INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Dezentrale Datenverteilung mit TPM-basierter Authentifizierung

Sebastian Rehms

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller
Betreuer:	Dr. Nils gentschen-Felde Tobias Guggemos, M.Sc Stefan Tatschner, M.Sc. (Fraunhofer AISEC) Norbert Wiedermann, M.Sc. (Fraunhofer AISEC)
Abgabetermin:	22. Januar 2018

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Dezentrale Datenverteilung mit TPM-basierter Authentifizierung

Sebastian Rehms

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller
Betreuer:	Dr. Nils gentschen-Felde Tobias Guggemos Stefan Tatschner, M.Sc. (Fraunhofer AISEC) Norbert Wiedermann, M.Sc. (Fraunhofer AISEC)
Abgabetermin:	22. Januar 2018

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 22. Januar 2018

. . . . . .
(Unterschrift des Kandidaten)

Abstract

By utilizing the potentials of an interconnected world, a transformation of the industry has been initiated. Changes issued by the implementation of an Industry 4.0 have impact on the current production landscape. New requirements for infrastructure, communication and data exchange are identified. Essential for increasing production efficiency towards more autonomy is the implementation of these requirements in a secure way. In this interconnected world reliable data distribution becomes a central factor for secure and economic Industry 4.0. In this thesis three models are presented to analyze data management and distribution: Information Groups, Logical Connections and Data Flows. Informational Groups adapt aspects of the Reference Architectural Model for Industry 4.0 (RAMI 4.0) to propose recursive management shells as an easy way for transparent data management across companies and production sites. This work presents a concept along with a proof of concept implementation for a decentralized approach for data distribution to improve availability. By reusing already available basic building blocks, the Block Exchange Protocol (BEP) is adapted to this industrial use case. For further hardening of the underlying cryptographic primitives, Trusted Platform Modules (TPMs) are included as secure key storage as well as to generate cryptographic keys. As part of this thesis, the TPM connection is realized by adapting interfaces and libraries to enable applicability in the context of BEP. This hardening prevents the compromise of private keys. Recent publications such as Heartbleed, Meltdown, or Spectre emphasize the necessity to secure sensitive information like private keys.

Zusammenfassung

Durch die Potentiale, die die fortschreitende Vernetzung der Welt mit sich bringt, ist eine Transformation der Industrie angestoßen worden. Die Realisierung von Industrie 4.0 bringt Änderungen in der gegenwärtigen Produktionslandschaft mit sich. An Infrastruktur, Kommunikation und Datenaustausch werden neue Anforderungen gestellt. Um mehr Produktionseffizienz durch Autonomie zu erzielen, ist es essenziell, diesen Anforerungen auch im Hinblick auf Sicherheit zu genügen. Die verlässliche Verteilung von Daten in dieser vernetzten Welt spielt eine zentrale Rolle für eine sichere und ökonomische Industrie 4.0. Die vorliegende Arbeit präsentiert drei Modelle, um Management und Verteilung von Daten zu analysieren: Informationsgruppen, logische Verbindungen und Datenflüsse. Das Informationsgruppenmodell adaptiert Aspekte des Reference Architecture Model for Industry 4.0 (RAMI 4.0), um rekursive Managementschalen vorzuschlagen, welche eine einfache Möglichkeit bieten, transparentes Datenmanagement über Standort- und Unternehmensgrenzen hinweg zu realisieren. Diese Arbeit stellt ein Konzept für einen dezentralen Ansatz der Datenverteilung vor und liefert einen Machbarkeitsbeweis durch Implementierung. Die Dezntentralität sichert die Verfügbarkeit von Daten. Indem existierende Komponenten verwendet werden, wird das Block Exchange Procol (BEP) auf den industriellen Anwendungsfall hin adaptiert. Um die tieferliegenden kryptografischen Primitiven sicherer zu gestalten, werden Trusted Platform Modules (TPMs) sowohl als sicherer Speicher als auch als Generator für kryptografische Schlüssel eingesetzt. Die Arbeit zeigt den Einsatz von TPMs im Kontext des BEPs durch die Anbindung von Interfaces und Bibliotheken. Diese Härtung der Sicherheit verhindert die Kompromittierung von privaten Schlüsseln. Jüngste Publikationen wie Heartbleed, Meltdown und Spectre zeigen die Dringlichkeit, sensible Informationen wie private Schlüssel speziell zu sichern.

Contents

1. Introduction

The combination of computing and communication technologies leads to new ways in the economy. Secure data sharing between multiple agents is an important factor for benefiting from the new possibilities. This thesis aims to analyze requirements for the secure distribution of information in company scenarios and proposes an adequate concept.

1.1. Motivation

With the term *Industry 4.0* one refers to a predicted fourth industrial revolution. It is driven by the possibilities that new technological developments open. Communication between people, machines, and resources has become easy and the interleaving along value chains is a main property of this upcoming future. There is a paradigm shift from centralized controls to decentralized production processes. As the term Industry 4.0 refers to a process that has not yet terminated - in contrast to the other industry-changing processes which were described in retrospec - it is possible to shape the upcoming landscape so that the needs and challenges for the industry are met from the outset and the result is suitable [20].

Data distribution is a central point in Industry 4.0. For reliable and secure data distribution in industry, IT-protection goals should be taken into account. In an environment where lots of participants are interconnected, which is one of the driving ideas of Industry 4.0, it is crucial to secure the communication between members, especially in cross-company scenarios. Classical security goals are Confidentiality, Integrity and Availability (CIA). With lots of business-intern entities connected to each other and some of them having connections to the outside world, being able to tell if a communication partner is who it claims to be, is as well important [3]. Thus in an Industry 4.0 setup this fourth goal is also a relevant factor, i.e. Authenticity.

1.2. Problem Statement

Loss of production poses a big threat to companies. Self-organization is a declared goal in the upcoming world and interruption should only occur in special cases. As this upcoming terrain of self-organization and cross-border information flow heavily relies on data, one has to ensure that data gets distributed reliable. Automatizing the distribution process is thus an implicit objective of Industry 4.0 as the whole point is to profit from the new technological possibilities and become more efficient. Distributing information has never been easier than today. This lies mainly in the possible interconnectedness of distributed systems.

Availability plays a major role in the design concepts of Industry 4.0. The above mentioned paradigm change to decentralized production processes can also be applied to data distribution, because decentralized structures are much more robust to small or even medium scaled technical disruptions. As information can easily be shared between partners one can take advantage of the interconnectedness and build a dependable system in which the failure of

single points does not pose disruptive problems to the information flow of the system. If relevant data is shared between all agents, one can acquire the data by every agent, not only a central one.

Under the new conditions the information flows are much more interconnected than today. Currently, data belonging to a machine is of interest only by the machine itself and the maintainer of the machine. For self-organizing structures it is required, that data is shared between all participants to whom the data may be relevant. Data flow across company sites or even cross-companies is a wishful property of upcoming setups, thus enabling monitoring and configuration along value chains. Not every bit of data is useful to everybody and, in particular, information may be classified and should only be accessible by a certain group of agents. This means that one has to ensure the Confidentiality of data, i.e. enable access only for those roles which are allowed. Authenticity, Confidentiality and Integrity can all be achieved by cryptographic means. However, there is need for a root of trust. By providing essential cryptographic capabilities and storage facilities to all involved partners of a network, one can make the network harder to penetrate and limit the damage of a security breach. These capabilities can be provided by Hardware Security Modules (HSMs), which ensure secure storage of keys that are never directly accessible. Hence, the security of a machine and of the whole network is strongly supported by hardware.

1.3. Structure of the Thesis

The goal of this thesis is to analyze basic requirements of a decentralized data distribution infrastructure, which meet the needs of Industry 4.0. A concept is built, which is then examined through an implementation for an exhibit to analyze the feasibility. In Chapter 2 the requirements are identified. Beforehand, three models are introduced in order to examine data distribution from different perspectives. In Chapter 3 technological approaches for decentralized data exchange are examined. Also, other techniques are surveyed which are relevant in the context of the results of the previous Section. Subsequently, in Chapter 4, the concept is drawn and an exhibit setup is presented, which tries to implement a subset of the concept. In Chapter 5 the findings of the thesis are discussed with help of the models from Chapter 2. The exhibit with view to the requirements is examined, as well as advantages and drawbacks and improvement possibilities. Lastly, in Chapter 6 a Conclusion together with proposals for future work is given.

2. Scenario and Requirements

It is not the task of this thesis to analyze the exact changes referring to Industry 4.0. However, to find requirements and to build a concept for decentralized and secure data distribution, at least the relevant changes and needs have to be examined.

The term Industry 4.0 lacks a concrete definition and thus it is hard to identify and design corresponding scenarios [30]. Initially, in Section 2.1, possible views will be introduced which can be taken for exploring data distribution. Secondly, in Section 2.2, changes in the industry will be discussed which are relevant in the context of this thesis. For this, a Scenario is set up to provide a link to Industry 4.0. Thereafter, in Section 2.3, the analysis will move to a more specific investigation with focus on secure data distribution and relevant related topics in order to infer requirements for the painted scenario. Lastly, in Section 2.4, the scope of the thesis is defined, to delimit the area it is concerned with.

2.1. Views on the Architecture for Data Distribution

When concerned with data distribution between several agents, one can take different views which put the focus on various issues. In this Section, three models are proposed, which will be referred to throughout the thesis.

For distributing data it can be of big concern to whom it is distributed, especially for enterprises, as it will become clear in Section 2.3. Information is shared between groups of agents. An agent might be any entity: a production facility, a personal computer but also bigger and more abstract entities like other companies. An agent might have the role of a representative for a group. Depending on the taken view, such a representative has different tasks.

Informational Groups. Figure 2.1 shows a view on data distribution with focus on *groups* with a bipartite hierarchical structure. It shows an exemplary configuration. Groups consist of *nodes*. For ever group there is a *manager node* which represents the group to the outside. Theses managers are decision makers for which information is distributed to other groups. Hence, there are communication channels between the representatives of the groups. The internal channels of the groups are not relevant for this informational group view as the focus is on the separation and classification of information. The question of the exact internal structure of the groups is of concern with the other two view models. The external channels accentuate that distribution of information is controlled. More details on this topic is investigated in Section 2.3.1.

Taking this viewpoint on data distribution, one is concerned with managing the information which is represented by data. Manager nodes are proposed as abstract entities for supervising groups. Thus, there is no need to restrict the groups to only one manager node. They are representatives of an informational group following a group policy.

Figure 2.1.: **View of Informational Groups for Data Distribution**

Logical Connections. Figure 2.2 takes the view of how entities are connected, i.e. which *peers* can directly talk to each other.

All peers are organized in Communication Groups. Logical connections between peers indicate that they can establish a connection to each other, e.g., via a Virtual Private Network (VPN) or Ethernet. Peers of a given Communication Group can have the role of an *internal peer* or of an *external peer*. An internal peer has connections only to members of its group. An external peer has connections to other external or internal peers of its own group but also to external ones of other groups.

Information may be routed to other peers by peers which then act as relays. It is implied that for one peer of a given group A, it may be necessary to use peers of the same group A to connect to a distant peer. In Figure 2.2 the right group is structured in this manner. For any internal peer of a communication group, to connect to a peer of another group, the connection needs to be relayed through an external peer of the group.

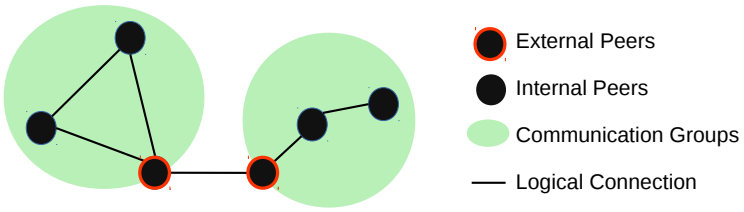

Figure 2.2.: **View on Logical Connections for Data Distribution**

Data Flow. Figure 2.3 represents a view on data distribution with focus on *data flow* and *communication processes*. All nodes are organized in Data Groups. A Master node distributes data to Slave nodes within a Data Group (Figure 2.3 a)). Also, a Master node can accumulate data form Slave nodes of a Data Group (Figure 2.3 b)). In the course of this Chapter it will become more clear why these two are relevant in this thesis.

