaranteed. Traditional approaches to authentication from Kerberos to public-key certificates, are therefore unworkable, because they rely on online connectivity to an authentication or revocation server.”

Stajano & Anderson (ibid.) introduce in their work The Resurrecting Duckling: security issues for ubiquitous computing the concept of secure transient association. The ubiquitous devices act like ducklings that associate themselves to the first thing they see and unlike the ducklings, the devices would ideally re-associate or re-imprint themselves when the master releases them. This is certainly pioneering work on new security models needed with ad-hoc networks.

Stajano & Anderson (ibid.) suggest that for integrity management purposes a sensor or device in general should be equipped with functions that provide tamper evidence, information that can be used to detect if a sensor or device has been tampered with in order to make it send false data. The only weak link in the the security model or chain of controls is the attacker having unmonitored physical access to the device and that this should be tackled with tamper resistance features. In case of remotely connecting devices Stajano & Anderson (ibid.) state that “The main objection to this strategy is that it breaks open the loop of machine-based verification. A physical seal’s integrity cannot be verified as part of the authentication protocol; instead, it requires human inspection. Some might see this as a security vulnerability, but it could actually be an advantage. It means that the responsibility for protection rests with the person relying on that protection, rather than with some third party who might have different motives.”

Management of integrity seems to be outsourced again, it falls outside of one relationship and requires forming another relationship to manage the security. As Stajano Stajano & Anderson (Stajano 2002) present, they recognize that device to master device or software forms one relationship and someone checking the the remote device physically form another relationship, this time with the owner of the master and slave or master or slave.

This study, though, aims to keep management of integrity within one relationship and assumes that sensor devices are likely to be rather independently operating as those may not have enough value and complexity to the owner to be interested in each and every device. Say, we have smart pill that is meant to travel through patient's intestines and collect information or administer drug to a specific location among the organs. The smart pill's relationship with data collection device cannot last too long and the smart pill may have been designed to be disposable. In this kind of scenario, the first and second principle of resurrecting duckling apply. The first is called two states and it means that a duckling or a device is either imprinted or imprintable. The second one called imprinting, meaning the process of a device becoming associated with a master.

The third principle is death, referring to process where a device becomes imprintable again. The 3rd principle works well with the independently operating devices but the fourth principle of assassination no longer does. The assassination principle assumes that it is in the interest of the attacker or the device master keep control over the device or to keep the sensor or device sending false data and hence prevent programmed or remotely controlled death of a device by the device master or the attacker. If we turn our attention to managing the big picture of integrity, towards situational awareness of integrity, it becomes more relevant to understand how many or how small an amount of imprinted devices is actually sending data with uncompromised integrity and hence, what is the quality of decisions being made or analysis being done based on the sensor or device data. This study aims to explore active assassination, or rather actively allowing possibly compromised devices to

become imprintable again as a tool to manage the overall picture of integrity in cloud ecosystem. Understanding the goodness of data rather than detecting advanced persistent threat in a system or systems becomes of importance in environments where decision making is highly automated.

4.4 Integrity Management through Digital Signing

Tamper evidence seems to be important concept. This has been showcased e.g. with Stuxnet, a complex industrial malware used to attack Natanz nuclear power plant in Iran turning out to be cyberwarfare weapon directed at Iranian nuclear weapon development. (Langner 2011).

“Contrary to initial belief, Stuxnet was not about industrial espionage: it did not steal, manipulate, or erase information. Rather, Stuxnet’s goal was to physically destroy a military target—not just metaphorically, but literally. The real attack was not against SCADA software—it was aimed at industrial controllers that might or might not be attached to a SCADA system. To get rid of another misconception, the attack was not remotely controlled, either — it was completely stand-alone and did not require Internet access. The command-and-control servers that Stuxnet contacted appear to have been used primarily for evidence of compromise” (Langner 2011).

Stuxnet exploited Siemens 315 and 417 Controllers. Remarkable feature of these attacks according to Langner (ibid.) was that “Stuxnet was a stealth control system that resided on the controller alongside legitimate code, which continued to be executed. The fake data that the Siemens Controller 417 attack code fed to the legitimate controller program was recorded from real, unsuspicious inputs. Stuxnet replayed pre-recorded input to the legitimate code during the attack”.

Langner’s observations continue with suggested mitigations: “The major vulnerability that Stuxnet exploits is that present controllers do not allow for digital code signing. A controller treats code—as long as it’s syntactically correct—as legitimate, no matter where it came from.” Langner points out, however, that

compute-intense calculations may not be possible due to scarce resources and heavy emphasis on runtime performance (Langner 2011).

Recent proposal of Keyless Signature Infrastructure (KSI) proposes solution to document signing. According to Buldas et al (Buldas et al. 2013) KSI15 is based on keyless hashing functions that create signatures and borrows from blockchain. “Hash-tree timestamping uses a one-way hash function to convert a list of documents into a fixed length digest that is associated with time. User sends a hash of a document to the service and receives a signature token—proof that the data existed at the given time and that the request was received through a specific access point.” (Buldas et al. 2013)

Being able to prove that a piece of data existed at given time does not make any statements about the correctness or integrity of the process like in the case of Stuxnet. Timestamps and differences in signature seeds may provide randomness needed for digital signing. Even Langner (Langner 2011) suggests independent monitoring as the next best solution. This study aims to find practical solutions to monitoring of integrity in IoT sensors and devices or embedded devices.

4.5 Integrity and Intrusion Detection

Hong, Liu and Govindarasu (Hong et al. 2014) suggest a “network-based cyber intrusion detection system (NIDS) using multicast messages in substation automation systems (SAS)” based on IEC 61850. “NIDS detects anomalies and intrusions that violate predefined security rules using a specification-based algorithm. The performance test has been conducted for different cyber intrusion scenarios (e.g., packet modification, replay and denial-of-service attacks) using a cyber security testbed. The IEEE 39-bus system model has been used for testing of the proposed intrusion detection method for simultaneous cyber-attacks” (ibid.).

NIDS type of systems will certainly provide some visibility on signatures of attack activity on network level. When it comes to integrity based on time stamps as it is

the case with Keyless Signature Infrastructure, Hong et al. (Hong et al. 2014) point out that the time service or GPS time available in power grid substations may be susceptible to cyber-attacks against time synchronization. Source of time may thus be unreliable if time is needed in the process.

In 2013, Hong et al published another work on Integrated Anomaly Detection (ADS) for Cyber security of the Substations. “The host-based anomaly detection considers temporal anomalies in the substation facilities, e.g., user-interfaces, Intelligent Electronic Devices (IEDs) and circuit breakers. The malicious behaviors of substation automation based on multicast messages, e.g., Generic Object Oriented Substation Event (GOOSE) and Sampled Measured Value (SMV), are incorporated in the proposed network-based anomaly detection” (Hong 2013). Hong et al (ibid.) describe the temporal change as n bit matrix where one bit represents existence or absence of a security event and a matrix row represents measurement point in time. A bit is set to a number between 0 and 1 if an event occurs and remains 0 if not. For each substation, an index is calculated after measurement of events for given time period. The index can be compared with neighbouring substations. This is signature based evaluation.

Hong et al (ibid.) suggest to use message header information and timestamps to deduce if an event is indicator or not. The main assumption of the temporal anomaly detection for host-based anomaly detection is that the engineering software and hardware are able to generate system and security logs. This would require that there are capabilities to do this resource hungry activity somewhere close to the source and this approach probably is possible with electric power network substations and alike with multiple components and possibility to complexity. However, simpler sensors may need collection point where network traffic could be monitored and this may not be the case. More importantly, how would this approach predict or identify falsified data being fed into normal traffic cycles? The contents of data do not affect the object reference of the dataset being sent.

The Stuxnet infection had two phases. Firstly, the attacker infected the ICT infrastructure that was running in front of the SCADA controllers to make it a dropper of the attack code. The actual attack code, the second phase of infection, that was meant to affect the centrifuges in Natanz nuclear power plant was in the malicious ladder logic. The malicious ladder logic was dropped to the systems by replacing certain key driver file with manipulated one. The attack code in SCADA controller was basically in dormant state but activated independently every now and then to run its course. In the dormant phase it basically recorded data from normal activities. While active, the attack code had ability to provide pre-recorded data outbound to outputs of the SCADA controller hence keeping the controller in presumed state of normal conditions and functions but at the same time running ladder logic that would eventually make the centrifuges break. “Stuxnet is very difficult to detect. Antivirus software could not have prevented the attack because they have difficulties in dealing with low-volume threats.: the first signatures were issued after the worm had been in circulation for more than six months.” (Das et al. 2012). Hence, Stuxnet affected both integrity of the system assembly and integrity of the data being produced about the state of the centrifuges. And the first phase of the attack, even though closer to traditional ICT domain, was difficult to detect due to inherent feature of malware analysis and detection.

Stuxnet analysis showcases that there are those software and firmware components that are important when trying to understand the attack surface against data integrity. In their work, Hong et al (2014) have recognized that the cyber-attacks are targeted against certain components of the substation. They also point out that power grid substations have good degree of standardization in them as the standardization work has benefitted the industry. “The main goals of substation automation standards are: (1) interoperability, (2) simplified configuration, and (3) long term stability. Interoperability enables substations to accommodate intelligent electronic devices (IEDs) from different vendors”. There are differences in implementation between device vendors, though. These differences can be related

to e.g. to scheduling of messaging. The power grid substations may have some cyber protection like firewall component in them. This however, may not be the case with all components of the critical infrastructure or the simplest sensors and IoT devices or embedded devices.

5 Problem Statement

5.1 Managing Data Integrity

This study suggests solution for predictive caching based data integrity loss detection in distributed computing systems, e.g. embedded systems, automation systems.

The main challenge in data and device integrity management of connected but distributed computing systems is their restricted resources to provide metadata about data integrity by the data source itself, e.g. to provide structured tree representations of birth certificates for data.

This is particularly problematic on one hand if data is being gathered from continuously changing systems or on the other hand, if large amounts of data are being gathered from separate systems producing data. When the datasets are removed from the context of original source and the dataset does not bear with it any indicators of integrity or other means to deduct the integrity, it seems quite difficult to control the integrity of any data produced or any actions taken based on such datasets.

This happens in connected embedded systems that form an ecosystem that collects data in order to harvest the added value provided by that data. These ecosystems may form backbone of business of significant amount of private companies or even the whole society may rely on this information from its critical infrastructure.

5.2 On Building Situational Awareness

If one looks at how much weight is being put in keeping the average application database or cloud content data in order in terms of integrity management, it is noticeable that the integrity of the data relies very much on the following idea: If the data can be handled only by those who have permission to do so, the data must be correct. This is the first challenge.

What actually happens in an IoT or embedded systems ecosystem is that the central data storage or cloud operates based on role based access control model and all activities meant to verify the integrity of the incoming data are supplements to the core access control model and may even be based on merely access control or protection of the traffic rather than integrity checks. Hence, the weakest link remains to be the sensor or the controller itself, the source.

The second challenge is data models and how to maintain the trustworthiness of constantly updating data. Additionally, each data source can be understood as a layer of data that reside under or on top of other layers of data. Each layer may well have data points that no other layer has but the layers must have minimal set of comparable data points. Geographic Information Systems are good example of location being the minimum commonly shared data point that in turn makes it possible to geoprocessing algorithms to consume all layers' data points and conduct analysis. In this case, set of layers may also translate as distributed independent data resource infrastructure. Hence, the commonly shared data model can be minimum set of common data points that are used to normalize data sets. These minimal shared data points can be called hooks. Hooks support analysis but may not have anything to do with updating data within different layers. Extreme example of this is any relational database that handles changes through database locks. The database locks are means to queue management that ensure the data model remains intact and data is being written in orderly manner. The database locks keep appends in order but do not really indicate the state of integrity of the source. Yet another

aspect of layering is the fact that if analysis is carried out using several layers of data and one or more layers are flawed, the analysis result will be negatively affected.

One aspect to consider in distributed computing environments is that very often low footprint sensors are not necessarily capable of producing and storing log data. Data about systems' behaviour may not be available. Low footprint systems are often restricted on resources available to implement traditional access control and authorization mechanisms and therefore implement different kinds of access control models at all. It is natural for IoT and embedded systems ecosystems to grow and reduce connections and maintaining this ability is of vital importance. In dynamically developing connected systems especially the data models are in constant change but in addition to that these systems do not share means of access control nor authorization either.