A Master node has an authoritative function in the group. In Figure 2.3 a) a Master distributes data, e.g., configuration data, within a group. It is the root of a distribution tree for that group. In a second step a Slave can act as a second level Master for other Slaves and, in case, it is the new root of a subtree. The concept can be applied recursively. Thus, nodes can be made transparent to a Master on a higher level.

In Figure 2.3 b) a Master polls data, e.g., sensor data, from its Slaves. A Slave may beforehand act as a Master to its own Slaves, accumulating data, which are then passed on to the root of the tree.

For this view no statement has been made on pushing or pulling data. In case a) of Figure 2.3 a Master could simply push new configuration data to its Slaves; but the Slaves could also pull changes in configuration data from their Master in timed intervals. The same holds for data accumulation, in which the Master could periodically pull data from its Slaves; or the Slaves push changes in data to their Master.

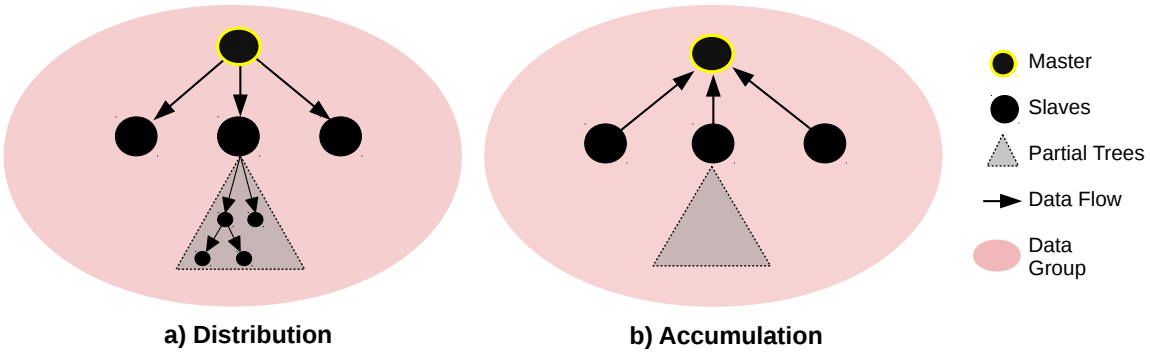

Figure 2.3.: **View on the Data Flow for Data Distribution**

All three views need to take failures into account. If a central point like a Master drops out, there needs to be a protocol to determine a new Master, e.g., by voting or using a metric like ping time. If the model allows for multiple centralized points as for the Informational Groups, redundancy reduces the risk of loss of Availability as it will be discussed in Section 2.3.2.

2.2. Changes with Industry 4.0

The main focus of Industry 4.0 is the establishment of intelligent production processes with a high amount of flexibility to build personalized products. This implies rapid product development and flexible production in complex environments [7]. Most of the technologies are not new. The sum and collaboration of these technologies are the decisive factor that leads to an evolution in the industry [30].[1]

One main driving technology for the fourth industrial revolution is the Internet, which allows for easy communication between entities. As well important is the fusion of physical systems and the virtual world resulting in integration of computational abilities in physical machines, namely Cyber Physical Systems (CPSs) [7]. The setups are build on a combination of Ubiquitous Computing (UC), which means the possibility of every object in a system to

[1] Using the term 'evolution' in contrast to 'revolution' respects the fact, that there is no new decisive technology which leads to perturbations in the industry.

handle and work on data, and the Internet of Things (IoT), which enables those objects to communicate with each other. This requires an unique identity for every object which is possible for example due to IPv6 or RFID tags [30].

The possibility of sharing data is the basis for joint collaboration and has becomes easier [42]. The depth of added-value within one factory and company decreases which leads to a need for collaborative development and manufacturing environments, especially for small and medium enterprises. This leads to collaborative networks. A global optimization of production processes relies heavily on the availability of product and production data across factories and company boundaries. Such networks are sometimes called virtual corporations [7]. Today this is not a ubiquitous phenomenon [59].

A more concrete example for a scenario is depicted in Figure 2.4. This example does not represent a dedicated Industry 4.0 company. It rather offers a picture in which Industry 4.0 takes place. A company *A* manufactures one or more types of products. It has several sites (Site 1 and Site 2) which act dependent on each other, for example as suppliers for components or information. There are also several other supplier companies (in this case company B). Company *A* may also act as a supplier to others (Company C) or be a final seller of their products.

The items are assembled along value chains and the company *A* plays an major role in this chain. Along the whole value chain compliance with demands like security standards is of great importance. Especially with the production processes becoming more digitalized the surface for attacks grows and the risk of production loss rises.

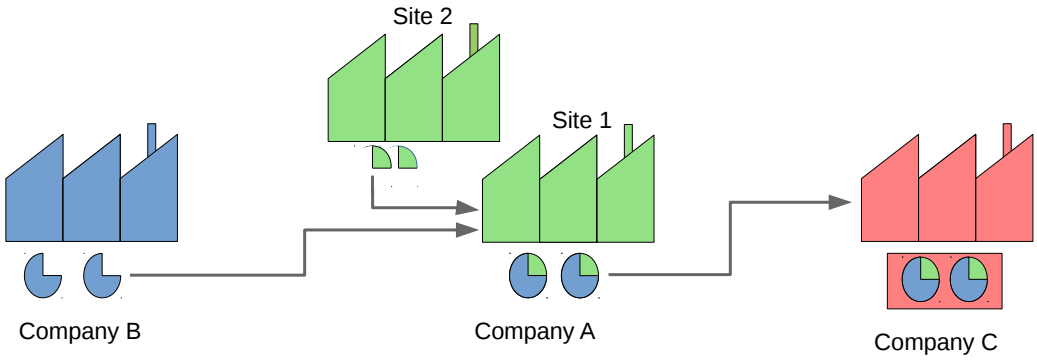

Figure 2.4.: **Organization along a value chain**

As Industry 4.0 aims to integrate digitalization to optimize the production along value chains using emerging technological possibilities like UC and the IoT. Sites of company *A* may gather live data and exchange them with each other in order to be more flexible and reactive to changes for example in production volume. Different sites want to know the status and progress of production. Data of external suppliers is acquired as well.

Accumulating such information offers opportunities for higher level processes to decide for certain changes in production schemes. These decisions can be automatized, which is an aim for Industry 4.0, i.e. autonomous acting systems. Acquired data may also be passed to other companies for which *A* acts as a supplier. This all implies a much more network dependent fabrication. In order to minimize the risk of digital failures, redundancy is as well important as IT-security. Countermeasures against newly detected cyber-threats have to be build as soon as possible to minimize the risk of loss of production. However, it is as well important

to protect against theft of knowledge which plays a even more important role to survive on the market.

A company can be logically modeled as a hierarchical structured pyramid, the so called automatization pyramid. Figure 2.5 shows the information flow in a pyramid model which is derived from the IEC62264 norm and also used in the Reference Architecture Model for Industrie 4.0 (RAMI 4.0), a reference model for Industry 4.0 [60]. In the old model the structure is given by hardware connected mainly via Local Area Network (LAN). Data flows linear from one hierarchy level to the next one. Information of one factory is gathered at Enterprise level and, if needed, forwarded to another site of the same company or an external supplier, where it is again passed on in a linear way.

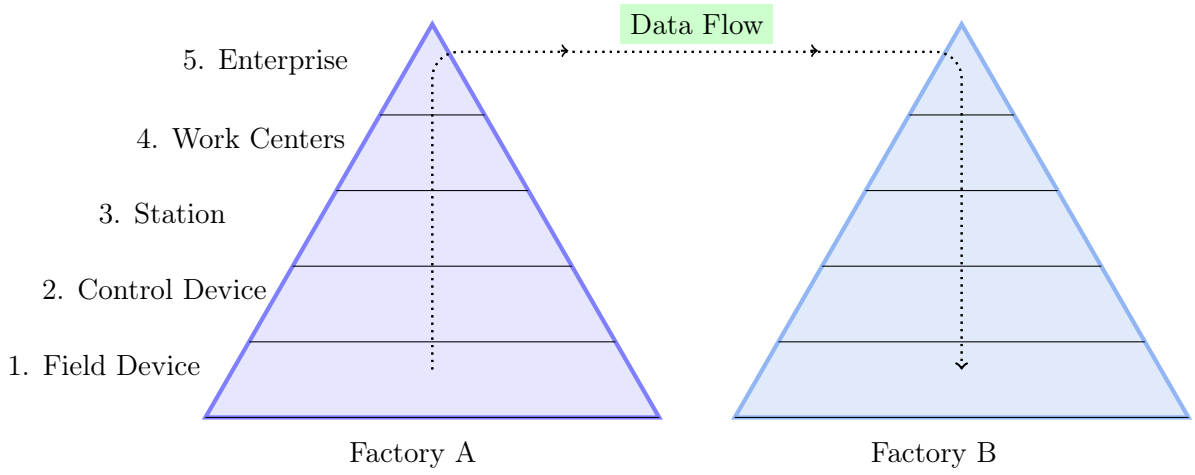

Figure 2.5.: **Data flow between different sites (old world)**

In the RAMI 4.0 a breakup of the hierarchy levels is proposed, leading to a strongly connected network, in which each entity talks to many other entities. Also products should be part of this network so they might advertise their status to all partners [60].

This model abstracts from some conditions which should be considered if one wants to derive requirements for a Industry 4.0 scenario. The changes bring new possibilities for information flow which can be represented on an abstract level by a restructuring of the automation pyramid as shown in figure 2.6. The shown Figure sticks to the pyramid model to represent the logical structuring of devices and processes in levels.

While in the old model the information flows linear through all stages of the pyramid in order to enable communication between different factories, in the new model the exchange of information can occur among all levels. In the Figure 2.6 an exemplary path is shown.

2.3. Requirements for Secure Data Distribution

Both, humans as well as machines need to know about the physical state of the environment in order to find reasonable decisions. The IoT together with other agents like real world persons and abstract entities is called the Internet of Everything (IoE). Getting knowledge of the world-state requires the aggregation of raw sensor data to higher-value context information. The

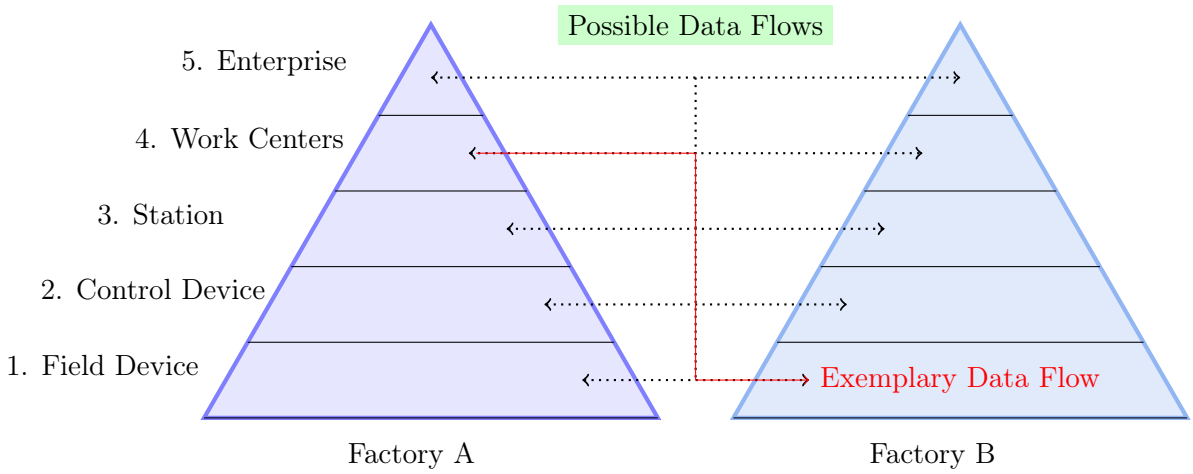

Figure 2.6.: **New possible data flow between different sites**

analytics results need to be available to several participants of the IoE, e.g., through assistance systems. Decentralized decisions rely on this so called transparency of information [42].

Another area, in which data distribution is of great relevance is IT-security. With the ever growing number of systems connected to the Internet, effective and prompt defense is essential for solid production. For such areas the reliable and prompt distribution of, e.g., signatures of malicious software is important in order to avoid downtimes and information theft [3].

In order to enable connectedness of different types of entities in a network, common communication standards are of great importance, especially in cases of modularization for flexible machine combinations from different vendors [42]. In the following Subsections requirements for data distribution are deviated. As stated, the picture of interconnected entities to a IoE simplifies some facts that are of need for a trustworthy industrial environment.

2.3.1. Group and Identity Management

The possibility of sharing data is the basis for joint collaboration and becomes with the IoE much more easy. Currently, data belonging to a machine is of direct interest only by the machine itself and the maintainer of the machine. For self-organizing structures, however, it is required that the data is shared between all participants to whom the data may be relevant. But the data should only be shared between those entities which it is intended for; information can be confidential and access needs to be restricted. One of the main obstacles for close collaboration between companies is trust [7] thus it is of great importance that one takes care of the information flows. For ensuring trust in collaboration, information needs to be shared responsible. Data management is required. Still, data flow across company sites or even cross-companies is a wishful property of upcoming setups, in order to enable monitoring and configuration along value chains.

In an Industry 4.0 scenario, company A accumulates data of sensors which are then shared between all sites. It is also available to higher levels of the organizational structure, independent of the location. Information, e.g., on production status and problems with

respect to Availability, are then passed to second parties for which A acts as a supplier.

On all levels processes also gather and integrate information from other external entities, for example suppliers, but also on market status or on the cyber-threat landscape. This information is also propagated to all participants of the network for whom it is intended; and it should only be forwarded to those agents for whom it is intended. There is also sensible data that should not be shared with second parties as for example intellectual property or other classified information like suppliers and incidents.

From a logical perspective this requires a management of Identities or agents which are subsumed into groups. All participants of a Infromational Group share data. The RAMI 4.0 proposes for every asset of a company to implement its own management shell, which acts as an interpreter and interface for an asset. Such assets may communicate with each other using their shells. This is comparable to an IoT gateway [20]. Figure 2.7 shows this concept. Every asset has its own management shell (blue) with a central interface for other entities to connect to (black dots). Using the central interfaces the entities can establish communication channels with each other (black lines).

Figure 2.7.: **Communication Between Entities via Management Shells**

These shells ensure a common communication basis. Additionally, a shell should hold all information for an asset. In practice such a shell resides in the digital capacities of a CPS. It is also possible to represent abstract entities like processes in the digital sphere which then appear as individual assets or agents represented by a Management Shell. The entities are assumed to be placed on any hierarchical level of the automatization pyramid as an interleaving is aspired.

In a next step such agents can form logical entities or groups. For example a group may represent all stations (the third level in Figure 2.5) or all control devices (the second level). Those grouped units are again represented and coordinated by management shells of higher levels, which can be again grouped and so on. Referring back to the different views on data distribution in Section 2.1 the view of Informational Groups is taken. A high level management shell corresponds to an Informational Group. On a low level a management node can have the role of a Master for aggregation. This recursive concept is depicted in in Figure 2.8. The Figure also embeds the second view of Logical Connections . This emphasizes the fact that there is no complete intermeshing in a group required.

Every black dot represents an asset or process. In a given shell a node may act as a regular one but have the role of a management node for a nested shell. It should be stressed that assets may also be part of different shells. One group X may be formed by all control devices

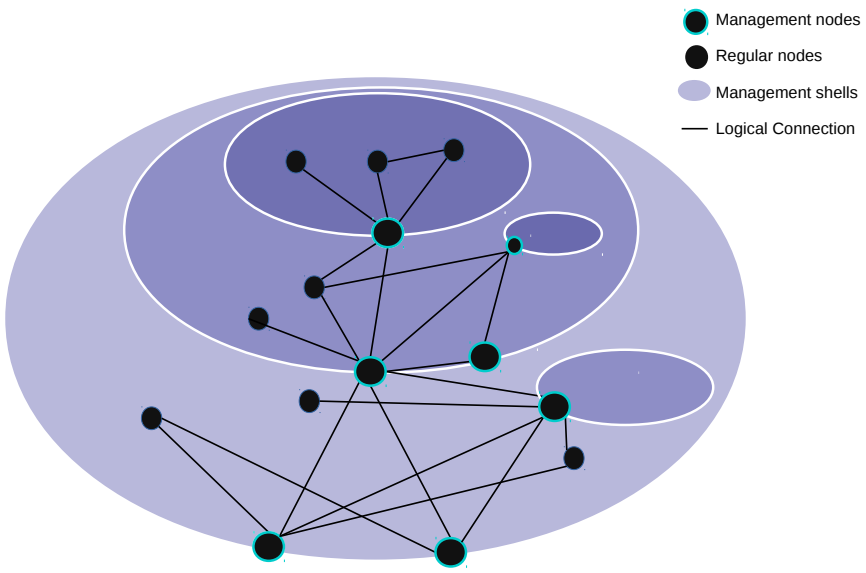

Figure 2.8.: **Recursive Management Shells**

of a company across all factories. Another entity on the network may only be interested in the devices of one particular site, so only these are part of this specific shell Y.

If a process p on the network asks for information from a given shell X it may not be interested in the information of every asset that is part of this shell. A higher level process which manages X's information is the interface for this process p to gather data. In the Figure those interfaces are marked by the colored circles around the agents. For an interface it is necessary to hold all relevant data of its group. Thus they act as managers for their cluster. For clarity groups of other managers are only indicated in the figure.

If a manager wants to hold all relevant data it has either to be connected to every member of its shell or information has to be forwarded to him through other nodes which then act as relays. Those interconnected agents may also act as a coordinator for the group. This architecture allows a recursive structure in which shells become to a great extent transparent to agents outside the shell, which means external nodes do not need to care of the inner structure of another shell.

On the other hand the inner structure is hidden for such external nodes which is relevant in terms of Confidentiality. In case that a part of the inner workings of a shell are of interest for an external entity there needs to be an appropriate Informational Group, i.e. a Shell. Mapped back to the pyramid model the hierarchical structures are given up in order to enable communication between all network participants. However, a logical hierarchical structure, as explained, needs to be implemented in order to enable application in an enterprise scenario for purposes of information transparency and management to ensure confidentiality. The most last point will be discussed more in depth among others in the next Subsection 2.3.2.

2.3.2. Security Objectives

The IoE is growing. With it being the basis of industry in the future, monetary and political interests will increase the number of attacks on Industry 4.0 production facilities and, therefore, increase the need for IT-security [42]. Real world incidents like the Mirai botnet have shown

that the IoT is a critical structure which holds much power meanwhile it is very easy to attack [13]. For the industry, to use the potentials of the IoT, it is urgent to address information security.

Objectives for this topic have been standardized by the norm ISO 27000 [37]. This norm is a standard for implementing Information Security Management Systems. The purpose of such systems is to assure that certain security objectives are appropriately addressed through processes like security incident handling or access rights management. The main three objectives ISO 27000 proposes are Confidentiality, Integrity and Availability, but there are others too, like Authenticity. In this Subsection security objectives are evaluated as requirements for this thesis.

Throughout the thesis security objectives are used with a capital letter at the beginning. The lowercase refers to the regular term, not the security objective.

Confidentiality is defined by the ISO 27000 as the "property that information is not made available or disclosed to unauthorized individuals, entities, or processes" [37]. Like pointed out, not every bit of data is useful to everybody and in particular information can be classified and should only be accessible by a certain groups of agents. For an industry scenario it is necessary to take this into account.

To ensure Confidentiality, access to sensitive data has to be prohibited. There are lots of situations that need to be considered like employees accidentally providing sensitive information by mail. For self managing structures this is not of direct concern. More importantly, automatized communication processes need to be properly managed to ensure Confidentiality of data. Data stays private if it is only disclosed to those entities it should be to. Applying the Group management model from above, access shells can be defined. Subsequently, the problem is condensed to the statement that for given information this information is only accessible by an the appropiate Informational Group.

Avoiding access to classified data is mainly realized by two mechanisms. The first one is to disable handling of the data. This can be realized by a group management infrastructure. If some agent requests access to information, a management process needs to check if the agent is part of the group that should have access to it. This means to create the group which actually holds the data or whose data is accumulated at a central point, which can be refered to as a Data Group; and another logical group for which the data is intended, which can be refered to as an Informational Group.

The other possibility is to allow handling of the data but disabling the extraction of the real useful information by means of cryptography. In practice this means that data is stored encrypted and only those agents who are part of the appropriate group have access to the according decryption keys. This implies of course that the cryptography can be trusted. One also has to ensure that data, which has been accessed by a process, is kept private afterwards. Additionally, if a memeber leaves a group, the revocation of the membership needs to be addressed, which is the task of an Identity management process.

Another important point to consider is the data transfer itself. Depending on the channel it can be easy for third parties to intercept the communication and thus the information if no precautions are taken. Hence, it can be necessary to set up a secure channel between the two peers. Encryption is an important method to uphold privacy. Some usable technology will be discussed in Chapter 3. If the data is encrypted before the transfer, this a minor problem, granted that there is no confidential data leaked through meta data. However,

this still implies the problem of Authenticity of the communication partners which will be examined as an extra point.

Authenticity is the "property that an entity is what it claims to be" [37]. In order to build on a reliable infrastructure for Group management it is necessary to integrate the assurance of Authenticity. A standard technique to gain unauthorized access to data is pretending to be someone authorized. This is the problem of impersonation. Therefore, Authenticity is closely related to Confidentiality.

However, avoiding direct disclosure is not the only point. With respect to data distribution it is possible that some malicious agent distributes faked or manipulated data, e.g., malformed signatures of malware which subsequently will not be detected. Thus, a data donor has to prove that it is the one it claims to be and that it is authenticated to distribute the data. In return, the first peer needs to know with whom it wants to communicate and optionally hold required information in order to check the Authenticity, like fingerprints of trusted peers. This will also be explained more closely in Chapter 3. For now it can be stated that Authenticity mechanisms are an important requirement for an Industry 4.0 scenario as it ensures reliability of such a system in not fully trusted environments as it is the case for ubiquitous connectedness and virtual corporations. It addresses two problems. The first one is the problem of avoiding impersonation. The second one is to proove the membership of a group.

Integrity. Trustable data is not only addressed by Authenticity but also by Integrity. Data may become corrupted or manipulated. To ensure data integrity the data must not be altered and also be complete. This can be realized by checksums, especially cryptographic ones as they are much harder to fake. To stay with the recent example of malicious signatures one has to ensure that the delivered signatures are accurate. If not, they are useless to detect malware. In fact such signatures are also just kind of checksums of malware.

In the context of data distribution Integrity also has the dimension of data consistence. If data is distributed in a group, the data on every single member should be consistent with the data of the other membmers.

Availability. Loss of production is a critical situation in industry as it is the core of this business. Availability is defined as the "property of being accessible and usable upon demand by an authorized entity" [37]. As explained in Section 2.3 the new industry relies on the Availability of data. For production facilities which are connected to each other it is required to evaluate the risks of this connectedness due to an increase of the attack surface. Other security objectives from above like Authenticity and Integrity are important to be considered in this course.