In such distributed independent data source infrastructure, no unified, consequent and coordinated data model exists. Non-coordination in updating of data is natural and controls such as database locks cannot be used. Lack of coordination and consecutiveness asks for new approach on understanding the integrity management. There are few features of data that remain shared but one of those is definitely the relational or linked nature of the data.

The bottom line of integrity is, after all, this: does this data support decision making process? The raw data at the source itself has value, it stands out on its own.

5.3 Challenges of the Current Architectures

There are number of ways to draw visibility on state of the system in IT and ICT systems. Quite often these means rely on the solution's ability to produce knowledge of its own state and as often this ability is attempted to build in the architecture of the systems. Traditionally, unwanted security related behaviors within an IT or ICT system are considered to be found either from data about the physical or virtual devices and protocols used to run the physical process or from attempting to

understand what normal behaviour from security standpoint looks like based on specific business data, i.e. business behaviour. In case of Stuxnet, the business data remained nominal, hence making it difficult to base suspicion of foul play in data profiling. It can be argued that if the Siemens industrial controller had detection in place that had in turn warned about tampering of the industrial controller, the event would have been detected from the data about the physical or virtual devices used to implement a physical process. However, when the data are taken away from the original IT or ICT system, the bond between the original context and the original architecture disappear and cannot be used to deduce foul play, e.g. if the data has been tampered with.

6 Research Methods

Research methods used in this study will be explained in this chapter. At first the research is brought to the context of research traditions and the possible influence of research tradition is discussed. This study is also positioned in regard of that influence and the difference between research tradition and some possible theoretical background to be used in this research is proposed.

6.1 On Choosing Research Methods

Based on previous research and prior understanding of researched phenomena, quantitative postivist research (QPR) approach offers road to objective or near-objective reality (Khan et al. 2015) since computer systems are binary in nature. In objective reality truth is assumed and hence the reality becomes binary in nature. Something either exist or it it does not exist. Something is either true or untrue.

Additionally, QPR takes time factor into account as QPR assumes that it “will allways fall short of mark of perfect representation” and “assumes that constructs and measures of these constructs are moving toward perfection over the years” (Khan et al. 2015). This is analogous to basic functions of one of the more important

knowledge domains, the machine learning. Lastly, as QPR assumes that theory and testable predictions are to be distinguished (Khan et al. 2015) and that empirical data gathering is part of QPR, QPR forms solid foundation to research methdology. Presence of quantities is predominant (Avison et al. 2005) in this study.

6.2 Influence of the Methods Used

As previous research indicates, testing a theory through formulating hypothesis, translating the consepts into variables to be measured within collected data has been essential in researching phenomena around integrity. For example, Hong, Liu and Govindarasu look into one research tradition, namely intrusion detection and define research variables through recognized ways of the research tradition. They collected data to test the hypothesis of intrusion detection being the way to detect attacks. This is characteristic of quantitative research.

This study will follow this similar deductive empirical cycle (Jonker et al. 2010). But the previous reasearch has also proved that the matter of data integrity and non-repudiation can be tested or defined directly through examining the implementation of integrity. Previous research has been testing this indirectly. In case of indirect approach, loss of integrity or non-repudiation is being a interpretation of a consequence or something that follows. In the tradition of intrusion detection research, the detection of attack is the desired end result. Loss of integrity or non-repudiation is only deduced as the nature of the detected attack and other influencing factors from the detection environment are better understood. The previous research that has taken the direct path has often addressed integrity management through structures or means to label the data with metainformation about its state of integrity. It is thought that integrity can be somehow visual part of the data and integrity can be directly observed. Hence, when researching integrity or non-repudiation, either direct or indirect path can be chosen.

6.3 On Research Methods

6.3.1 The Path to Applied Application

This study chooses to take the direct path, as machine learning is for good measure based on mathematical modelling, algorithms that tend to be built so that the actual result is understood in one way, it is unique and direct. This study does not study algorithms as such but rather explores possibilities to build applied application based on mathematical models using quantitative methods. This study takes certain approach on machine learning itself in terms of how the data classification and prediction are being supervised. Additionally, some of the requirements for practical implementation are explored to better understand if the theoretical background supports the implemenation of practical application in choosing the mathematical tools, computational efficiency, sampling, data modelling, reliability or supporting testing and learning.

6.3.2 Designing Data Sets and Choosing Algorithms

Means to design the right kind of data set are in the core of this research. With the chosen tooling, machine learning, collecting the data comes down to designing the data set as much as defining the means of collecting data due to limitations on computing power or available time.

The emphasis is on ensuring that reasonably all possible data components or classes or subgroups are represented (Masters 1993) without exceeding the optimal data set size. Assessing optimal size of the dataset could be comparable to computing resource demand caused by incomplete or broken data. By choosing models that survive incomplete data may in turn require great deal of computing resources.

Supervised learning could be used to manage the need of resources and time. As Raschka (Raschka 2016) suggests with supervised learning, the training dataset should be used to train and optimize the model itself and the test dataset should

only be used to final evaluation of the model and, in addition, picking up and comparing certain amount of algorithms is important to train the best performing model.

However, when data is relational, large in size and unknown, unsupervised learning should be more suitable approach. This in turn may have negative impact in computational resource use. Hence, this study consider use (semi) supervised algorithms and efficiently implemented unsupervised algorithms for identifying the the data profile. Based on that profile to understand what the data profile is likely to look like in the future, where it is going to develop to. Graph Convolutional Networks (GCN) are capable of such classification and Variational Auto-Encoders (VAE) are capable of producing new data, like new images out of existing image corpus.

This study benefits from use of algorithms that are capable of classification and prediction because the observations, the raw data obtained from 3rd party systems is likely to have varying product and protocol based data models with relational structure, i.e. the labels and relations may be unknown with every new dataset entering the process.

Convolutional networks fit in predictions with input data having grid topology, e.g. time series or an image (Ketkar 2017). In order to adapt to the time and computational resources available semi-supervised learning should be good approach. Here semi-supervised indicates that the dataset used is pre-arranged to matrices with some outlier data reduction(Kipf 2016).

Pre-processing can be carried out using graph embeddings, feature vectors and one-hot labels. Graph embedding or graph is an index of neighboring nodes. Feature vector can represent labeled training instances or superset of labelled and unlabelled training instances. One-hot encoding is the benchmark of desired location of an observation within certain known class in form of a matrix.

These matrices are there to provide something that could be described as dimensionality in the dataset. As Deferrarard et al (Deferrarard 2017) suggest, with

convolutional neural networks the dimensionality manifests itself with network's ability to locate similar content independent of its spatial location. The dimensionality follows the feed forward nature of convolutional neural networks. The convolutional neural networks can be generalized to graphs and thus the model is computationally efficient.

Lastly, this study benefits from application of means to generate graph data. This can be achieved through Generative Adversarial Networks (GAN) proposed by Goodfellow et al. for testing generative models(Goodfellow 2014) or through Reinforcement Learning to learn the model (Dai 2018).

Generative adversarial networks (GAN) bear some similarity with Variational Auto-Encoders (VAE) in ability to produce new data. Generative Adversarial networks (GAN) consist of generator and discriminator and while the generator attempts to create synthetic data that the discriminator then attempts to classify as it would classify data that is known to be real. As soon as difference between these two is detected, the generator network's parameters will be corrected to produce better data, to fool the discriminator. GANs require some kind of a starting point to be able to make data with specific features. VAE, in turn, have the ability of looking back to the original data and this way the new data or the prediction is likely to have less striking anomalies or errors in new data than if produced with GAN (Patterson 2017).

When it comes to ability to look back to original, Reinforcement Learning aims to fool a known learner by learning an attack policy to create malicious data that could be able to lure the learner. This approach could allow for subtle and slow degredation of data which give the attacker the possibility to remain unnoticed over longer periods of time than simple introduction of complete but totally unrelated dataset. Stuxnet attack in Natanz nuclear facility was quiet and took place over a longer period of time.

GAN is rather a family of generative networks. The question in this case could be formulated like which Generative Adversarial Networks (GANs) generates the most

plausible graph data. GANs have been evaluated on use with Image data (Tsai 2018) but images are different from relational data and in this case, GAN must be able to produce graphs. Recent research proposes several interesting approches to this. In order to better understand the capabilities of different ways to generate networked data, it must be understood how network structures form (Dong 2017) in large natural networks. As an example, NetGANs have been proposed to be able to generate graph data using random walks, to mimic real world networks (Bojchevski 2018). Reinforcement Learning (RL) approach in turn needs some of the same input from the known learner, for example the prediction labels (Dai 2018) but the graph exist and RL approach can be applied to it.

Understanding the true capability of well trained model requires adversarial attack against it. This is different from using of mechanically broken or fuzzed datasets since the fuzzed datasets are not the result of a calculation that intends to produce synthetic data similar to data in the target system.

6.3.3 Collecting and Sampling Data from a Physical Process

In this case, the main data sources are physical processes where the phased nature of the process and linked nature of data are clearly visible in dataset. The cyber physical systems and processes are phased and phases are often scheduled based on control logic of the process. Phases may produce different sized samples of data with varying set of observations and features. Also, there might be many or just a few greatly dissimilar classes to be found within the data. The data collection may as well take place in such a moment in time where all the classes are not present in the datamodel and this is quite natural.

Since training is required with both GCNs and VAEs, the models might be incomplete with regards to new classes and relations within new observations or raw data over time. There are some strategies to manage such situations with convolutional networks, e.g. Partial Learning (Grisvold 2015), Stohastic Training (Chen 2017) or training with shallow neural network and large set of auxiliar data that adds to the

dimensionality of data (Zhang 2016), thus reducing the need of depth in neural networks. Just recently, Duvanaud et al. proposed (Duvenaud 2018) method to replace the hidden layers with ordinary differential equations (ODE).

In Partial Learning, the existing model could be fed as input in the following stage where training based on new features takes place. In Stohastic Training, additional algorithm is used to tone down the per-epoch cost of training. Also, Auto-Encoders can be stacked up (Gehring 2013) if needed for similar purpose. Gehring (ibid.) suggest that stacking Auto-Encoders could be useful when unsupervised training is wanted and if the amount of data is small, some pre-training can be applied. If Auto-Encoders rely on pre-existing GCNs, FastGCN could offer means to shorten time from data read to detection of integrity loss (Tengfei 2018).

These strategies may come handy with datasets with gradually changing data models and addressing the possible false positive detections and managing the need to re-train every time new dataset is read in. Also, there might be many or just a few greatly dissimilar classes to be found within the data. By choosing close to random seeds and while having many different seed groups and picking random seed "words" from several seed groups, it is probably possible to manage overfitting as well. Seeds are group of labels that may be indicative of the data (Yang 2016). Yang (2016) suggests using pseudorandom selection of seeds to be used further in creation of training and evaluation datasplits.

Another question is, from where the datasets should be collected. The embedded systems produce data about themselves and about the physical processes the embedded systems are to maintain. As an example, a chemical plant is built to host set of chemical reactions that take place independent of the mechanics of controllers and sensors and devices used to transfer data over networks. The chemical reactions would take place regardless if network was not be attached to it to convey the observations to monitoring and collection.

Together these two components form a supersystem that provides information. Which parts of this supersystem are vital when assessing the data integrity when the basis of predicting information produced is so fundamentally different? When it comes to using data provided by the IT systems, such as protocols, network devices and applications, this data suffers from the shortcomings of current architectures. The data produced by the physical process itself with data produced by the controlling logic of the physical process, for example a chemical reaction, is independent of these shortcomings.

As Rajkumar & Bardina (2003) point out, neural networks work well in predicting cyber physical systems such as test measurements and numerical simulations of aerodynamic coefficients. They discuss, “how many training data points are required to produce an efficient neural network prediction” (Rajkumar 2003) for subsystem that can be modelled as functions capable of predicting coefficients that describe the state of a physical process or a system. This paper assumes that other physical processes are likely to be similar. One of the goals of cyber-attacks may be to affect the physical process through the control devices like in case of Stuxnet infection of the Natanz nuclear power plant (Langner 2011).