One has to ensure these in order to decrease the risk of compromise which can also lead to not accessible or unusable assets, which is a loss of Availability but also to a loss of privacy. A direct loss of Availability is also a risk and may occur with denial of service attacks. Ways for mitigating such risks is not part of this thesis.

It is important to think about the information assets a smart factory relies on. One needs to take into account situations in which required information is not at hand. If production must not lack certain data to avoid loss of production, there need to be redundant sources. This holds true no only in case of criminal activities but also with technical failures.

Generally, redundancy is a valuable possibility to raise Availability. Ubiquitous data sharing offers great opportunities for reducing the risk of unavailable data. Nevertheless, it should be accentuated that a more connected infrastructure increases the possibilities for attacks. However, this is the nature of the idea of Industry 4.0. Thus Availability is on the one hand increased through the interconnectedness but on the other hand at higher risk by a largely increased attack surface.

2.3.3. Decentrality

The classical approach for data distribution is a client-server architecture. One client stores information on a central server and another client pulls it from there. Even for the case that a client needs to talk directly to another client this is mediated through the central server. This poses big risks as the server is a single point of failure. The Availability of data is directly lost with a failure of the central server. It also scales bad with an increasing number of peers due to an increased amount of data throughput and a larger number of connections all running through the server [3]. For growing infrastructures more servers are required which need to be coordinated.

The client-server model is of advantage for application in environments where the clients have very low computational resources. In the context of data distribution Decentrality is a long proposed approach. It can ensure the Availability of information, even if single points drop out. The best known application of this principle is the Internet itself. With decentralized methods a central point is given up.

Some datum is hold by several peers in the network which ensures redundancy. The paradigm of decentrality is best realized in Peer-to-Peer (P2P) networks. In so called structured P2P networks, peers announce their status to their neighbors for self organization. If some resource is required, e.g., in file sharing, there needs to be a protocol which locates a provider and the peers may then directly exchange the data or send it via peers which forward it, acting as relays [3].

Thanks to the possibilities like self organizing design and features like direct communication a P2P approach greatly correlates with the ideas behind Industry 4.0. Yet a decentralized environment is only feasible if the peers have enough resources so that every agent can act as a client and a server at once. The shell conept from Section 2.3.1 allows for a hybrid version in which the management nodes accumulate data and then act on a higher level as peers in a decentralized network. This could also be a possible approach for scaling problems. In Chapter 5 a short discussion of these problems takes place.

One obstacle for a fully decentralized network in a company context is the bootstrapping problem: If an agent wants to joint the network it somehow has to find an entry point. This can be resolved by setting up bootstrapping servers. This erodes partly the decentrality paradigm though. Further, to build a trustable network these servers are also required to trust the new peer. For example the servers need to have a list of agents who are allowed on the network. Thus a management of identities is required. Likewise, somehow the peers need to know of each other whom they can trust, i.e., who is part of the trusted network, which is a problem of Authenticity and of Identity management. Again, it holds that centralized servers which act as "identity ledgers" can adopt this task. A distributed data base can also act as a identity ledger, which again poses the problem of bootstrapping.

The distributed paradigm for system and network management has become more relevant over the last decades in enterprise networks giving up the paradigm of centrality [40].

2.4. Scope

So far it can be concluded that a simplified model in which everything just is connected falls to short for needs of an industry scenario if one takes information security into account. In order to achieve a reliable setup lots of problems have to be considered of which each is not trivial to solve. It is beyond the scope of this thesis to inspect every part in detail. In Chapter 3 some technical approaches are presented and discussed that relate to the evaluated requirements. None of them alone claims to solve all problems. The mere task of the thesis is to show that there are several approaches which can be useful depending on the context. A second objective is to show the appliance of some of the technologies in order to realize a decentralized data distribution infrastructures that meets the needs of Industry 4.0. However, the complexities of this topic force to make several assumptions for the exhibit and the concept it builds on. The assumptions are:

1. There is an Identity management infrastructure and corresponding processes. This means that every peer has the knowledge about other peer's identity that it might need. This is mainly information about the public keys of the peers.

2. The peers are able to talk to each other over a network, i.e. they are able to address and establish end-to-end connections to each other. For example the peers reside on the same production facility. In this case they are on the same LAN. If they are located at different sites, it is assumed that there is a possibility for them to know each other - which is realized by assumption 1 - and also to connect to each other. This implies that obstacles like Network Address Translation (NAT) are ignored. How the exact communication between factories is realized, is not of concern. In sum network management is assumed. Referring back to the different models from Section 2.1 this point excludes problems that are mainly considered in the view of Logical Connections.

3. To simplify the setup and put the focus on relevant aspects, other technical barriers relevant in industrial context are left aside. For example firewalls or Demilitarized Zones (DMZs) are omitted. In a productive setting it can be assumed they are installed and set up with a reasonable rule-set.

4. The problem of data distribution is realized by file exchange. Data can be distributed in many ways. For example sockets can exchange data with each other. This can of course be used to transfer files if any upstream protocol specifies a procedure.

5. There is a total ordering on data blocks with respect to change versioning. This means that for data of version A and the data of version B there is a way to determine the higher change version of the two.

2.5. Summary

In this Chapter three possible views where proposed which emphasize different factors for data distribution. The first is concerned with Informational Groups with a bipartite hierarchy, manager nodes and regular nodes. The second one puts the focus on Logical Connections and inspects the network connections on a logical level. The third one examines data flow and communication processes in Data Groups.

Having established these base terms, changes in the context of Industry 4.0 where surveyed. Data distribution came up as an important factor. A scenario with the requirement for secure and reliable data distribution emerged. The first group of requirements was an appropriate Group and Identity management, which takes the complexities of an industrial context into account. Management shells can tie communication channels to a common base and abstract multiple agents on higher levels into Informational Groups.

A second group of requirements were the three classical security objectives defined in the norm ISO 27000, Confidentiality, Integrity and Availability, where the latter is very important in industrial settings. Additionally, to ensure these, Authenticity plays an important role for an Industry 4.0 scenario due to the increased possible threats through interconnectedness.

As a last requirement, Decentrality was examined which reflects the idea behind Industry 4.0 as well as it increases greatly the Availability by avoiding Single Point of Failures (SPFs) and provides scalability.

Lastly, the scope of the thesis was restricted to decentralized data distribution which assumes Identity management processes, end-to-end communication and appropriate enterprise infrastructure like firewalls.

3. Related Work and Background

In this Chapter approaches are examined which are relevant in the context of secure data distribution. First, in Section 3.1 possible approaches for P2P data exchange are discussed. It also investigates two ways for reducing the load, repeated exchanges may induce. Next, in Section 3.2 possible approaches for securing the communication between actors are inspected. Lastly, in Section 3.3 hardware possibilities for security improvements are examined.

3.1. Decentralized Data Exchange

This section is concerned with related approaches for decentralized data exchange. It discusses several possible designs for decentralized solution as well as proposed ideas for data exchange between two peers.

The task related tachnologies and algorithms have to fulfill is to build an efficient decentralized data distribution infrastructure. It is motivated by ensuring Availability and to stick to the paradigm of Industry 4.0 of interconnectedness. The challenges for such an infrastructure are to ensure the Integrity and Confidentiality of the Data. This is only possible if the Authenticity of the peers is ensured.

Tracker based Peer Networks. One approach for decentralized data exchange is a P2P network that uses a tracker topology as shown in Figure 3.1. A central peer acts as a *tracker*. Other peers may pull information on the network from this central peers and then connect to each other based on the information they received from the tracker.

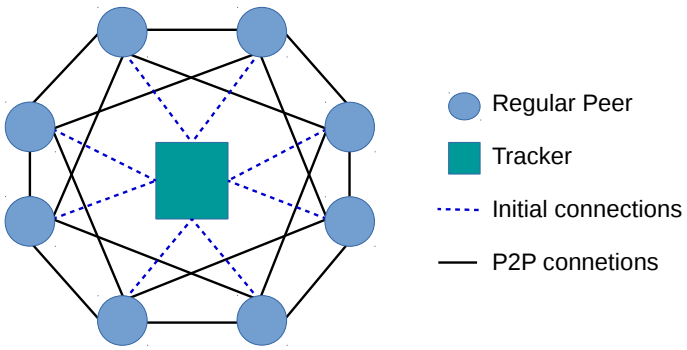

Figure 3.1.: **Tracker based topology**

One example implementing this topology is the BitTorrent protocol. For every file a dedicated peer tracks other peers which hold copies of parts of this file. A peer requiring the file pulls information from the tracker, which peer holds which part of the file. Next, it connects to these peers and downloads the single parts.

This approach distributes files efficiently between peers as it uses the capacities of other peers to distribute the data. Thus, the impact on the bandwidth for a central server is reduced.

The tracker has the drawback that it still acts as a central distribution server for information on the network. This implies two main risks which are related to the scenario. The first one is that the Availability of information on the network partners depends on this SPF. In cases of downtimes of the tracker devices the network cannot access the data anymore. Connected with this problem is the scalability as a single tracker may not be able to handle a large number of queries.

The second risk of this central point is concerned with Authenticity. The information on the peers of the network is centralized at one device. If the device is corrupted, the whole network cannot be trusted anymore. Additionally there is no mechanism implemented which ensures the Authenticity of the other peers. This is not directly a problem as long as the Integrity of the data is ensured. With BitTorrent this is realized by known hashes for the single parts of the files. If a malicious peer sends wrong data, the hashes would not match [49][39]. However, this approach does not satisfy the requirements for secure decentralized data exchange inf future industrial contexts.

Distributed Hash Tables. In order to avoid the centralized structure, distributed content-addressable data storage (distributed indexing structures) is proposed. These so called Distributed Hashtables (DHTs) were developed to provide distributed indexing as well as scalability, reliability, and fault tolerance. In the literature such P2P networks are also called structured P2P networks. Commonly, a data item can be retrieved from the network with a complexity of O(log N) which is equal to the complexity of well-known non-distributed search and indexing techniques[49]. One example of a DHT algorithm is Kademilla which performs good compared to other approaches and which has been implemented into BitTorrent [46].

To distribute the information former managed by a central point, every DHT node manages a small number of references to other nodes. Data items and nodes are mapped into a common address space, forming a lookup table of (key, value)-pairs which is distributed in parts to all nodes. Every node is responsible for a certain set of data items. Routing to a node leads eventually to the data items for which a certain node is responsible, reducing the distance to the data item. Kademilla uses a Exclusive Or (XOR) metric in order to achieve a targeted routing.

DHTs provide a global view of data, distributed among many nodes, independent of the actual location. For decentralized data exchange it is in general a possible approach which ensures Availability without the drawback of a bottleneck or a SPF like in the tracker approach.

The two approaches are not directly suitable for enterprise contexts for three reasons.

1. Both strive to make data available to all participants of the network. With reference to Confidentiality this is not acceptable. As Chapter 2 has already stated, a system is required which makes information only selectively available to peers, i.e. an Group management system is required. The existing P2P-approaches do not provide corresponding features.

2. Both approaches are suitably for retrieving files from the network. This makes it usable only for situations in which devices actively request files. For the scenario painted in

Chapter 2 the active distribution of data is as well important. There is no way to push files to a group of peers. If there are changes to a set of files, the peers do not actively try to get hold of the latest version. A recurring, timed retrieval of n seconds of the file set would solve this only partly as it would introduce a expected lag of $T = n/2$ seconds until the change is recognized.

3. Known problems like the Eclipse attack and the Sybil attacks raise concerns with Authenticity. The Eclipse attack tries to monopolize all connections to a peer. In a Sybil attack a single malicious device impersonates a bunch of different nodes. The goal is to isolate one node from the network without the node noticing it. Subsequently the attacking nodes are able to simulate a network with wrong information to the target. To prevent this, every node has to prove its Authenticity to each other node.

4. The discussed approaches so far have the drawback that with a change in a file the whole file has to be retransmitted. To reduce the data throughput it would suffice to only transmit the changes in the file. Solutions for this problem are discussed now.

Fixed Size Block Hashes. If two files differ only partly there is no need to transfer the whole file. Rather only the deltas need to be communicated. In order to build an efficient data exchange infrastructure this has to be addressed.

A common approach to realize this is given by cutting files into blocks. This technique is used e.g. in BitTorrent[1][39] or the Block Exchange Protocol (BEP) [54], which will also be explained in this Section. By interpreting a file as continuous bytestream it can be split into blocks of a fixed size. The last block is allowed to be smaller or needs to be padded. In a next step, for every block a hash value is calculated. Assuming the hash to be collision resistant, a change in a file would lead to a change of the hash sum of the respective blocks. By comparing the hash values of the blocks of two files, the differing parts of the files can rapidly be found. This procedure would allow a peer to ask only for the parts of files that changed, if it knows the hashes. This implies a two step exchange. First, the block hashes have to be exchanged and in a second step the concerning blocks.

By using the technique of block hashes not only the data throughput is reduced, it also ensures the Integrity of the data on application level.

Content Defined Slicing. With a fixed block size only the blocks are transferred that have changed. The approach of Content Defined Slicing (CDS) tries to address a drawback of this approach.

Suppose that bits of data not equal to the block size are inserted somewhere in the file which might happen quite often. Then, from the insertion point on to the end of the file the bytes are shifted right while the position of the blocks for the hash calculation stays the same. This is shown in Figure 3.2. As a consequence a change of the hash of all following blocks occurs starting from the insertion point block. At the end an incomplete block might be generated whose hash also has to be calculated. In the worst case appending at the beginning leads to a complete retransfer of the whole file. Essentially no data changed (there was only data added) but due to the changed hash sums all blocks would be retransmitted.

[1] BitTorrent splits a file into slices in order to distribute the different parts to different peers. Once all slices have been distributed to the network, the tracker can be asked for the location of missing slices. This approach still misses the required use case of retransmission of changes, as there is no way to request the

Figure 3.2.: **Change of hash sums of all block from insertion point of data**

CDS addresses this problem by using rolling hash algorithms with a following check if the computed hash fulfills certain criteria. If so the corresponding block is used as a slice. This results in blocks of different sizes based on the contents which eliminates the problem of shifted data as long the content does not change. An example application implementing Content Defined Slicing is BorgBackup using the rolling BUZ Hash [58] [16].

The functioning of a rolling hash is explained in Figure 3.3. A hash window of size 8 is chosen. In the first step a hash value over the first 8 bits of data is calculated. The resulting value is `0x1234abcd`. This candidate block is then checked against some criterion, e.g., if the last 2 bytes are 0 using the hash mask `0x0000ffff` which extracts the last 2 bytes. In this case one calculates: `0x1234abcd & 0x0000ffff` which is not 0. In the next step the last 2 bytes of the hash value are 0. So `0x08760000 & 0x0000ffff` evaluates to 0 so a block has been found. In this example it becomes clear that the bit mask is crucial for the block size. The easier the condition is to match the smaller the blocks will be (if the hash function produces uniformly distributed results). In the case of BUZ hash the calculation of the hashes is efficient because for every move of the hash window only two bytes change and the values for the rest of the bits can be reused. [3].

Figure 3.3.: **Rolling hash**

BorgBackup thus implements for some occasions a more efficient way for data transfer. However it is intended as a backup solution. For decentralized data exchange it is not usable

status of single slices.

as it sticks to a client-server-architecture.

Block Exchange Protocol (BEP). The BEP defines a procedure for file synchronization between multiple devices in a decentralized manner [38]. Every device holds a *local model* of the configuration of all tracked folders. Tracking means the contents but also the meta data like permissions, chage time and other information; particularly the *block hashes* are monitored: Every file is split up into blocks of size 128 KiB. The `BlockInfo` contains the size[2] and position and additionally a hash sum of each block. This information is stored in the local index records. The union of all files in the highest change version of all peers is called the *global model*. The task, the BEP fulfills, is to update the local model of every peer to the global model. It does this by requesting data from other peers that are more up-to-date than the local instance. The procedure is as follows (assuming that the peers are connected and authenticated to each other):

In a first step, each peer advertises the directories it wants to synchronize together with the devices that are assigned to these directories, as well as other properties [3].

In a second step, the peers advertise their local index records to each other. After a device has received this information, it can compare the hash of each block of each file to the list of hashes it got from the peer. If some blocks do not match, a change has occurred in the file. If so, the peer has to check which version of the file it holds. For the case the other peer holds a higher change version, it request the not matching blocks in a third step. If it already holds the latest version, the other peer should request the affected blocks.

3.2. Identity Management

To ensure security in decentralized infrastructures the single devices need to authenticate themselves to each other, as already motivated by examples like the Eclipse attack or the Sybil attack. This Section addresses the security goal of Authenticity. Confidentiality and Integrity of data also can only be ensured if there is no way for an attacker to infiltrate a distribution infrastructure. It is required for the peers to establish secured and authenticated end to end connections.

A proper management of Identities is also important for the separation of communication domains. As motivated in Chapter 2 not every bit of information should be accessible by everyone, especially for cross company setups.

To ensure Confidentiality for a channel asymmetric cryptography can be used. To motivate the requirement of Identity management and to show ways for mitigation, a Man In The Middle (MITM) attack is now explained.

A simplified version of a public key exchange is depicted in Figure 3.4. If A wants securely communicate with B it sends as step 1 a Hello to B. B knowing that A wants to start a encrypted session, sends its public key B_{pub}, step 2. If A sends a message to B, it encrypts the message using the public key it got from B and only B is able to decrypt it, given that B is the exclusive owner of the private key. Now they are able to establish a secure connection,

[2] The size field is required for the last block of a file, which may be smaller than the normal block size.

[3] Those properties are mainly made up of synchronization information. For example the protocol supports the skipping of deleted files which then will not be deleted. Other information contains for instance compression modi.

e.g., A may send an encrypted session key. This is a simplified version of the proceedings. It shows only the required parts for explaining a MITM attack.

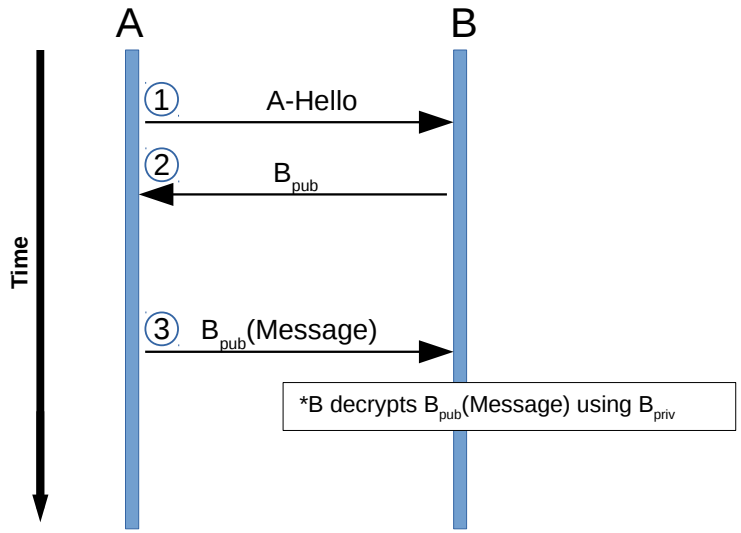

Figure 3.4.: **Simplified key exchange for the establishment of a secure channel.**

A MITM attack sets in at the key exchange as shown in Figure 3.5. When A sends an Hello to B, an attacker M intercepts it and forwards it with the modified address pretending to be A, step 2. When B sends its public key to A (step 3), M might intercept the message and send its own public key to A pretending to be its desired partner B (step 4). If A wants to send a message to B it would use the public key it got from M. Now, M can decrypt the message, read it, and then encrypt is again using the public key it intercepted from B. In this simple case there is no way for the partners to find out, that their keys have been intercepted and they are using the wrong key of M.

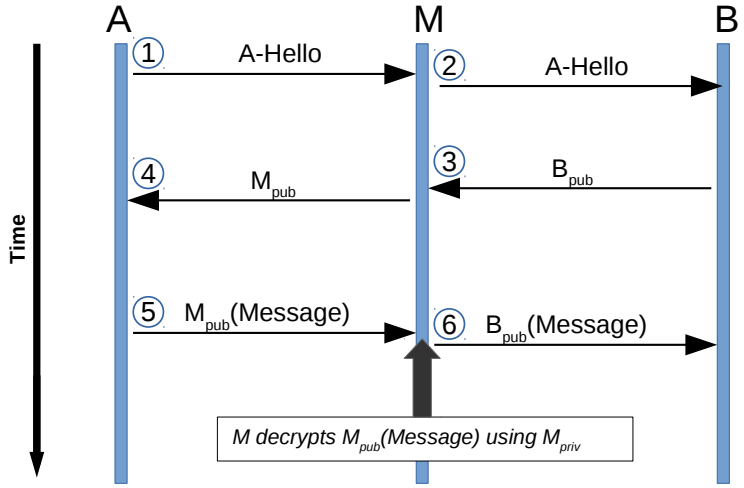

Figure 3.5.: **MITM attack on Key Exchange**

A proper Identity management aims to prevent these kinds of attacks. Protocols like

Transport Layer Security (TLS) provide possibilities for ensuring Authenticity through asymmetric cryptography. In TLS, public keys are normally distributed through certificates which hold information on the owner of a public key and the key itself. The Integrity of such certificates is assured by cryptographic signatures. For the sake of the argument it is assumed that the certificates are signed using the key pair of the owner. So if M would intercept a certificate and exchange the public key on it with its own, an integrity check of the certificate would fail. B would know something is wrong. However M might simply craft its own certificate with its own public key on it and a correct signature. There is still no way for B to check the Authenticity of the certificate. Public Key Infrastructures (PKIs) are designed to use not self signed certificates to solve this problem. Another approach is Public Key Pinning (PKP). Both are examined now.

Public Key Infrastructures are a common way to ensure Authenticity for TLS. The goal is to enable devices to verify the binding of a public key to an entity. This commonly happens by using a Certificate Authority (CA). One reason to implement a PKI is to reduce the complexity of Identity management.

In PKIs certificates are signed by a Trusted Third Party (TTP). Every entity of a network would have to generate its own public/private key pair. Then, the single entities have to approach a CA which issues signed certificates for them. This CA acts as a TTP. The task of the CA is to ensure the identity of the requesting entity. Any device which receives a peer certificate for a TLS secured connection has to verify the Integrity of it. Therefore, it needs to hold the public key of the signer which is the CA. Therefore this approach requires every entity to have access to the public key of the CA, which is normally distributed in form of self-signed certificates. In an enterprise the TTP could of course be part of the enterprise itself. PKIs also allow for certificate chains in which the root CA needs to be trusted. Other CAs can then be verified in a chain of trust.

A MITM attack would not be possible anymore as the attacker M should not have access to a certificate signed by an trusted CA. This requires the private keys of the certificates not to be stolen. For the case of a key compromise a PKI normally has a mechanism in place through which entities can check received certificates against a list of invalidated ones.

A PKI depends strongly on trust of central entities whose keys must not be stolen. If so, all certificates issued under the regarded key do not assure Authenticity anymore as the holder of the stolen key is able to calculate valid signatures. Security Modules are a possible approach to keep private keys safe; they will be examined in Section 3.3.

The PKI approach is normally taken for large infrastructures due to its good scaling. The implementation of a PKIs normally requires major conceptual work beforehand due to the complexity [3]. However, the complexity allows for fine granularity of rights through sub keys. Another advantage is the implementation of a revocation list. This property fits into the requirements of an industry context as, e.g., working groups may have only a limited lifetime and after that time the corresponding information group should be removed; this could be realized by revoking the corresponding certificates. Another way to realize this is to confine the liftime of certificate.