Additionally, it is possible to model the physical process through modelling of the sensors and other field devices from the ladder logic (McLaughlin 2011) by reversing the programmatic representation of the physical process in a programmable logic controller or (PLC) and field devices it is controlling.

Physical processes or human interaction such as social media interaction can also be affected by manipulating the inputs of the process and cascading a causal failure effect. This was the case of Stuxnet where centrifuges broke as a result of dangerous configuration inputs. Many physical processes, e.g. chemical processes are such that cascading causal changes are in the very nature of the process itself. Manipulation of the physical process through chemical substance, heat, light or such elements that fall out of the computing environment takes place outside of the computing systems.

This leads to conclusion that data collection from cyber physical systems must not rely on signature data depicting the state of participating people, devices or networks. One must also consider the fact that the collected data is likely to have been removed from its source environment and the analysis takes place without possibility to access the source environment for any kind of verification.

This study relies on preceding research and implementations in terms of choosing sampling strategies. Data used in this study is the same as with research of Kipf (2017). Original dataset has been split into training and testing dataset without natural boundary between the two data sets. Additionally, programmatic masks have been used in sampling (Kipf 2017).

6.3.4 Loss function and other means of assessing quality and efficiency

The goodness of the classification but also, need to find the definitive links between data points are central for this study. At the same time, it must be possible to create datasets that are best possible to fool the classifier and link finder. Since the models have so distinctly different targets , the quality of each model is considered in the domain of each model separately. The aim of this study is not to evaluate the quality of proposed models nor test the computational efficiency of the chosen ones. This chapter intends to shortly evaluate the approach of quality assessment, efficiency and data structure impact on the goodness of each model for the purposes of this study based on previous research. Aim of this is to define how the chosen, existing models or approaches could or could not cover the requirements for chosen research approach.

The neural networks can cause natural data loss while attempting to classify data. This loss can be understood through identifying suitable loss function. Loss function depicts the goodness of created model through evaluating how well the model performs (Patterson 2017) . Kipf suggests that it is possible to create a graph structure otherwise than by regularizing the data with specific loss function. It is possible to carry out semi-supervised learning by introducing some suitably

structured data with known labels and at the same time, achieve improved granularity of data by finding new labels that do not yet exist. The classification goodness is evaluated by choosing their loss function to be cross entropy that measures how well the classification worked (Kipf 2017). This is in the best interests of this study as well since the goodness of data integrity evaluation is based on learner's ability to extend known classes with new adjacent ones.

The analysis dataset may have effect on how well the classifier and autoencoder will perform on filling the gaps in known facts or finding new probable links for a graph (Schlichtkrull 2017). The results were similar or less similar to results comparison group when the analysis data depicted presence of a feature or simple relationship. Schlichtkrull (ibid.) suggest that autoencoders would work well in combination with this kind of classifier in finding new relations. In terms of computational efficiency, existing research (Kipf 2017; Hammond 2011) suggest that there are means to implement computationally lighter spectral convolutions on large graphs where computational cost might grow too big.

Variational Auto-Encoders (VAE) have loss function as well. As Variational Auto-Encoders resemble data compressors, the loss function measures how well the compressed data is decompressed back to its original form and additionally, how much data is lost during the compression. The smaller the result of this loss function, the better the compression. When applying this to probability or predicting the future data, the target of training is to create a model that outputs good values close to certain likelihood, the closer the better. Again, this kind of approach on understanding the quality supports this study, since the target is to understand how links between nodes might or might not change and if the change can be considered trustworthy. Resent research suggest that the computational efficiency of VAE could be improved with multi-task learning (Phi 2018).

7 Knowledge Domains

This chapter describes the essential knowledge domains that support the practical implementation in this study as well as the solution.

7.1 Data structures and data profiling

Understanding and managing data structures of information systems, industrial automation systems and cyber-physical processes are essential in this study. The content and configurations of those can be understood as relational structures. When data is presented together with its relationships, together those form a graph.

When data and the relationships are disconnected from the original system or cyber-physical process, it already has data structure. A graph can be extended to contain data originating from several different sources.

A graph can be used as input to algorithm capable of creating a model. A machine learning model can learn to create a new data structure, e.g. classification or link prediction or image of a person that does not exist. For one, this study uses semi-supervised learning with graph convolutional networks for classification. Semi-supervised learning helps algorithm, like graph convolutional network, perform efficiently (Kipf 2017).

Data may require some pre-processing before it can be used effectively. For example, McLaughlin (McLaughlin 2011) reverse engineered the cyber-physical process from a Programmable Logic Controller (PLC). The approach of McLaughlin gives a clue on how some of the information needed to design data pre-processing could be retrieved. Another example of this are publicly available datasets like CiteSeer and Cora (Getoor 2019). The scientific community may go even further by lending conveniently processed datasets from one research to another. Kipf et al. used so called Planetoid data splits (Kipf 2017). Retrieving the relationships and the relevant features is often the aim in pre-processing.

This study benefits from using adversarial graph data generated or tampered well known scientific dataset. When a malicious dataset is generated by reinforcement learning, it is possible to produce data similar to data in target system but without dependency to same algorithm or method with which ws used to pre-process other datasets used in experiments.

This study resorts to using well-known open source datasets that have a track record of compliance within scientific research. This data has similar qualities to embedded system data with one-to-one and one-to-many relationships and known vocabulary or features.

Here, CiteSeer dataset (Getoor 2019) is used for explaining the pre-processing of data. CiteSeer is known text classification data set. The dataset is split into training and test datasets which have similar structure in terms of relations.

At preprocessing, the data is organized e.g. in matrices. *Adjacency, feature and binary matrices* (Kipf 2017) are created. With, feature matrix, it is possible to depict if a feature, in case of CiteSeer a word, exists in a node, in case of CiteSeer in a scientific paper. Adjacency or connectivity matrix aims to define how nodes of the graph connect to every other node of the graph.

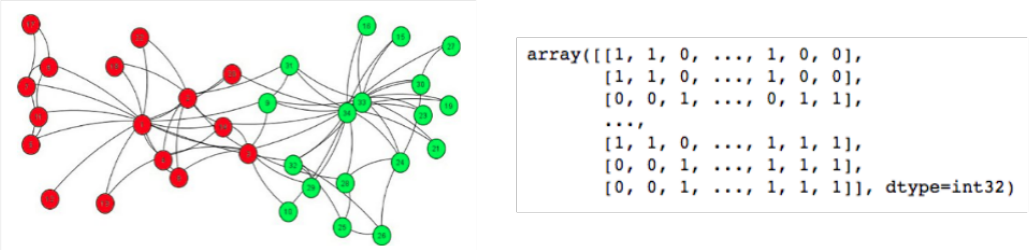

Figure 1. Classified graph and adjacency matrix

In figure 1, to the left, there is example of classified graph (Silva 2012) and an adjacency matrix to the right. Adjacency matrix is a representation of a graph in tabular form. The binary matrix depicts true or false value for a node or scientic paper on membership in a pre-defined class. With CiteSeer, the original data is

classified into 6 distinct classes. These classes are used as labels. The reinforcement learning implementation adopted here use CiteSeer dataset (Dai 2018) and have generated, e.g. a degradated dataset where the dataset splits conform with Planetoid (Yang 2016) dataset splits.

7.2 Neural networks and reinforcement learning

Graph convolutional networks are used in quality control of data needed in experiments. This study implement semi-supervised learning using graph convolutional networks based node and node feature classifier with training, validation of model and testing using masks for sampling of input data (Kipf 2017).

Prediction has been thought to be powerful in finding the missing information (Schlichtkrull 2017) and researchers have been looking for means to bring network structures, also known as graphs, with features linked with nodes or the whole graphs available for further inputs of algorithms. The research community has developed algorithms that can produce graph embeddings (Hamilton 2017).

Hence, Variational Graph Autoencoders are used for modelling relational data. Data modelling can be carried out by classification of nodes and node features or link prediction. Variational Graph Autoencoders that are based on autoencoders. The classifier uses GCN as encoder with softmax activation on each node and loss evaluation for each node. The link predictor use GCN as encoder and inner product as decoder. Loss is evaluated on each edge. Here, learning is unsupervised (Kipf, 2016). Further developed implementation in later research prove that it is possible to teach an algorithm to classify datasets MUTAG, AIFB, AM and BGC, each with their original data model into Resource Description Framework (RDF) data model (Kipf 2017).

Reinforcement learning has been used in data degradation attacks on graphs. This study use the implementation where 5 different attacks are suggested (Dai 2018), e.g. RandSampling , GeneticAlg, Exhaust and GradArgmax, and have tested these

vectors with graph classification and node classification in white box, practical black box and restrict black box scenarios. Reinforcement learning is used in node attacks and result is are verified against graph convolutional network. The implementation make use of recent framework RL-2SV capable of learning algorithms (Dai 2018).

RL-S2V seems to work well in black box scenario but there are other possibilities. E.g. RandSampling that deletes or adds relations in a graph and require least amount of knowledge about the target classifier. GrandArgmax makes selection based on gradient information in white box scenario. GeneticAlg works based on evolution like components like population, fitness, selection, cross-over and mutation and it is a white box method. From successfulness perspective, it seems to be significant if there is all or at least some information available about the classifier. White box scenario resulted in best in most cases (Dai 2018).

From perspective of practical implementation of security, the problems are four-fold. Firstly, relational data is in a format that has not been fitting for solving problems using machine learning algorithms. Secondly, cyber-physical systems not only produce relational data by nature but also tend to change their data profile over time as the system state or process state evolve. Thirdly, data degradation attacks against integrity of data are hard to detect without ability to predict the future and at the same time, manage poisoning of the ground truth to prevent false prediction of the future. And finally, when combining data originating from different sources in order to make automated decisions, the first obstacle to overcome is to establish the missing common data model.

Graph convolutional networks and autoencoders can be used to refine graph data into form that algorithms can consume. Classification and link prediction are tools for data modelling. Those bring ability to predict the future (Kipf 2016 ; Hamilton 2017) and compare data profiles for detection.

8 Solution

8.1 On detecting loss of integrity

This work suggests and tests a solution for detecting loss of integrity with means of classification and prediction in distributed, low-power computing environment or when collecting relational data disconnected from original source into new data collection, such as, to cloud based data ecosystems.

In embedded systems and industrial automation, there is typically a control system to manage the field devices and collect field data. Sometimes the field data is collected in a historian database. With distributed computing like vehicle-to-vehicle communication or vehicle-to-infrastructure communication, the computing may take place at the edge instead of centralized management system.

The suggested solution comprises of pre-processing and two analysis cycles.

The pre-processing aims on one hand prepare raw data into form that is normalized for analysis purposes. This include cleaning raw data from incomplete or misleading data but also, choosing the features presenting the raw data the best. For example in citation datasets Cora and CiteSeer, features are bag-of-words that represent scientific content of the papers and relations are citations from one scientific paper to another. The pre-processing aims on the other hand processing the cleaned data into programmatic objects that can be processed with implemented algorithms. Pre-processing is also be needed when combining data from different sources.

The other analysis cycle, the *learning-testing cycle*, aims to predict what the future, known good data model will look like. The learning-testing cycle in Figure 2. should also include an evaluation loop of the trained model, as well as data pre-processing phase to create the dimensionality needed in training and evaluation. The result is displayed on a plot.

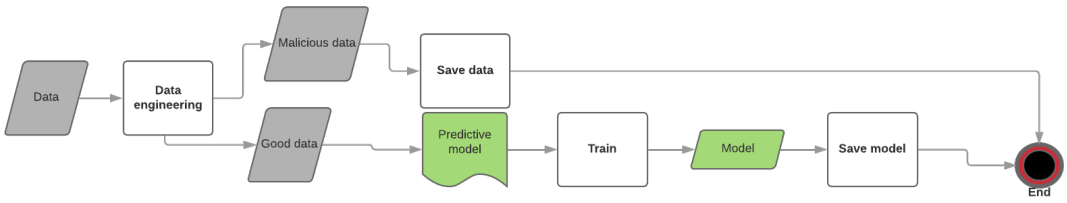

Figure 2. The learning-testing cycle for predictive model learning and testing

The other analysis cycle, the *prediction-detection cycle*, aims to predict if the candidate data is capable of producing similar data profile to that of already stored data. The prediction-detection cycle should include pre-processing of the candidate input data and running the input data classification and data profile prediction. After classification and prediction, the result is displayed on a plot and this plot is compared with the plot from learning-testing cycle as can be seen in Figure 3.