A drawback is that the CA is a SPF. If the private key is stolen, all signed certificates are invalidated at once. Also the management might produce an unnecessary overhead. Some of the features may not be required.

For realizing proper isolation of shells, as explained in Chapter 2, another process is required

which binds entities to groups. If a peer A asks for data of another peer B, both have to authenticate themselves to each other. This Authentication is not simply given by each peer providing its CA signed certificate. It also requires for both peers to check if the other one is part of the same logical shell.[4] PKP enables administrators to take care of this problem on the application layer above the TLS layer.

Public Key Pinning realizes Authenticity by exchanging the public keys of communication partners beforehand. Another way is not to exchange the public keys but only cryptographic fingerprints of the public keys. In either case a peer can compare the public key of a partner's certificate with the expected one.

This requires a secure distribution of the public keys. As for every node of a network all connections to other peers of the network are required to be defined beforehand, the complexity is strongly dependent on the number of connections, it scales with O(n!) [3].

A benefit of this approach is, that it directly addresses the problem of managing groups. For a given group, the members of the group are defined by the distributed keys. This allows for a direct control on the topology of a group as the connections between the peers are defined individually.

For the case a private key is stolen, all fingerprints of the corresponding public key have to be invalidated on each partner device. To minimize the risk of private key theft and the work of reconfiguration of all respective devices, again Security Modules can be used.

3.3. Hardware Security Modules

Security Modules have the task to provide secure storage and generation of cryptographic material. They are designed as a root of trust for cryptographic operations. Their functionality is supplemented through a crypto API which provides a machine with hardware supported crypto functionalities. In the context of this thesis their benefit is to secure end-to-end communication between entities.

In order to ensure trust, a device should act exactly as one expects. Thus, standards play an important role for Hardware Security Modules (HSMs).

The cryptographic material is stored on an external physical device which is not accessible by software but can only be communicated with through an API. The secret keys should never be accessable outside the device. Thus, the keys are never hold in the memory of a machine. This ensures that the key cannot be compromised by attacks like RowHammer [11] or Heartbleed [3].

Examples are smartcards or the Yubikey [57] for single users but also server attachable devices which can be used in server infrastructures. They are designed to be tamper resistant which means that they, e.g., delete their memory if one tries to access it physically. In general side channel attacks should be taken into account. Also, unauthorized access should leave evidence.

Trusted Platform Modules are a group of HSMs which are designed according to an international standard written by the Trusted Computing Group (TCG) [28]. Especially a TPM is able to proof its own Integrity to the user [47]. In the year 2014 the latest version of the Trusted Platform Module (TPM) specification was published, the version TPM 2.0.

[4] This might be realized through multiple CAs, each acting as a signer for every logical shell.

TPMs are microcontrollers that aim to provide a root of trust for maintaining the security policy of a system, i.e. it allows an independent entity to determine if the trusted computing base has been compromised. This can be used to prevent a system from starting if it has been tampered with [25, p. 21].

There is an encryption key permanently embedded into the TPM which cannot be deleted or altered and which never leaves the device. This Endorsement Key (EK) is the root for all derived keys. The Cryptography subsystem implements the TPM's cryptographic functions. Available operations are

- hash functions,
- asymmetric encryption and decryption,
- asymmetric signing and signature verification,
- symmetric encryption and decryption,
- symmetric signing (only HMAC currently defined) and signature verification,
- key generation [25].

Aside of this subsystem there are several others of which important ones are

- an extra Random Number Generator (RNG) Module. RNG has an entropy collection module which seeds a mix function (not specified in the standard).
- an Authorization Subsystem. It is called by the Command Dispatch module at the beginning and end of command execution. Before a command cacn be executed, the Authorization Subsystem checks that proper authorization for use of each of the Shielded Locations has been provided. Shielded locations are areas in the TPM which are security sensitive and, therefore, particularly protected.
- A Non Volatile Memory (NVM) [25].

There is not much space on the TPM, so the preferred way to store information is outside the device encrypting it by using the EK. This also holds for derived private keys. If one needs access to the data they have to be decrypted sending them into the TPM where they are decrypted and sent back out. More detailed functionality is explained in Chapter 4.

TPMs provide a standardized interface, which makes it easy to integrate them into IoT manager devices for industrial contexts [20]. As they ensure a secure storage of the private key, they are suitable for hardening the end-to-end communication between peers. Through the EK they can provide Authenticity of a device if the corresponding public key is known.

3.4. Summary

In Section 3.1 possible approaches for P2P data exchange where discussed. Tracker based methods where rejected as a possible approach for decentralized data distribution for the intended context. Firstly, the requirement of Decentrality is not realy satisfied as well as they lack the possibility of Identity management and related problems. Next, structured P2P networks where examined, whose distinguishing factor is a DHT. They scale well and are

decentralized. Still, they lack the possibility of Identity and Group management. Additionally, they lack required features for ensuring Authenticity on a reliable basis. Also, they are designed for requesting files, not efficiently distributing them in industrial contexts.

For reducing the load of exchanging already shared files, which differ only partly, two approaches have been discussed. The first one uses fixed sized blocks. Another possibility is CDS, which tries to overcome the drawback of data shifts resulting from insertions or deletions.

Having explained fixed block methods, the BEP was introduced as a method for file synchronization between peers.

In Section 3.2, first, the problem of impersonation was described, which endangers Authenticity. Thereafter, PKIs and PKP where discussed as possible approaches for ensuring this security objective. Which one to use depends on the case.

Lastly, in Section 3.3, HSMs where propsed for improving the security of data. TPMs where examined as a subset of HSMs which are standardized by the TCG.

4. Setup and Implementation

This Chapter proposes a concept for decentralized data distribution, building on the models from Chapter 2. It takes into account the respective requirements and additionally the findings from Chapter 3. Afterward, the setup for an exhibit is shown which aims to implement a subset of the Concept to examine the Concept. For preparation, in Section 4.2, the used software and hardware is introduced. Afterwards, the implementation is explained. Lastly, in Section 4.4, the exhibit is evaluated. The Section surveys the requirements and the applicability for other cases.

4.1. Concept

The problem to be solved is a decentralized and secure data distribution, which satisfies the requirements of the new industrial context which was evaluated in Chapter 2. End-to-end data flows across different sites through the Internet and across different organizational levels have to be secured. As it emerged in Chapter 3 several P2P approaches are not suitable for this. One way to realize this is to configure dedicated communication partners in order to form information groups. In this section a suitable concept is outlined.

The concept is based on the scenario from Chapter 2. The requirements are listed in Table 4.1. From Chapter 3 additional points are derived which are not directly requirements from the Scenario but are as well relevant for it. They are also included in Table 4.1.

On a single device a wrapper software aggregates data into files and manages the organization and exchange with other peers like the proposed management shells from the RAMI 4.0 standard. With respect to group management the concept requires to make logical structuring of groups possible. A management process is assumed and outside of the scope of this thesis, as stated in Section 2.3 in Chapter 2.

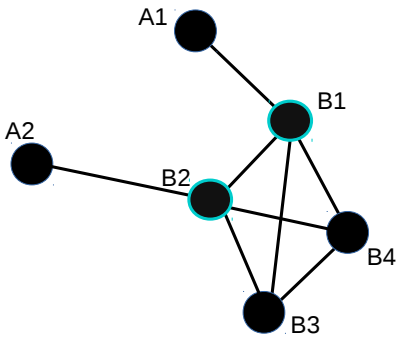

Figure 4.1.: **Exemplary Group Setup**

Requirement	**Short Explanation**
Confidentiality	If Information is transfered over untrusted networks, e.g. the Internet, it should stay Confidential. Also Information needs to be separated according to Informational Groups as proposed in Chapter 2.
Authenticity	For companies connecting lots of devices across hierarchical levels and to external partners, Authenticity is urgent. First, there needs a way for the devices to avoid MITM attacks. Second, the devices need to be able to verify the membership in respective Informational Groups as explained in Chapter 2.
Integrity	Data needs to be intact after transfer. Another point is consistency which can be subsumed under Integrity. It is evaluated as an extra point in the table.
Availability	For industrial contexts Availability is an important security objective. Availability of data is a major factor in Industry 4.0. Failures of single devices should lead to a breakdown of the distribution system. The requirement behind this is, that data should be available when it is needed.
Decentrality	For Industry 4.0 a strong interconnectedness and agile environments, together with the desire for self organizing setups, Decentrality is a reasonable requirement. It ensures Scalability and Availability.
Group and Identity management	There must be a way to separate information for Confidentiality which refers to Informational Groups. Also, Authenticity must be ensured, which requires an Identity management.
Delta Exchange	There is no need to transfer the complete files if they have changed. It suffices to exchange only the changed parts in order to reduce the network load.
Further Hardening	For industrial context the loss of important information like encryption keys can be disruptive. Using HSMs the security can be inceased significantly.
Data Consistence	With Integrity this point is indirectly covered. It stresses the fact that data distributed to a cluster should eventually be consistent. There should also be no infinite running loops which relates to Time, the next point.
Time	Data should be distributed in appropriate time. This point is already included in Availability. The exact time is not of interest. But it is relevant that data is available if it is needed. If it is needed in real time an instant distribution is necessary.

Table 4.1.: Summary of the requirements for the exhibit

For the concept an example is given in Figures 4.1 and 4.2. The first Figure takes the views of Informational Groups and Logical Connections. The B-nodes form a group. B1 and B2 act as Management nodes for the group and have connections to other nodes outside the group. In the real world this setup can be thought of as the group B being an enterprise which shares certain information with two partners. The intern connections of the group B are not visible for A1 and A2. Of course it is possible that behind these two also complex groups are hidden, for which the two act as Manager nodes. The same holds for B3 and B4. Thus it becomes clear that data distribution for a single group has to be examined. If data has to be forwarded to nested groups the concept can be applied recursive.

Next, the third view of Data Flows needs to be taken into account. To recall, for a data group there exists one Master which has an authoritative function. Exemplary it is assumed that there are configuration data to be distributed to the peers in group B. Let B1 be the Master for the group. The new data are made available for B1 and have to be distributed to its Slaves. This is shown in Figure 4.2. The role of the Master is not bound to a dedicated device. With the concrete view of Data Flow each member of a Data Group may be the Master, leaving aside computational requirements. This does not hold for the Informational Group view. Here the Manager is required to be appropriately connected and configured, as it represents the group to external entities and needs to filter information.

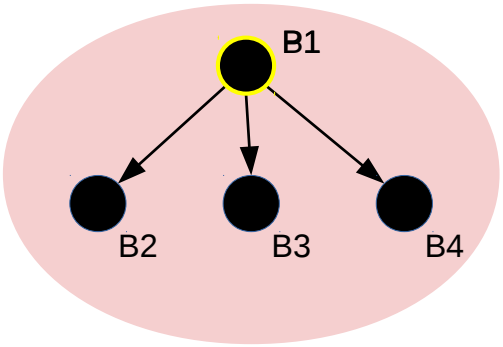

Figure 4.2.: **Data Flow for the exemplary Setup**

of the single files. If for every node its *local model* equals the *master model* the data has been distributed successfully. This description is inspired by the BEP [54].

Figure 4.3 proposes a concept for the communication steps required.

1. The Master initializes the connection.

2. The two partners establish a secure channel to protect further communication, e.g., TLS. This step is abstracted, as it may be realized in several ways. Any reliable method can be inserted here.

3. Each partner authenticates itself to the other. This step is also abstracted. It should ensure that a) every partner is who it claims to be and b) that they are indeed members of the same Data Group, including the roles.

4. The Master pushes meta info of its model including the change version.

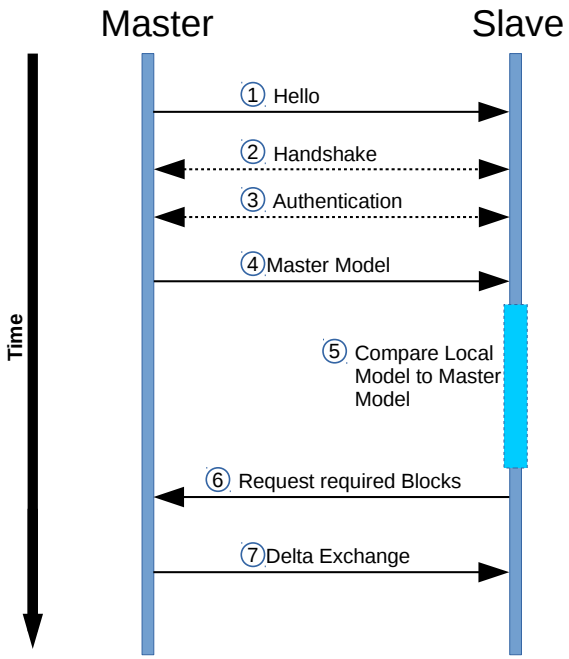

Figure 4.3.: **Communication Concept between a Master and a Slave**

5. The Slave compares for every file its local change version with the one it got from its Master.
6. The Slave sends a request for all differences in files of which it does not hold the latest version.
7. The Master pushes the differences to the Slave.

Each of the endpoints has to authenticate itself to the other. The connection must be secured end to end, thus the network in between is assumed to not be trustable. In Figure 4.3 the Master initiated the connection. In this way the Master can transfer new data without delay to its Slaves. Another approach would be to let the Slaves periodically connect to their Master. This would invert step 1. For an interval of time T the mean waiting time until the transfer starts would then be $T/2$. It depends on the concrete case which approach fits best. Step 4 to 7 can also be realized the other way around: The Slave could send its local model to the Master. The Master would then compare it to its own model and push the differences.

In the concept above the case of distributing data from one Master to several Slaves has been illustrated. For the case that, e.g., sensor data are aggregated by the Master, the approach still works. Depending on the concrete case, a change in the local model of a Slave may trigger the communication to the Master or the Master periodically asks for changes. The same holds for steps 4 to 7. There is no concrete direction required. The ordering of the change version allows for both ways.

Availability is to be realized by building on the paradigm of Decentrality. Thus, there should be no server required which acts as a storage center for nodes sharing data. The Concept above requires a node to have the role of the Master. If the Master drops out there needs to be a protocol how the Master can be replaced by another node.

Confidentiality and Integrity of the data is to be realized by cryptographic means. As already mentioned a standardized way of secure communication like TLS is favorable. In the concept this is abstracted from in step 2.

Authenticity is of great importance as Confidentiality and Integrity of data can only be assured by reliable Authenticity. If it is compromised the other two are not ensured anymore. In the concept above Authenticity consists not only of assuring the peer's identity but also of ensuring the membership of the relevant group.

Taking again the view of Informational groups, there is the case of multiple Manager nodes. The concept from above forces one node to be the Master. Given the version ordering there should be no problem as for every file the latest version is defined. The role of Manager nodes is not to be confused with the role of a Master. If for an Informational Group files simply need to be synchronized between all members or between two Managers, this can be realized by applying the concept from above multiple times as shown in Figure 4.4.

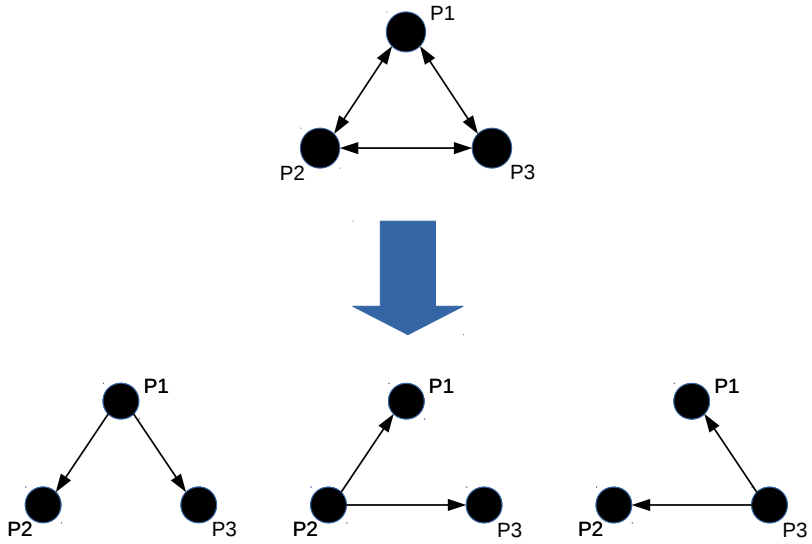

Figure 4.4.: **Distribution Between Equal Nodes**

The remaining part of the Chapter explains the realization of an exhibit. Its goal is to show how the evaluated requirements can be implemented building on the explained concept of data exchange. To simplify the setup and to be able to put focus on relevant aspects, again, assumptions are made that where already enumerated at the end of Chapter 2. In a productive setting it can be assumed that solutions for these assumptions are installed and setup with a reasonable rule-set. These are an appropriate Identity management, a network with possibility for each device to connect to each other, and properly configured firewalls.

The exhibit implements an exemplary part of the concept. It shows the data exchange for two devices with possibility for expansion in mind.

4.2. Technologies Used in the Setup

The exhibit setup emerged out from the idea of using Infineon Iridium SLB 9670 TPM 2.0 SPI Boards [32] for securing end to end communication. This evaluation board allows interfacing over SPI. Infineon provides a patch for Raspian, a dedicated Debian-based Linux distribution

for the Raspberry Pi (RPi) [33]. Using the open source Linux kernel has the advantage, that the code can be investigated for security issues. It is also modifiable to adjust it to special situations. As Raspian was chosen as Operating System (OS), corresponding boards, RPis, where used. These mini-computers are suitable as they provide pins which are configurable for SPI and also have Ethernet ports, which were used for setting up a LAN.

In order to realize decentralized data distribution, the program Syncthing was used [38]. The software implements decentralized data exchange using the BEP. It is publicized under the Mozilla Public License 2.0 which allows to examine and modify the code. The Authenticity is ensured on application level using PKP. The certificate of a peer, sent for a TLS handshake, is only checked for Integrity.

Another reason for choosing this software is its general approach for communication based on TLS. Communication standards are of high importance for Industry 4.0 as already mentioned. Building on this approved standard ensures security through well tested technologies. Using a management shell for a device which makes use of uniform communication channels has already been motivated. Syncthing can be interpreted as a simplified version of a management shell which aggregates data on a device and shares it with others.

Syncthing is written in the memory safe programming language Go [50]. This reduces the risk of possible attacks and errors like buffer overflows. Through its static typing system it also reduces the possible errors. Its platform independence is also of advantage. Syncthing aims to be secure against attackers and be safe from data loss or corruption. The exhibit implements an approach for improving the security by using TPMs. The part for this task was also written in Go.

For addressing peers Internet Protocol Version 6 (IPv6) was used. This has several advantages as for example it enables end-to-end addressing. The exhibit should show a possible implementation for an Industry 4.0 scenario which relies presumably on IPv6.

Before the implementation of the exhibit is explained, relevant basics for the TPM are introduced as it is necessary for using it. As already, stated a TPM has so called shielded areas. Those may only by accessed if proper authentication is provided, given authentication is configured. For accessing shielded locations a TPM uses handles or objects which need to be referenced or loaded into the TPM. A handle references one action. This is necessary as actions depend on different hierarchical areas.

Hierarchical areas are, e.g., the owner of the TPM who may set authentication values (ownerAuth). Another area is the endorsement hierarchy for normal storage and handling (endorsementAuth). A third one is the lockout area which manages the dictionary attack prevention mechanism (lockoutAuth). If the TPM is reset or cleared these values are empty [25].

Handles are needed to reference structures in the TPM which should be manipulated. These handles are values which the TPM sets. A handle identifies the shielded location and is only required if such areas are accessed. A handle may require authorization values loaded into the TPM. Using TPM objects allows to store the context of commands which means, e.g., necessary internal environment variables and also the handle of the action, making it possible to use the commands in a non chained manner by loading these context objects. This was used in the implementation.

In order to use a TPM for cryptographic actions using secret keys, these keys need to be generated and stored. For storage a Storage Parent is required which grants protection for the area in which the respective keys are stored. Such a Storage Parent key can be generated

from the Storage Primary Seed (SPS)[1]. It is generated new at restart or when clearing the TPM. A SPS is used for the Random Number Generator (RNG) module as seed to generate keys. If a SPS changes, all objects in the corresponding Storage Hierarchy are invalidated thus the data is not recreatable [25].

For communicating with devices designed according to the standard, the Trusted Computing Group made the code for the TPM2 Software Stack (TSS) available [36]. It is based on the specification for the TSS System Level API [53]. The TCG also provides the code for the TPM2 tools, which builds on the TSS and makes tools available to interface easily with the TPM device [35]. Both are used for the TPM addition.

4.3. Establishment of the Setup

In this Section the realization of the exhibit is explained in two steps. First, the hardware setup is shown. Afterwards, the software and written code is explained.

4.3.1. Hardware and Operating System

The exhibit consists of two RPis connected to a switch using Ethernet cables thus forming a LAN. The configuration is shown in Figure 4.5. On each of the two RPis a TPM is attached, indicated by the black boxes. The controlling machine is also attached to the switch and is used to log into the RPis via SSH. The two mini computers have fixed Internet Protocol (IP) addresses for their interfaces eth0 and use the Internet Protocol Version 4 (IPv4) and IPv6 address of the controlling machine as default gateway as shown in the Figure. For purposes of installing necessary software the controlling machine was configured with network forwarding and NAT using iptables.

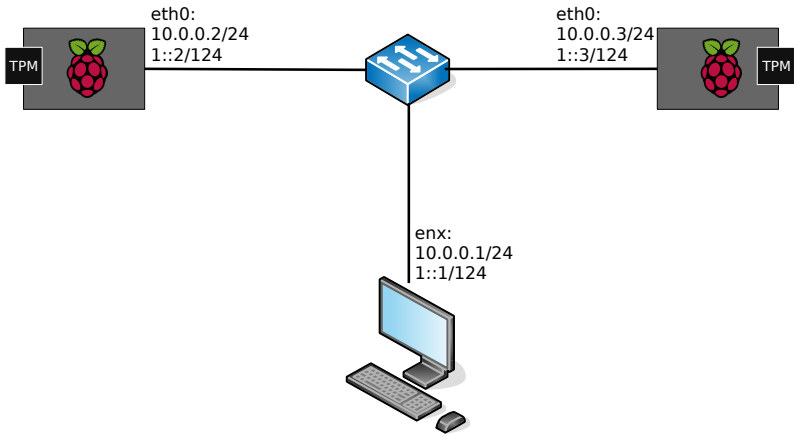

Figure 4.5.: **Schema of the Setup**

Each of the two RPis runs a modified version of the Raspian OS, with the available Infineon patch for using the TPM evaluation board compiled into the kernel [33]. The exact steps for this are available in the Appendix of this thesis.

[1] A Primary Seed is required to have at least twice the number of bits as the security strength of any symmetric or asymmetric algorithm implemented on the TPM

With the kernel being able to use a TPM device at `/dev/tpm0`, the required tools where installed, i.e. the TPM software stack and the TPM tools, for which the steps can also be found in the Appendix.

4.3.2. Implementation

Now the implementation of the exhibit is explained. First it is shown how decentralized data distribution was realized. Afterwards it is explained, how TPMs where used to increase the security.

Implementing Decentralized Data Distribution

The data exchange between peers is realized by installing Syncthing on the two RPis. This provides exchange via the Block Exchange Protocol. This protocol defines procedures for communication, authentication and file synchronization between devices [38] as explained in Chapter 3.

To summarize, the BEP uses index records to update the local model of every peer to the global model, which is defined as the set of all files in the highest change version. To realize deduplication, files are managed as blocks of a fixed size.