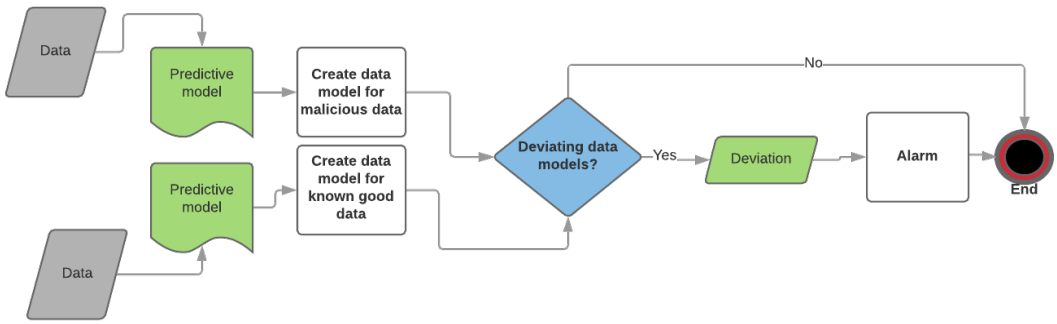

Figure 3. The prediction-detection cycle for predictive integrity monitoring

Detection takes place if the input data does not seem to be able to predict similar data profile as what the data that has already been accepted in has been able to predict. If the plots are significantly different, detection should take place.

As the system is learning from data and it cannot use any other knowledge to evaluate the reliablity of the original source of candidate input data, learner's behavior is dependent on incoming candidate data. To manage the possible learning of wrong data profile and thus succeeding in tampering with the integrity of data,

this study suggest consequent and independent prediction of data profile for the accepted data.

8.2 Attacks

In this solution, it is expected that the data producing physical process and its behaviour changes over time naturally and possibly due to adversarial actions as well. The system that collects data may not have control over nor the ability to review the process state. The data will have relations, which means there are characteristics or actors that become nodes. The data will have identifiable features and it is known if certain feature should exist in data or not. Here, the attacks are directed at nodes, classes and features. in data degradation attack, either amount or features of the nodes will be affected. In replay attack, same data is fed to the learner several times.

9 Implementation Plan

9.1 Set-up

9.1.1 Programming languages and libraries used

The programming was done with Python and several machine learning software libraries were used. The classification and predictions had been developed mainly using Google TensorFlow. For detailed list of programming languages and software libraries used are listed in Appendix 1.

9.1.2 Laboratory setup and experiment tracks

The solution was built with Docker images. There is one Docker image for each phase. Docker images are built based on Docker files. Each Docker file builds a

Ubuntu Linux container with software libraries listed in Appendix 1 and Appendix 2. Technical details of the laboratory setup are described in Appendix 2.

The laboratory setup was chosen this way for number of benefits. Firstly, the Docker containers made it possible to secure that contamination from another phase of the experiment was controlled. For example, dataset sources and target folders for possible outputs of the experiments could be defined unambiguously to prevent experiment contamination. Also, since it is necessary to control change in laboratory setup, containerization offered a self-documenting tool to manage installations and configurations through container configuration file, a dockerfile, in infrastructure as a code manner. Containerization was thought to support possible need to be able to move the experiments from one environment to another or scale the experiments since Docker containerization is widely supported technology. It was identified that during the experiments, there had to be a well-working mechanism to ensure that each of the experiments could be started from zero after multiple attempts to complete the experiments and thus avoid contamination of results by results or remnants from preceding experiments. Maintenance and build-up of the experiment tracks was quite straight forward using container technologies. Docker container itself offered a stable and proven server like environment where the 3rd party implementations of neural networks and other algorithms (Kipf 2016 ; Kipf, 2017 ; Dai 2018) could be proven to run as intended. The laboratory setup and experiment tracks were set up as follows.

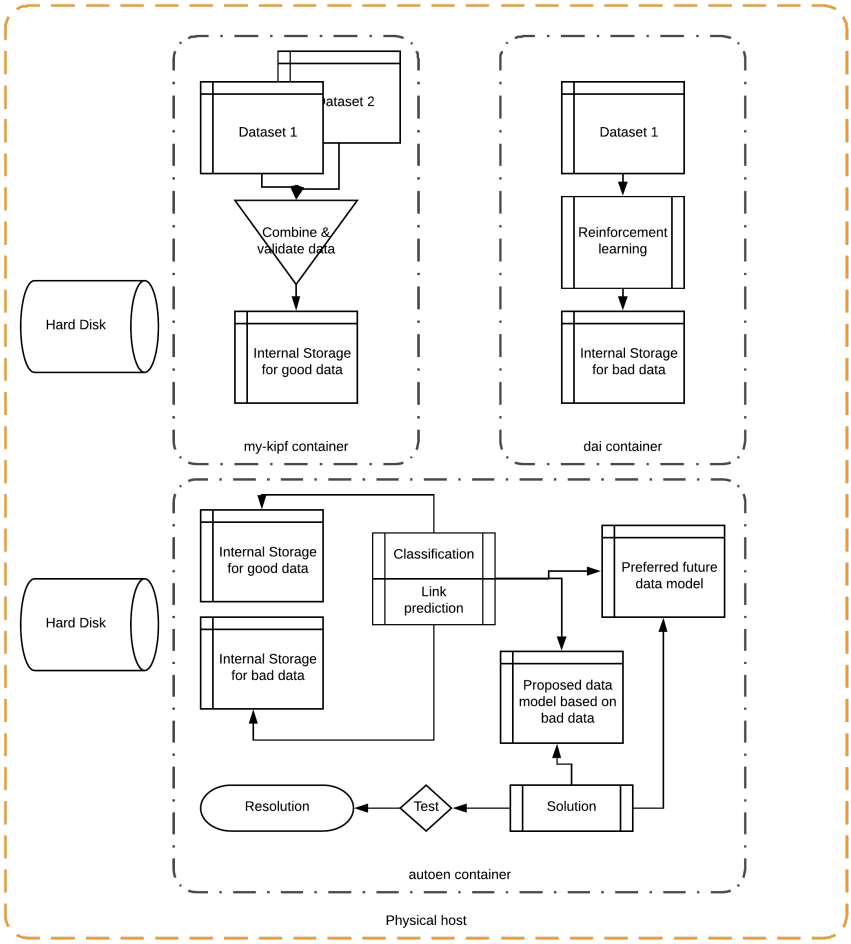

Figure 4. Experiment tracks

Each of the experiment tracks relied on dedicated container with container specific configuration file that was built with Docker and spawned as container. The datasets are explained in detail in chapter 9.1.3 Dataset engineering.

Data validation was identified to be needed to be part of the laboratory setup. Container my-kipf role was to test the success of data engineering where combined dataset was created. Combined dataset was to mimic the data assets received from multiple different sources. This 3rd party convolutional network implementation (Kipf 2017) was good fit for testing the form and completeness of a combined dataset, since it validates the form of dataset well for its original purpose to test and proves the efficiency of the convolutional network solution.

Container “reinforcement” role was to output the bad dataset through reinforcement learning and save files to predefined location. The dataset processed and altered in reinforcement learning was to mimic data from single data source. This 3rd party convolutional network implementation (Dai 2018) was good fit for producing test dataset for malicious data, since the implementation was originally created to produce graph based attack datasets to prove the usefulness of reinforcement learning with graph data.

Container autoencoder role was to carry out the classification and prediction. The experiment was fed with good data to produce the preferred, predicted future data model with combined dataset. Then, separately, this container was fed with bad dataset to see if the input dataset was able to produce anything similar to the preferred, predicted future data model. The solution proposed here was set up in container autoencoder to test the proposed hypothesis and produce the results.

9.1.3 Dataset engineering

This study resorts to using well-known open source datasets that have a track record of compliance with research. The open source scientific datasets used are listed in Appendix 1.

In pre-processing, CiteSeer for document classification and Cora datasets (Getoor, 2019) were merged into one dataset by appending CiteSeer file allx with and Cora file allx, CiteSeer file ally with and Cora file ally and CiteSeer graph file with Cora graph file. Similar operation was subjected with tx and ty files in both datasets. As a result, a new combined dataset ind.CiteseerCora was created.

For the 3rd party implementations described in chapter 9.1.2 Laboratory setup and experiment tracks, the combined CiteSeer and Cora data must be serialized or pickled as Python objects and these serialized objects are written on local datastore. The 3rd party implementations use Python serialization library Pickle or cPickle to

load the datasets into processing. In order to combine datasets, the datasets must be first deserialized, then combined and serialized again as combined dataset.

Y files were merged by feeding these Numpy arrays in Pandas data frames and concatenated with Pandas into new Numpy array. Before concatenation, another one of the Y arrays was added with the same number of columns as the data to be concatenated had classes. CiteSeer data is the starting point and CiteSeer Y data frame was added with 7 more classes to make room for Cora data. As a result, Cora's seven classes were concatenated with CiteSeer's six classes. Data frame rows were appended on top of each other since, in this case, both datasets are publication datasets and share dissimilarly named but content-wise synonymous column for research paper identifier. Simple append like this worked for these datasets for the time being but it is clear that more analysis needs to be subjected with data engineering when more structurally dissimilar datasets are combined. The original CiteSeer data split has isolated nodes in the graph. Those nodes were added as 0-vectors in CiteSeer ty.

X files were merged using SciPy hstack and vstack functions. Function hstack was used to extend the width of CiteSeer and Cora X files so that with CiteSeer, a block was added on the right side of its structure. With Cora, a block was added on the left side of its structure. This way both datasets made room for a block of data that was not originally part of the dataset. Then, with function vstack, CiteSeer and Cora dataset X files were appended with each other maintaining the order of CiteSeer and Cora data similar to Y and graph files.

In figure 5., the CiteSeer dataset looks light grey and Cora looks dark grey. The image is created from adjacency matrix and it depicts how CiteSeer and Cora data splits of were appended with each other. Light pixels show existing relation between nodes.

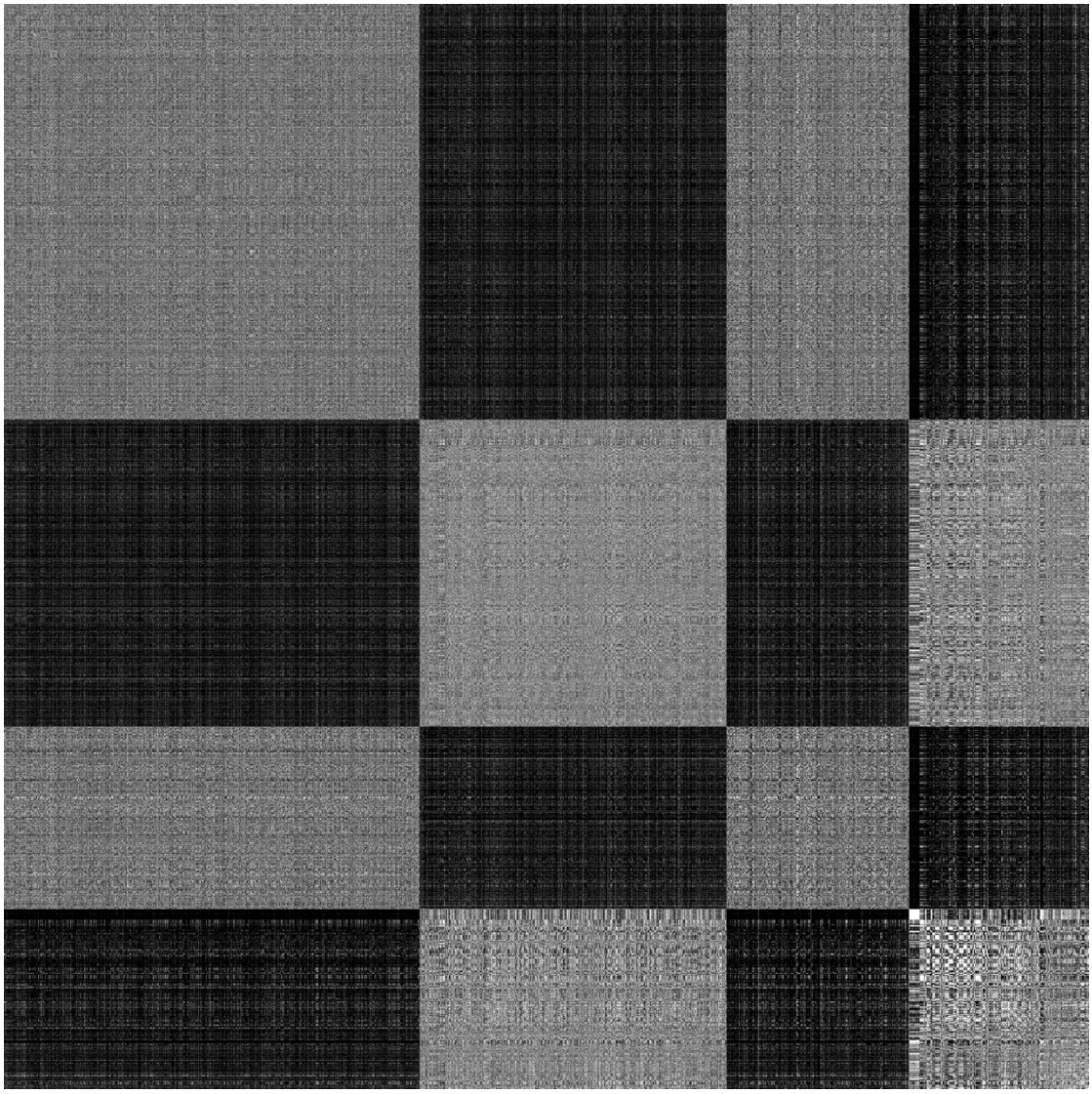

Figure 5. Combined representation of CiteSeer and Cora datasets.

With graph files, it was necessary to prepare for implementation of Graph Convolutional Networks where allx and tx data splits are appended in this order into feature representation in memory. As a result of this, the graph file had to be rearranged to match the feature representation order of alternating allx and tx dataset splits for CiteSeer and Cora. The order is: CiteSeer allx, Cora allx, CiteSeer tx, Cora tx.

Graph files were merged as Python defaultdicts. Since the CiteSeer graph was the starting point of this append, it was necessary to add Cora dataset's dictionary keys

after the CiteSeer dictionary keys so that the order of nodes or objects remained similar to their order in X and Y. Overlapping keys in datasets could be avoided. Since the dictionary in this case includes the linked nodes or objects as Python list of data in integer type, merging the dictionaries was not likely to affect the graph structure of the originating datasets as long as the order remained the same compared with X. The research paper idenfier was not in all cases resembling a number but had other characters in it. The indices of combined ind.CiteseerCora.test.index were rearranged to match graph order.

Attack dataset implementation use CiteSeer, Cora, Pubmed and Finance datasets (Dai 2018). CiteSeer is organised in similar data splits. The graph of CiteSeer is mutilated with reinforcement learning and allx and ally splits of CiteSeer ar altered to match the graph for this study. First, the degraded CiteSeer dataset is produced. Second, the degraded dataset is combined with CiteseerCora dataset resulting in new combined dataset, a malicious dataset.

9.1.4 Source Code Explained

In this implementation, the experiments were programmed using either Tensorflow or Keras tensors and consequent execution environment called Session. A tensor can be considered to be a group of functions which the execution environment processes. In this code, the functions typically perform data type transformations and mathematical functions.

Tensor is a convenient wrapper to provide valid return values within the context in the tensor or the tensor is used to perform learning with, e.g. optimizers. Typical objects to be handled are files and memory based representations of files in form of arrays, dictionaries and lists. Adjacency matrix can be used as representation of a graph. Adjacency matrix can be understood as a table, where nodes are placed on x and y axis of the table and existence of a relation between given two nodes is expressed with e.g. Boolean True or with a number like 1.

Graph Convolutional network can be thought as stack of programmed filters with feed forward nature to provide data to the following filter to achieve a goal. In figure 6 there is generalization of a graph convolutional network with input layer to feed in the graph data and other data like labels and bag-of-words (Kipf 2017).

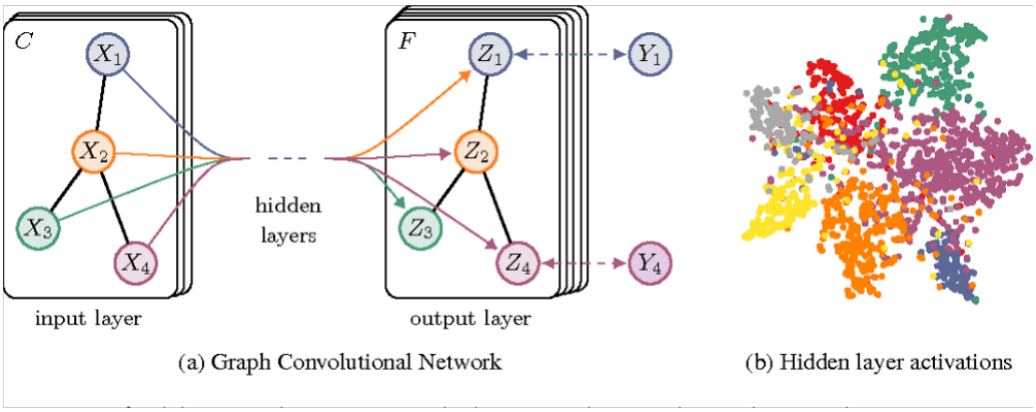

Figure 6. Graph Convolutional Network for classification

Each layer Between input and output layers, there is a hidden layer that transforms the inputs into form that the output layer can use. In this example, the hidden layer is capable of doing classification that is turned into node classification by output layer.

Training function plays major role. Training helps the model to learn optimal output in order to reach given target. Learning require iterations and following example in Frames form animation in Figure. 7 show on the left side how the learner makes per node classification primarily in CiteSeer originating class (0-5) and to the right, after number of iterations the learner begins to find class matches from Cora originating classes (6-12) as well.

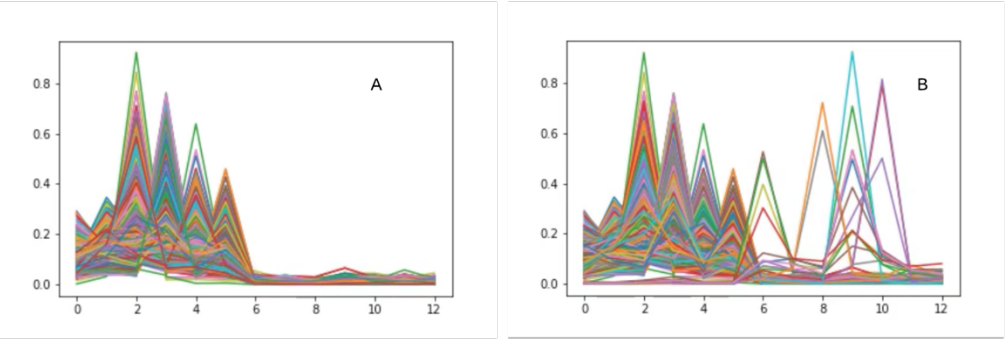

Figure 7. Per node iterations to find probabilities of classes

Training may require considerable amount of computing resources. Machine learning typically require good resource management and optimization of computing environment. Data type and array type transformations were often needed to manage resources. It require active memory and CPU processing power, too. Python was chosen as coding language since Python is powerful in processing arrays, lists and dictionaries and there are several well-established software development frameworks, e.g. TensorFlow and Keras, that support Python.

Data quality validation was implemented in my-kipf container as is basis based on existing opensource implementation (Kipf 2016) as described in Appendix 2. The source code itself was not changed in any way but relevant parts of it were brought in Jupyter notebook that was used, along Docker and Python, as an execution environment within the container.

With container Reinforcement, Docker and Python were used as execution environment to load pre-trained models for graphs created with RandSampling, Exhaust and GrandArgmax attacks. Software code of reinforcement learning implementation (Dai 2018) was changed so that attack scenarios could be run using CPU instead of GPU. The implementation provided number of pre-calculated models that were applied in attack code. The PyTorch library used supported GPU parallel computing that relied on NVIDIA drivers. Models were loaded using PyTorch (Dai 2018). In order to move the computations to CPU in an environment where there

were no NVIDIA graphics card or drivers working, GPU version of PyTorch 0.3.1 was replaced by CPU version of PyTorch 0.3.1 library. Additionally, Makefile and build.py files were modified so that NVCC GPU compiler was replaced with CPU based compiler. Additionally, PyTorch load function was instructed to move the computation in CPU by setting map_location parameter to follow the ctx command line argument value. While this ctx command line argument reads cpu, computations should take place cpu based. For this, files node_attack_common.py and node_dqn.py were modified. Docker container implementation is described in Appendix 2.

The autoencoder code (Kipf 2017), was brought to Jupyter notebook running on autoencoder Docker container. In Jupyter notebook, the autoencoder implementation was included in two workflows.

The purpose of the classification workflow was to predict new adjacency matrix and re-classify. Adjacency matrix is representation of a graph. The new adjacency matrix was included in semi-supervised classification with Graph Convolutional Networks, using datasets appropriate to attack scenario and predicting future data.

The classification workflow initiated first imports, configurations and most important functions needed in the following phases. Next, to create the encoder, model gcn_ae was trained. In order to predict new adjacency matrix, the trained model reconstructions were fed into Adam optimizer based decoder. Following in the workflow, the GCN layers and base class for GCN model was introduced. The new adjacency matrix was loaded in GCN classifier and after training the GCN model, softmax function based classification was returned from the workflow. Finally, the classification was written to a locally saved text files called predict, predicted and labels. Text file Predict contains probabilities for all classes. Text file Predicted contains decision on chosen class and probability used in making the class placement. Text file Labels contains the original classes provided in original datasets Cora and CiteSeer.

Link prediction through node and edge scoring is implemented with the encoder-decoder or graph autoencoder (Kipf 2017).

The purpose of the link prediction workflow was to predict new adjacency matrix and links. The link prediction workflow initiated imports, configurations and most important functions needed in the following phases. Before training any models, the malicious data splits for BadciteseerCora were calculated and stored on disk following naming conventions and divisions. Next, model gcn_vae was trained. The gcn_vae model was created and its reconstructions were fed in Adam Optimizer with BadciteseerCora data splits. Both known good data and malicious data were used in separate model training cycles and the resulting adjacency matrices, one for known good data and one for malicious data were compared with each other. Matrix cell values (1 or 0) were compared with same cell in the other matrix and if the values matched, Numpy equal function returned Boolean True. Other cells got value False. Mean was calculated by dividing amount of cells with Boolean value True by amount of all cells.

9.2 Testing

9.2.1 Validating data quality

The combined ind.CiteseerCora data splits as well as the Cora and the Malicious data splits were processed with graph convolutional network in my-kipf container. The dataset was brought available for training and the quality of the training and hence quality of the model was validated using graph convolutional network model (Kipf 2017). The test results were provided for cost, test accuracy and time.

First test case was carried out with pure Cora data splits using GCN model with original training, validation and testing masks inherited from the implementation for getting comparable test results on known dataset. The second test case was carried out using ind.CiteseerCora using GCN model with Kipf's original training, validation and testing masks. The third test case was carried out using dataset splits of

CiteseerCora using GCN model with balancing training, validation and testing masks. Finally, at fourth test case, the Malicious data splits were processed using GCN model with Kipf's original training, validation and testing masks for Cora. Table 1 describes test cases for data quality.

Table 1 Test cases for data quality against performance of a model GCN

Test case	Training, validation and testing masks	Dataset name
Test 1 data engineering	Original (Kipf, 2017)	**ind.Cora**
Test 2 data engineering	Original (Kipf, 2017)	**ind.CiteseerCora**
Test 3 data engineering	Balanced	**ind.CiteseerCora**
Test 4 data engineering	Balanced	**ind.BadciteseerCora**

Table 2 describes measured test cost, accuracy and execution time for each test case.

Table 2 Measurements against performance of a model GCN.

	Cost	Test accuracy	Time	Dataset name
Test 1 data engineering	1.01263	0.81400	0.01685	**ind.Cora**
Test 2 data engineering	2.06524	0.37450	0.03479	**ind.CiteseerCora**
Test 3 data engineering	1.38495	0.79050	0.04789	**ind.CiteseerCora**
Test 4 data engineering	1.49961	0.73250	0.03532	**ind.BadciteseerCora**

Classification was conducted for SiteseerCora data dataset. When comparing original classification of Cora or CiteSeer datasets and classification of combined CiteseerCora, the classification matched with original classes in over half of the cases. Table 3 describes how classification changed.

Table 3 Analysis of GCN classification introduced data model changes.

	Amount	%
Similarly classified	3644	60.38
Dissimilarly classified	2391	39.62
Total amount	3035	100.00

Amount of newly classified papers is two thirds. The model seems to perform within the targets of original research. The classification, if understood as data model, seems to indicate evolution of data model in combined dataset. The combined dataset seems to have dissimilar data model due to algorithm learning new classification. Table 4 describes how classification changed with autoencoder.

Table 4 Analysis of data model changes introduced by autoencoder.

	Amount	%
Similarly classified	2602	43.12
Dissimilarly classified	3433	56.88
Moved to class of other original dataset	248	4.12
Probabilities in other original dataset class	173	2.87
Total amount	3035	100.00

The data model of the predicted future data shows the following development. After prediction with autoencoder and re-classification the classification provided following insight into autoencoder's ability to suggest new data model.

Dataset used here is CiteseerCora. Comparison of classification was based on classes provided with original Cora and CiteSeer datasets and predicted classes. Similarly classified means that class is same as in the original class placement. Dissimilarly classified show prediction. Average prediction probability is 0.98 and its standard deviation is 0.08.

9.2.2 Data quality degredation attack

This test scenario assumes that the attacker attempts to fool algorithm with degraded data. With Graph Autoencoder implementation, Graph Convolutional Network can be used for classification and link prediction (Kipf 2016). It is possible to cause degradation of graph data by tampering nodes. Graph nodes are attacked using RandSampling , GradArgmax and Exhaust attacks (Dai 2018).

Data degradation attack is carried out by running prepared node attack models by Dai et al. against CiteSeer data split. The degraded CiteSeer data split is combined with original Cora data split into malicious dataset that is then run through data quality verification and then through autoencoder for link prediction. First, the degraded CiteSeer dataset, BadciteseerCora, is produced. Second, the BadciteseerCora dataset is fed in to graph autoencoder and the resulting adjacency matrix is compared with a adjacency matrix resulting in the CiteseerCora dataset link prediction.

In RandSampling attack, GrandArgMax and Exhaust attacks, the graph was considerably shorter than the graph of CiteseerCora dataset. The graph was populated with indexes equivalent keys in allx so that it reached the size of CiteseerCora data. The values were left empty with knowledge that the empty values

would be populated with self-reference as soon as the graph was turned into adjacency matrix at initial loading in of the data.

The two adjacency matrices are compared with each other. Cell values (1 or 0) are compared with same cell in the other matrix and if the values match, Numpy equal function returns Boolean True. Other cells get value False and as soon as all comparisons have been completed, mean is calculated. Table 5 describes similarity in data models.

Table 5 Similarity of data models in data degradation attack

	Mean
Exhaust attack modified data	0.56
Rand-Sampling modified data	0.57
GrandArgMax modified data	0.55

This means that the links in CiteseerCora data and maliciously modified data are different and the difference can be detected.

9.2.3 Record and replay attack

This test scenario assumes that the attacker attempts to fool algorithm with repeating same data. Data degradation attack is carried out by running prepared node attack models against CiteSeer data split (Dai 2018). Graph nodes are attacked using maliciously generated graphs that are grown to the size of CiteseerCora dataset by repeatedly populating the graph with data generated with RandSampling , GradArgmax and Exhaust attacks until there are no empty relations in the graph. The graph indexes are not altered.

The degraded CiteSeer data split is combined with original Cora data split into malicious dataset and then through autoencoder for link prediction.

The two adjacency matrices are compared with each other. Cell values (1 or 0) are compared with same cell in the other matrix and if the values match, Numpy equal function returns Boolean True. Other cells get value False and as soon as all comparisons have been completed, mean is calculated. This means that the links in CiteseerCora data and maliciously modified data are different and the difference can be detected. Table 6 describes similarity in data models.

Table 6 Similarity of data models in record and replay attack.

	Mean
Exhaust attack modified, repeated data	0.57
Rand-Sampling modified, repeated data	0.56
GrandArgMax modified, repeated data	0.56

10 Results

10.1 Data model for combined datasets

Data model was created by classifying the input data with graph convolutional networks on graph structure and the node features together. The data model represent the new data model of known good data. Graph Convolutional Network (GCN) is capable of creating the data model and this way, the original relations and features of data splits remain unchanged and available for other use. The solution is storage efficient, since the preservation of the new data model does not require restructuring of the data on disk but the data manipulation operations needed can be done without storing several concurrent datasets. The data model is produced when necessary using a model that the graph convolutional network has learnt. In

this solution, the balancing of data splits was carried out using optimized masks for training, validation and testing. The GCN model was able to reach similar testing accuracy to original data splits of Cora.

Combining CiteSeer and Cora datasets and evaluating the new dataset with graph convolutional network demonstrated that data engineering and merging of datasets must take into account how training mask, validation mask and testing masks are created for the combined dataset.

Original implementation use training, validation and testing masks the range of which is defined based on lengths of original datasets (Kipf 2017). When datasets are combined but the masks remain unchanged, the training and validation masks only pick data that originates from CiteSeer dataset. Testing mask, on the other hand, contain data originating from all datasets. Figure 8 describes lengths of the original masks.

```
In [6]: train_mask, val_mask, test_mask
Out[6]: (array([ True,  True,  True, ..., False, False, False], dtype=bool),
         array([False, False, False, ..., False, False, False], dtype=bool),
         array([False, False, False, ...,  True,  True,  True], dtype=bool))

In [7]: train_mask.shape
Out[7]: (6035,)

In [8]: val_mask.shape
Out[8]: (6035,)

In [9]: test_mask.shape
Out[9]: (6035,)

In [10]: sum(train_mask)
Out[10]: 260

In [11]: sum(val_mask)
Out[11]: 500

In [12]: sum(test_mask)
Out[12]: 2000
```

Figure 8. Original training, validation and testing masks.

By using the masks, it is aimed to ensure that training dataset is not overfitting (Kipf 2017). The training and validation in effect take place using CiteSeer, only.

The test results show that when the CiteSeer data was imbalanced in feature representation, the model did not perform as expected and testing accuracy was low

as shown in Table 2. The results also show that with balanced masks the model performed reasonably well compared with testing accuracy of single original data split Cora.

The experiments relied on balanced and widely used scientific data splits in order to ensure that quality of the experiments did not suffer from data quality and to manage the work needed in data engineering in preparing good quality and balanced datasets. Both Cora and CiteSeer datasets have undergone some choices and changes while those were prepared for publication. For example, in CiteSeer, every included scientific paper cites at least one other scientific paper within the corpus. CiteSeer include 3703 words after removing stopwords and all words that existed in less than 10 papers (Getoor 2019). Effects of misplaced or erroneous node-to-node connections, labels and words were not evaluated in this experiment other than following the original implementation in adding some isolated nodes as zero vectors in y data split (Kipf 2017).

While combining datasets, the datasets may become imbalanced. Imbalance became evident from the structure of the combined dataset and the Graph Convolutional Network performance. Original training, validation and testing masks were set up using the dimensions of uncombined data splits that did not take into account that training and validation data did not include balanced data from all included data splits.

Managing the balance is likely to increase the accuracy performance of convolutional neural networks (Hensman 2015). In this solution, the balancing of data splits was successfully carried out using optimized masks for training, validation and testing.

10.2 Controlling the state of Integrity

From perspective of practical implementation of security, the problems are tri-fold. Firstly, relational data has not been fitting for solving problems using algorithms before development of algorithms capable of dealing with relational data. Secondly,

cyber-physical systems not only produce relational data by nature but also tend to change their data profile over time as the system state or process state evolve. Thirdly, data degradation attacks against integrity of data are hard to detect without ability to predict the future and at the same time, prevent poisoning of the learner.

Graph Convolutional Networks and autoencoders bring the ability to predict the future and recreate and compare data models. It is proposed in this study that for a defender, it is important to be able to control the building blocks of graph data, e.g. the bag-of-words and nodes. If relationships between the nodes change and the building blocks remain the same, it can be detected with prediction of what the future data will look like. If uncontrolled changes in building blocks are allowed, the loss of integrity is harder to detect.

The management of data dimensions are in the core of machine learning and in this case nodes and bag-of-words provide the dimensionality. Management of dimensionality requires identification and control of these building blocks. From integrity model point of view, e.g. according to Clark-Wilson integrity model commonly implemented in business environments (Tipton 2007), the following elements can be identified in this study. Unauthorized and authorized subjects and object for regulated changes can be identified and well-formed transactions and segregation of duties and authorisation management can be implemented.

The management of who or what can change the building blocks of the learner forms natural boundary between authorised and authorised users. If a data provider is considered to be untrusted, unauthorised user, data provider can tamper the relations within the raw data but is unable to tamper with the building blocks of the learner by providing maliciously modified data. Machine learning libraries naturally raise errors in case of unexpected dimensions. Data dimensionality changes are naturally detected and can be recorded according to principle of well-formed transactions. Since embedded systems usually form a complex ecosystem where each computational unit forms its own sphere in terms of operating system, role

within the ecosystem and authorisation management, the complexity of authorisation management grows. As the subjects will not trust the data coming from the data provider Authorisation management should also take place where the building blocks of graph data are managed. Segregation of duties and authorisation management practices are being implemented in environments with embedded systems already and these capabilities can be used here. This means that active data engineering as well as secure systems and software development are required when identifying and controlling the building blocks of graph data. Data engineering may be a critical function or asset for the defender in terms of information security management system.

This implementation was able to detect changes in data by comparing two predicted data models. This implementation is likely to be prone to misinterpreting some input data as malicious while it is not since there is likely to be natural alteration in data, reflecting the state of the process, despite using validation on receiving operating characteristic (ROC) score and average precision score. While values for these scores remain high, typically higher than 0.9, the evaluation is carried out for local predictions on node edges and not how well the predicted links fit in spatial or temporal dimension.

Cyber-physical processes tend to have temporal and sometimes spatial dimensions. This implementation does not provide means to understand temporal dimension or e.g. varying distance between nodes. In order to enhance the ability to react correctly to natural alteration of data over time, the link prediction and classification implementation could be added with e.g. weighted connected edges for nodes, where the weight marks distance to another, connected node. The higher the weight, the later in time is the occurrence of the relation. With spatial or temporal enhancement, it would be possible to analyse the predictions on how different is different and define alarm triggers that are less often false positive. With temporal dimension, the implementation would learn regularly recurring elements or phases or programming sequences of the cyber-physical process. In industrial control system

settings, this implementation could benefit from making it learn the weights itself by analysing the programming sequences of embedded systems in the ecosystem.

10.3 Attacks and attack footprints

The adversarial data used in experiments was produced by reinforcement learner that was learning to mimic another algorithm until capable of outputting a graph that was malicious but which would yieald reasonable accuracy while subjected to the mimiced algorithm (Dai 2018). In the experiments of this study, similar results on model accuracy were achieved by training a maliciously modified dataset with Graph Convolutional Network. Hence, model accuracy change does not seem to be reliable attack footprint on its own, since the adversarial learners attack the model itself. Changes in learner behaviour, such as with model accuracy, may indicate qualities of the incoming data.

It was possible to see difference in link prediction by comparing the adjacency matrices created with autoencoders. Only about half of the maliciously modified adjacency matrix relations matched with the relations in known good data. Again, if spatial or temporal dimensions are added in comparison, the precision of data model grows and the attack footprint gets clearer.

11 Conclusions

The proposed solution was able to detect data manipulation attacks against the generated data model. Graph convolutional network can be used in creating new data model through classification and link prediction for datasets from various data sources without common data model. Convolutional networks and sampling approaches used in machine learning can be used with analysis of cyber-physical processes that produces relational data. due to its programmatic nature and high ability to integrate, unsupervised learning fits well in embedded system ecosystems control functions where automated decisions are taken.

12 Discussion

This study suggests practical solutions to monitoring of integrity in IoT sensors and devices or embedded devices. Graph convolutional network can be used in creating new data model through classification and link prediction for datasets from various data sources without common data model. The original relations and features of data splits remain unchanged and available for other use without the need to create e.g. a relational or non-relational database to manage the data model.

The solution is storage efficient, since the preservation of the new data model does not require restructuring of the data on new storage. Data manipulation operations needed can be done both in memory and by storing some training and testing datasets that are fraction of size of all data. It is possible to learn a model that can be stored and served later on if needed. The data model is produced when needed.

Convolutional networks and sampling approaches used in machine learning can be used with analysis of cyber-physical processes that produce relational data. Due to its programmatic nature and high ability to integrate, unsupervised learning fits well in control and monitoring functions of embedded system ecosystems where automated decisions are taken. It is proposed in this study that for a defender, it is important to be able to control the building blocks of graph data, e.g. the bag-of-words and nodes.

While convolutional networks seem to be very powerful, those have also limitations and vulnerabilities. Graph Convolutional Networks have vulnerabilities. that can be exploited with malicious data generated with reinforcement learning (Dai 2018). The research of neural networks is developing fast at the moment and it is clear that the qualities have not been explored in full, e.g., Graph Convolutional Networks may have limitations with very symmetric relational data. Also, both Graph Convolutional Networks and Variational Autoencoders could be enhanced on prediction, sampling methods and performance (Kipf 2017).

When it comes to practical implementation, further emphasis could to be placed on researching suitable set of means for data engineering. The data engineering operations play a significant role while preparing data for integrity monitoring. The practical solution could offer, for example, means to investigate the datasets in order to do efficient data engineering.

References

Avison, D. E., Pries-Heje, J. 2005. *Research in Information Systems: Handbook for Research Supervisors and Their Students*. United Kingdom: Elsevier Butterworth-Heineman. Accessed 10.3.2017 Source: Google Books https://eprint.iacr.org/2013/834.pdf

Bojchevski, A., Shchur,O., Zugner, D., Gunnemann, S. 2018. *NetGAN: Generating Graphs via Random Walks.* Accessed 24.3.2019. Source: Carnegie Mellon University. https://arxiv.org/pdf/1806.02371.pdf

Bolshev, A., Larsen, J:, Krotofill, M., Wightman, J. 2016. *A Rising Tide: Design exploits in Indstrial Control Systems*. USENIX . Accessed 3.3.2016. Source: USENIX https://www.usenix.org/system/files/conference/woot16/woot16-paper-bolshev.pdf

Brandes, U., Dlling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., Wagner, D. 2008. *On Modularity Clustering.* Department of Computer and Information Science, University of Konstanz. Accessed 4.2.2018 Source: CiteSeer http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.6623&rep=rep1&type=pdf

Buldas, A., Kroonmaa, A., Laanoja, R. 2013. *Keyless Signatures' Infrastructure: How to build Global Distributed Hash-trees*. Accessed 3.3.2016 Source: International Association of Cryptologic Research. Accessed 8.2.2017. Source https://eprint.iacr.org/2013/834.pdf

Böttcher, S. 2002. *Concurrent Checking of Global Cross-database Integrity Constraints in Integrity and Internal Control* (In Information Systems V: IFIP TC11 / WQ11). 5th Working Conference on Inegrity and Internal Control in Information Systems (IICIS), November 11-12 2002, Bonn Germany.

Chen, J., Zhu, J., Song, L. 2017. *Stochastic Training of Graph Convolutional Networks with Variance Reduction*. Accessed 11.3.2018. Source: Cornell University Library https://arxiv.org/pdf/1710.10568.pdf

Dai H., Dai, B., Song, L. 2016. *Learning Combinatorial Optimization Algorithms over Graphs. ICML 2016.* Accessed 18.4.2019. Source: Carnegie Mellon University https://arxiv.org/pdf/1603.05629.pdf

Dai H., Li, H., Tian T., Huang, X., Wang, L., Zhu, J., Song, L. 2018. *Adversarial Attack on Graph Structured Data*. Accessed 24.3.2019. Source: Carnegie Mellon University. https://arxiv.org/pdf/1806.02371.pdf

Das, Sajal K., Kant, K., Zhang, N. 2012. *Handbook of Securing Cyber-physical Critical Infrastructure*. United States of America: Elsevier.

Devenaud, D., Bettencourt, J., Chen, R., Rubanova, Y. 2018. *Neural Ordinary Differential Equations*. University of Toronto, Vector Institute. Toronto, Canada. Accessed 14.12.2018. Source: Cornell University Library https://arxiv.org/pdf/1806.07366.pdf

Dong, Y.,Johnson, R.A.,Xu, J., Chawla, N. 2017. *Structural Diversity and Homophily: A Study Across More ThanOne Hundred Big Networks*. Research paper, KDD'17, August 13–17, 2017, Halifax, Canada. Accessed 5.1.2018. Source https://ericdongyx.github.io/papers/KDD17-dong-johnson-xu-chawla-structural-diversity.pdf

Ferguson, N., Schneier, B. 2003. *Practical Cryptography*. 5th ed. Indianapolis, Indiana, United States of America: Wiley Publishing Inc.

Gehring, J; Miao, Y., Metze, F., Weibel, A. 2013. *Extracting deep bottleneck features using stacked auto-encoders*. Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference. Accessed 11.3.2018 Source: Carnegie Mellon University http://repository.cmu.edu/cgi/viewcontent.cgi?article=1097&context=lti

Getoor, L. 2019. *CiteSeer for document classification*. LINQS Statistical Relational Learning Group. University Of California Santa Cruz (UCSC), Santa Cruz, United States. Accessed 30.3.2019. Source: https://linqs.soe.ucsc.edu/data

Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Bing, X., Warde-Ferley, D., Ozair, S., Courville, A., Bengio, Y. 2014. *Generative Adversarial Nets.* Neural Information Processing Systems Conference (NIPS) Proceedings 2014, Accessed 8.2.2017 source: https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

Griswold, K. 2015. *Applying Partial Learning to Convolutional Neural Networks*. Accessed 11.3.2018 Source: Stanford University http://cs231n.stanford.edu/reports/2015/pdfs/kggriswo_FinalReport.pdf

Haller, S., Karnouskos, S., Schroth, C. 2008. *The Internet of Things in an Enterprise Context.* Volume 5468 Lecture Notes in Computer Science, Future Internet – FIS 2008, Springer

Hamilton, L.W., Ying, R., Leskovec, J. 2017. *Representation Learning on Graphs: Methods and Applications.* Data Engineering Bulletin, September: Graph Data Processing. IEEE Technical Commottee on Data Enfineering. Accessed 24.4.2018 source: http://sites.computer.org/debull/A17sept/p52.pdf

Hensman, P., Masko, D. 2015. *The Impact of Imbalanced Training Data for Convolutional Neural Networks*. Degree project, in computer science, first level. KTH Royal Institute of Technology (KTH), CSC School. Stockholm, Sweden. Accessed 12.4.2019. Source: https://www.kth.se/social/files/588617ebf2765401cfcc478c/PHensmanDMasko_dkand15.pdf

Hong, J., Liu, C., Govindarasu, M. 2014. *Detection of Cyber Intrusions Using Network-based Multicast Messages for Substation Automation*. Innovative Smart Grid Technologies Conference (ISGT), 2014, IEEE PES

Irvine, C. E., Levin, T. E. 2002. *A Cautionary Note Regarding the Data Integrity Capacity of Certain Secure Systems in Integrity* (Collection of United States Defence Technical Information Center). Naval Post-graduate School Monterey, Department of Computer Science, United States of America

Jonker, J., Pennink, B. 2010. *The Essence of Research Methodology: A Concise Guide to Master and PhD Student in Management Science*. Berlin: Springer Science and Business Media.

Ketkar, N. 2017. *Deep Learning with Python—A Hands-on Introduction.* Bangalore: Apress. Accessed 28.1.2018 Source: Books 24x7 http://library.books24x7.com.ezproxy.jamk.fi:2048/toc.aspx?bookid=128152

Khan, E. R., Anwar, H. 2015. *Research Methods for Computer Science*. India: Laxmi Publications. Accessed 10.3.2017 Source: Books 24x7 http://common.books24x7.com.ezproxy.jamk.fi:2048/toc.aspx?bookid=78408

Kipf, T. N., Welling, M. 2016 *Variational Graph Auto-Encoders,* NIPS Workshop on Bayesian Deep Learning (2016). Accessed 28.1.2018. Source: Cornell University Library https://arxiv.org/pdf/1611.07308.pdf

Kipf, T.N., Welling, M. 2017. *Semi-supervised classification with Graph Convoutional Networks*, ICLR conference, 2017, accessed 28.1.2018 source: Cornell University Library https://arxiv.org/pdf/1609.02907.pdf

Kipf, T.N., Schlichtkrull, M., Welling, M., van den Berg, R., Titov, I., Bloem, P. 2017. *Modeling Relational Data with Graph Convolutional Networks*, accessed 28.1.2018. Source: Cornell University Library https://arxiv.org/pdf/1703.06103.pdf

Kingma D., Welling, M. 2013. *Auto-Encoding Variational Bayes.* (published In Proceedings of the International Conference on Learning Representations (ICLR) 2014)

Langner, R. 2011. *Stuxnet: Dissecting a Cyberwarfare Weapon* (In IEEE Security & Privacy, vol. 9 issue 3). IEEE Journals and Magazines

Masters, T. 1993. *Practical Neural Network Recipes in C++*. Boston: Academic Press, Inc. Source: Google Books https://books.google.fi/books?id=Bn2jBQAAQBAJ&lpg=PP1&hl=fi&pg=PR3#v=onepage&q&f=false

McLaughlin, E. S. 2011. *On Dynamic Malware Payloads Aimed at Programmable Logic Controllers*. Proceedings of the 6th USENIX Conference. Source: Researchgate https://www.researchgate.net/publication/262355936_On_dynamic_malware_payloads_aimed_at_programmable_logic_controllers

National Instutute of Standards and Technology 2006. *Minimum Security Requirements for Federal Informatin and Information Systems* (FIPS-200). United States of America: Federal Information Processing Standards Publication.

National Computer Security Center 1993. *A Guide to Understanding Covert Channel Analysis of Trusted Systems* (NCSC-TG-030). United States of America: National Computer Security Center.

Patterson, J., Gibson, A. 2017. *Deep Learning: A Practitioner's Approach*. 1st ed 2nd release. Sebastopol: O'Reilly Media Inc. Accessed 5.12.2018. Source Google Scholar: https://books.google.fi/books?id=rLcuDwAAQBAJ&printsec=frontcover&hl=fi&source=gbs_ge_summary_r&cad=0

Phi, V. T. 2018. *Learning to Make Predictions on Graphs with Autoencoders*. Strategic Innovation Group, Booz, Allen, Hamilton. San Diego, Unite States of America. Accessed 25.12.2018. Source: Cornell University https://arxiv.org/pdf/1802.08352

Qin, Y., Mitra, N., Wonka, P. 2018. *Do GAN Loss Functions Really Matter?* Cornell University, arXiv.org, Computer Science, Vision and Pattern Recognition. Accessed 28.12.2018. Source: https://arxiv.org/pdf/1802.08352

Rajkumar, T., Bardina, J. 2003. *Training Data Requirement for a Neural Network to Predict Aerodynamic Coefficients* . SPIE Proceedings *Proc. SPIE* 5102, Independent Component Analyses, Wavelets, and Neural Networks.

Raschka, S. 2016. *Python Machine Learning*. 2nd ed. Birmingham: Packt Publishing

Silva, T.C., Zhao, L. 2012: *Semi-Supervised Learning Guided by the Modularity Measure in Complex Networks*. February 2012 Neurocomputing 78(1):30-37, DBLP

Spezzano, G., Vinci, A. 2015. *Pattern Detetctionin Cyber-Physical Systems*. (In Procedia Computer Science (52) 2015 1016-1021), Elsevier

Stajano, F., Anderson, R. 2002. *The Resurrecting Duckling: Security Issues for Ubiqutous Computing* (In Computer, vol. 35). IEEE Journals and Magazines

Sutton, R.S., Barto, A.G. 2018. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, Accessed 19.4.2019. Source: https://books.google.fi/books?id=sWV0DwAAQBAJ

Tipton, H. F., Krause, M. 2007. *Information Security Management Handbook volume 1*. 6th ed. United States of America: Auerbach Publications.

Tengfei,M., Chen, J., Xiao, C. 2017. *FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling*. Accessed 11.3.2018 source: Cornell University Library https://arxiv.org/pdf/1801.10247.pdf

Tsai, Y., Wu, D., Salakhutdinov, R., Yamada, M., Takeuchi, I., Fukumizu, K.2018. *Selecting the best in GANs family: A post selection inference framework*. ICLR 2018

Seventh International Conference on Learning Representations. New Orleans, USA. Accessed 30.12.2018. Source https://arxiv.org/pdf/1802.05411.pdf

Vallim Maffei, R., Carvalho, A., Gama, J. 2012. *A density-based Clustering Approach for Behavior Change Detection in Data Streams.* 2012 Brasilian Symposium of Neural Networks

Yang, Z., Cohen, W.W., Salakhutdinov, R. 2016. *Revisiting Semi-Supervised Learning with Graph Embeddings*. ICML 2016. Accessed 21.3.2018, source: Cornell University Library https://arxiv.org/pdf/1603.08861.pdf

Zhang, K. Liu, Q., Wu, Y. , Yang M. H. 2016. *Robust Visual Tracking via Convolutional Networks Without Training*. IEEE Transactions on Image Processing, vol. 25, no. 4. IEEE Explore Digital Library.

Appendices

Appendix 1. Datasets and Software Libraries

The files have been marked with the following extensions to highlight the role of a data splits according to Yang's Planetoid research (Yang 2016).

Table 7 Datasets

	Dataset Splits
.x	Feature vectors of labelled training instances
.y	One-hot labels of the labelled training instances
.allx	The feature vectors of both labelled and unlabelled training instances
.graph	Python dict in form of {index: [index of neighbouring nodes]}
.tx	Feature vectors of test instances
.ty	One-hot labels of the test instances
.index	Index set of test instances in a graph
.ally	Labels for instances in .allx

Cora and CiteSeer datasets are a collection of scientific papers with known citation relationships. Cora papers have been divided into seven classes and classes are Case_Based, Genetic_Algorithms, Neural_Networks, Probabilistic_Methods, Reinforcement_Learning, Rule_Learning and Theory. CiteSeer papers have been divided into six classes and classes are Agents, AI, DB, IR, ML and HCI. Both datasets include bag-of-words, set of words that were features of the papers. A scientific

paper is node in the graph and relation from cited paper to one or many citing papers. Data splits are stored as pickled objects.

The programming was done with Python and several machine learning software libraries were used. The classification and prediction libraries had been developed mainly using Google TensorFlow (Kipf 2017 ; Kipf 2018) and PyTorch (Dai 2018).

Table 8 Main software Libraries

	Main software Libraries
Python 2	Python version 2.7
Python 3	Python version 3.6
TensorFlow	Google TensorFlow, version 0.12 and 1.0
PyTorch	Machine learning library Torch for Python, version 0.3.1 wihtout Cuda
SciPy	Machine learning Python library, latest version from upstream
NumPy	Machine learning Python library, version 1.13.1
Networx	Python library for managing graphs, latest version from upstream
Matplotlib	Python library for visualization, latest version from upstream
Scikit-learn	Machine learning Python library, latest version from upstream
Pandas	Machine learning Python library, latest version from upstream
Docker	Docker version 18.09.2
Docker Desktop	Docker Desktop for MacOS, version 2.0.0.3
Ubuntu	Ubuntu Linux operating system, version 16.04

Appendix 2. Laboratory set up

The laboratory was set up as Docker containers that were running on top of physical host computer. Machine learning relied on processor (cpu) instead of graphics card (gpu).

Table 9 Docker images and physical host

	Description
my-kipf	Testing data engineering work results for good and bad datasets. This docker container is based on dockerfile "gcn_dockerfile".
autoencoder	Classification and prediction using good and bad datasets. This docker container is based on dockerfile "autoencoder_dockerfile"
reinforcement	Creation of bad dataset. This docker container is based on dockerfile "reinforcement_dockerfile"
Jupyter Notebook	Programming and execution environment
MacBook Pro	MacBook Pro laptop where Docker containers were run using Docker and Docker Desktop with Intel integrated graphics card.

Dockerfile "gcn_dockerfile":

```
# Base python 2.7 build, inspired by
# https://github.com/crosbymichael/python-docker/blob/master/Dockerfile
FROM ubuntu:16.04
MAINTAINER Tuuli Siiskonen <k2394@student.jamk.fi >
ENV LANG C.UTF-8
```

```
RUN apt-get update \
    && apt-get upgrade -y \
    && apt-get install -y \
    build-essential \
    ca-certificates \
    gcc \
    git \
    libpq-dev \
    make \
    python-pip \
    python2.7 \
    python2.7-dev \
    curl \
    ca-certificates \
    gcc \
    git \
    make \
    ssh \
    && apt-get autoremove \
    && apt-get clean
RUN ln -s $(which ${PYTHON}) /usr/local/bin/python
#RUN pip install --upgrade pip
# Install pip
```

```
RUN curl -O https://bootstrap.pypa.io/get-pip.py && \
    python get-pip.py && \
    rm get-pip.py
RUN pip install -U "setuptools==3.4.1"
RUN pip install -U "Mercurial==2.9.1"
RUN pip install -U "virtualenv==1.11.4"
RUN pip install numpy==1.13.1
RUN pip install scipy
RUN pip install pandas
RUN pip install -U scikit-learn
RUN pip install tensorflow==0.12
RUN pip install matplotlib
RUN pip install networkx
RUN pip install pickle-mixin
#Install Kipf's GCN code
# @in proceedings{kipf2017semi,
# title={Semi-Supervised Classification with Graph Convolutional Networks},
 # author={Kipf, Thomas N. and Welling, Max},
 # book title={International Conference on Learning Representations (ICLR)},
 # year={2017}}
RUN git clone https://github.com/tkipf/gcn.git && \
    mkdir /gcn/tmp && \
    cd gcn && \
```

```
    python setup.py install

# Store data in this mounted directory

#VOLUME /gcn/data

#ENTRYPOINT ["/usr/bin/python2.7"]

CMD ["/bin/bash"]

EXPOSE 8888
```

Dockerfile “autoencoder_dockerfile”:

```
Base python 2.7 build, inspired by

# https://github.com/crosbymichael/python-docker/blob/master/Dockerfile

FROM ubuntu:16.04

MAINTAINER Tuuli Siiskonen <k2394@student.jamk.fi>

ENV LANG C.UTF-8

RUN apt-get update \

    && apt-get upgrade -y \

    && apt-get install -y \

    build-essential \

    ca-certificates \

    gcc \

    git \

    libpq-dev \

    make \

    python-pip \

    python2.7 \
```

```
    python2.7-dev \
    curl \
    ca-certificates \
    gcc \
    git \
    make \
    ssh \
    && apt-get autoremove \
    && apt-get clean
RUN ln -s $(which ${PYTHON}) /usr/local/bin/python
#RUN pip install --upgrade pip
# Install pip
RUN curl -O https://bootstrap.pypa.io/get-pip.py && \
    python get-pip.py && \
    rm get-pip.py
RUN pip install -U "setuptools==3.4.1"
RUN pip install -U "Mercurial==2.9.1"
RUN pip install -U "virtualenv==1.11.4"
RUN pip install scipy
RUN pip install pandas
RUN pip install numpy==1.13.1
RUN pip install -U scikit-learn
RUN pip install tensorflow==1.0
```

```
RUN pip install matplotlib

RUN pip install networkx

RUN pip install pickle-mixin

#Install Kipf's Generative Autoencoder code

#@article{kipf2016variational,

 #title={Variational Graph Auto-Encoders},

 #author={Kipf, Thomas N and Welling, Max},

 #journal={NIPS Workshop on Bayesian Deep Learning},

 #year={2016}}

RUN git clone https://github.com/tkipf/gae.git && \

   mkdir /gae/tmp && \

   cd gae && \

   python setup.py install

# Store data in this mounted directory

#VOLUME /gae/data

#ENTRYPOINT ["/usr/bin/python2.7"]

CMD ["/bin/bash"]

EXPOSE 8888
```

Dockerfile “reinforcement_dockerfile”:

```
# Base python 2.7 build, inspired by

# https://github.com/crosbymichael/python-docker/blob/master/Dockerfile

FROM ubuntu:16.04

MAINTAINER Tuuli Siiskonen <k2394@student.jamk.fi>
```

```
ENV LANG C.UTF-8

RUN apt-get update \
    && apt-get upgrade -y \
    && apt-get install -y \
    build-essential \
    ca-certificates \
    gcc \
    git \
    libpq-dev \
    make \
    python-pip \
    python2.7 \
    python2.7-dev \
    curl \
    ca-certificates \
    gcc \
    git \
    make \
    ssh \
    unzip\
    && apt-get autoremove \
    && apt-get clean
RUN ln -s $(which ${PYTHON}) /usr/local/bin/python
```

```
#RUN pip install --upgrade pip

# Install pip

RUN curl -O https://bootstrap.pypa.io/get-pip.py && \

   python get-pip.py && \

   rm get-pip.py

RUN pip install -U "setuptools==3.4.1"

RUN pip install -U "Mercurial==2.9.1"

RUN pip install -U "virtualenv==1.11.4"

RUN pip install numpy==1.13.1

RUN pip install scipy

RUN pip install pandas

RUN pip install -U scikit-learn

#Be sure to get cpu version of PyTorch so that model load works! Use only v.0.3.1-
cp27-cp27mu

RUN pip install http://download.pytorch.org/whl/cpu/torch-0.3.1-cp27-cp27mu-
linux_x86_64.whl

RUN pip install matplotlib

RUN pip install networkx

RUN pip install pickle-mixin

RUN pip install tqdm

RUN pip install cffi==1.11.2

#Install Dai's code

RUN git clone https://github.com/Hanjun-Dai/graph_adversarial_attack && \

   cd /graph_adversarial_attack && \
```

```
   git clone https://github.com/Hanjun-Dai/pytorch_structure2vec.git && \
   wget -O dropbox.zip
https://www.dropbox.com/sh/mu8odkd36x54rl3/AABg8ABiMqwcMEM5qKIY97nla?d
l=0 && \
   unzip dropbox.zip -x / -d dropbox && \
   rm -rf dropbox.zip && \
   cd
# Store data in this mounted directory
VOLUME /graph_adversarial_attack/dropbox
# NO CUDA NEEDED! This recursively added structure2vec project's libraries can be
compiled with cpu based compiler
RUN cd /graph_adversarial_attack/pytorch_structure2vec/s2v_lib && \
   make
#ENTRYPOINT ["/usr/bin/python2.7"]
CMD ["/bin/bash"]
EXPOSE 8888
```

Software code (Dai 2018) was changed so that attack scenarios could be run using CPU instead of GPU. Implementation provided number of pre-calculated models that can be applied to attack code. PyTorch library used supported GPU parallel computing that relied on NVIDIA drivers. Models were loaded using PyTorch (Dai 2018). In order to move the computations to CPU in an environment where there were no NVIDIA graphics card or drivers working, GPU version of PyTorch 0.3.1 was replaced by CPU version of PyTorch 0.3.1 library. Additionally, Makefile and build.py files were modified so that NVCC GPU compiler was replaced with CPU based compiler. Additionally, PyTorch load function was instructed to move the computation in CPU by setting map_location parameter to follow the ctx command

line argument value. While this ctx command line argument reads cpu, computations should take place cpu based. For this, files node_attack_common.py and node_dqn.py were modified.