Step 2 of Figure 4.3 is realized by Syncthing through a TLS handshake, exchanging self signed certificates. The handshake is initialized by a BEP Helo message which indicates the start of an synchronization wish (step 1). After the certificate exchange a secure channel based on the included public keys is possible. The two devices then arrange a symmetric session key.

Step 3 is then realized by PKP. The Authenticity is ensured through *IDs*, which have to be manually added to the single devices and consist of the base32 encoded SHA256 sums of the public keys, with added correction bits based on the Luhn algorithm [54]. PKP has already been motivated in Chapter 3 as a possible method for ensuring appropriate Authenticity.

Step 4 from the concept is split into two parts. First, each peer advertises the tracked directories and the associated peers. Then the peers advertise their local index records to each other, compare them to their local model and request missing or depricated blocks (step 5). This procedure has already been examined in Chapter 3 and corresponds to step 4 of the concept.

In the implementation of the exhibit a comparison of the local models of the peers occurs based on a time interval of 30 seconds. Syncthing also supports a reactive behavior, which initializes a connection after files changed in any of the monitored folders. On Linux this feature builds on the inotify kernel module [22].

The BEP, as implemented by Syncthing, realized the Concept from above in a modified way. If the steps from Figure 4.3 are applied twice, one time in the mirrored way, the implemented procedure is realized, comparable to Figure 4.4.

Security Improvement through TPMs

The certificates are stored in a local config directory, including the private key. This poses risks on two levels. Firstly, the private key can be accessed by the user, under which the process runs. Thus a secure handling is not ensured, especially if someone gets unauthorized access to the account. Secondly, in the process of the certificate generation as well as in the process of the TLS handshake the private key is stored in the memory of the devices.

In order to address the problems, efforts for hardening were undertaken. In Chapter 3 it has already been motivated, that this kind of problems can be addressed by TPMs. The approach aimed to integrate TPMs in order to provide secure storage for the private keys.

This part of the exhibit consists of three code files. The first one implements functions for a TPM library. The second file implements a simple server which builds on this library for creating a Certificate with the private key stored on a TPM. This certificate is used for providing a TLS port for connecting to the server. For the handshake the TPM is required as the private key is stored on it. The third file implements a client connecting to a server. It uses one function of the library for PKP which compares a given fingerprint with the fingerprint of the servers public key it wants connect to.

The implemented TPM library builds on two interfaces provided by the Go crypto library [31]. These two interfaces act as hooks to implement own subroutines for asymmetric encryption which can be used in the Go tls stack. In Go interfaces are satisfied implicitly by simply implementing the functions an interface type requires. This Section sticks to the Go convention, in which capitalizing names of functions and variables makes them public. Library references and basic types are in lowercase, e.g., 'crypto' refers to the library, 'struct' refers to a struct type and 'crypto.PublicKey' referst to a public accessable type of the library.

The first type is the Decrypter interface which requires the two functions, `Public()` and `Decrypt()`. Any self defined type which implements these two function can be used as an opaque private key in Go asymmetric cryptography for decryption.

The second one is the Signer interface which also requires the `Public()` function and additionally the `Sign()` function. An opaque private key type implementing this interface is usable for Signing.

Any self defined private key, implementing both interface is fully capable of asymmetric cryptography as used in TLS. The Go library calls the private key therefore an *opaque* key. The two interface suffice and there is no additional function like `encrypt()` needed due to the nature of asymmetric cryptography: If any data requires to be encrypted for a receiver, one uses the public key of the receiver to do so. The private key is needed by the receiver to decrypt the data which is ensured through the `Decrypt()` function. The second case that the private key is required is for signing which is implemented by the `Sign()` function. Using the private key for encryption would not make sense as the public key for decrypting is public. The public key is available through `Public()` which is only required for distribution to communication partners, e.g., by including it into a certificate.

This architecture allows to store the private key in a TPM and define the three function `Decrypt()`, `Sign()` and `Public()` for a indirect usage of the private key. The exhibit's TPM library does so by implementing them for an own type - Tpm struct - which can then act as an opaque private key for asymmetric cryptography. The struct contains the following fields:

Tpm struct:

- PublicKey. A `crypto.PublicKey` which is an abstract type and in practice is an untyped pointer. This field is implemented in the struct to satisfy the interfaces.
- PrivateKey. A `crypto.PrivateKey`, same as PublicKey.
- WorkingDirectory. A string which points to a directory on the system. This directory is the working directory for the TPM actions. In this directory, e.g., the context structures are stored and also other outputs like random numbers or signatures.

- Certificate. A `tls.Certificate`. There exist two certificate types in the crypto library which are used in a TLS connection in the setup. The first one is the *tls.Certificate*. It stores information relevant for managing TLS connections. Most important an instance contains an opaque private key and an unparsed DER encoded certificate.[2] The second type is *x509.Certificate*. This type works as a container for attributes of a x509 standardized certificate. The *leaf* attribute of a `tls.Certificate` is of this type and can automatically be parsed from a bytearray which holds a raw DER encoded certificate.

The Tpm struct thus has two tasks. First one is to store relevant information like the working directory and a `tls.Certificate` which also contains the encoded certificate. The second one is to act as an opaque private key. It is the receiver for the implementation of the interface required functions which will be explained in detail later. Next, the flow of the exhibit is explained step by step. For the Tpm struct a `New()` function is implemented.

New() function: This function takes care for the exhibit to reset the working directory and fill all required fields in the Tpm struct. It also initializes all required keys by calling the `InitKeys()` function and uses these keys for generating a x509 certificate. For this it also uses the random module of the TPM, which will be explained a bit later.

InitKeys() function: The purpose of it is to get the TPM device ready for TLS communication. It traverses the following steps:

1. Clearing the TPM ownership. After a restart the authentication values are empty. However, if the TPM was not restarted this approach ensures a smooth usage of the device as proposed by the manual [34].

2. Creating a primary key for storage. This key acts as the Storage Parent for keys created in a storage hierarchy as explained in Section 4.2. The name of the TPM object required to work on this is computed using the SHA256 algorithm and stored as a file in the working directory. The chosen encryption mechanism is RSA.

3. Creating a keypair. The public part is included in a x509 Certificate in the next step. The private part is used for signing of this certificate and decrypting of messages encrypted with the public key. Again a TPM object is created as output which is required for the keys to be used. The used algorithms are the same as for the primary key.

After the initialization of the required keys, a x509 certificate is created. The fields for the certificate are mainly adopted from the implementation of Syncthing [51]. An example certificate can be found in the Appendix.

Read() function: Like in Syncthing, the Serial Number is generated randomly. In the implementation for the exhibit, a Random Reader type is included which satisfies the go Reader interface. Using the `Read()` function, random bytes can be generated using the TPM

[2] The opaque private key only needs to satisfy the explained interfaces for getting the program compiled. It should, however, use the corresponding key to the public key of the encoded certificate. Otherwise cryptography will fail.

device. In this way the Serial Number is generated. The higher bit is masked in order to ensure a positive interpretation of the 64 bit integer.

The certificate is self signed. For creating it, the opaque private is passed to the generator function. As all steps so far work on the Tpm struct it can use itself for the private key. Also, the public part of the key pair which has been initialized by `InitKeys()` is included in the certificate.

Public() function: For accessing the public part the `Public()` function is implemented as required by the two private key interfaces. This function reads the public key file generated with the initialization. It contains the public key unencrypted. In practice the file is a TPM generated struct and the public key has to be extracted. The function first creates a PublicKey struct from the go rsa library which contains the two fields Modulus and Exponent. The latter is a fixed value, as the TPM specification proposes, 65537. This is also recommended by RFC 4871 [5]. The Modulus can be read from the public key file. However, the object also contains data for the TPM device. It is defined in the TPM specification. The object created by TPM2_Create is a TPMT_PUBLIC object which also stores information on the type, the used algorithm and several other fields [25, p. 174] [27, p. 45]. The relevant bytes are the bytes 102 to 358 [26, p. 135]. The file is read and the the relevant trunk is converted to an Integer and set for the Modulus.

Sign() function: The `Sign()` function uses the TPM and the keys generated in the `InitKeys()` function to sign a given digest. It assumes, that some text for which a signature should be created, is not directly passed to the function but a hash sum of the text. Two TPM related calls are required to achieve this. The first loads the required data into the TPM which are stored external. This produces a new TPM object file which contains information about this load action and is required for the loaded data to be used. The second call then outputs a RSA signature of the input digest. Such an action also produces a file which documents the action. After reading the digest from the relevant file, all data which is not relevant anymore is deleted.

The output file containing the digest is also a TPM object analogously to the public key. The relevant bytes start from 6 on to the end. In the course of the `New()` function the SHA256 sum of the public key is printed to stdout. It works as a fingerprint for the public key. This string can be passed to to any client which wants to pin the generated public key. The described library also implements a function for doing so and will be explained later.

After the creation of the x509 certificate the DER encoded bytes are added to the Certificate field of the Tpm struct which is a `tls.Certificate`. This struct already contains the opaque private key.

With the `tls.Certificate` struct available a server can be initialized listening for TLS connections. The usage of the required functions is shown in code snippet 4.1:

Listing 4.1: Code snippet from the TPM TLS server

```
1 var tpmHandle *tpm.Tpm = tpm.New("/path/to/workingdirectory")
2
3     cert := tpmHandle.Certificate
4
5     config := &tls.Config{
6         InsecureSkipVerify: true,
```

```
7            CipherSuites: []uint16{
8                tls.TLS_RSA_WITH_AES_256_CBC_SHA       ,
9                tls.TLS_RSA_WITH_3DES_EDE_CBC_SHA      ,
10               tls.TLS_RSA_WITH_AES_128_CBC_SHA       ,
11               tls.TLS_RSA_WITH_AES_256_CBC_SHA       ,
12               tls.TLS_RSA_WITH_AES_128_GCM_SHA256    ,
13               tls.TLS_RSA_WITH_AES_256_GCM_SHA384    ,
14           },
15       Certificates: []tls.Certificate{cert},
16   }
17
18   ln, err := tls.Listen("tcp", ":443", config)
```

Line 1: A Tpm struct is created. By calling the `New()` function the TPM attached to the device is cleared and a new key pair is created, as explained above.

Line 3: The certificate stored in the tpmHandle is extracted. This is not required but makes the usage more clear. This variable will be used in line 15.

Line 5 to 17: A TLS configuration is implemented. First, in line 6 the validation of the TLS Certificates in the handshake is deactivated. It is assumed to be done by each client through PKP. The following lines set the supported ciphers for the TLS connection. The implemented server supports all TLS ciphers of Go tls, which are based on RSA for the TLS handshake.

Line 15: The required certificate (a `tls.Certificate` struct) is set for the config. It was extracted in line 3. With this config a basic TLS server can be run, listening on port 443. This happens at line 18.

An instance of the server listening for connections sends the generated certificate whenever a clients initializes a handshake. The client will then normally use the public key from the sent certificate to encrypt messages to the server.

These are decrypted using the `Decrypt()` function. Like the `Sign()` function, it takes two steps. First, the required data structures are loaded into the TPM. In a second step the main task is fulfilled by decrypting the encrypted file to which the cipher text was written to. This time the output file contains the required text in pure form and not embedded in a special TPM struct. So the content of the file can simply be read and returned by the function.

CheckPeerCert() function: A TLS server using the TPM for secure generation and storage of keys is implemented by the explained code so far. The library also offers the function `CheckPeerCert()` which implements a technique comparable to Syncthing. Syncthing compares the fingerprint of a peer's public key to the ID listed in the configuration. The described function does the same. It marshals the public key of a TLS certificate [3] and compares its SHA256 hash value to a given fingerprint. The client implementation does so by reading the fingerprint from the first command line argument. Exemplary usage is shown in code snippet 4.2:

[3]The received certificate of a connection is stored in a struct, ConnectionState, from which it is extracted.

Listing 4.2: Usage of the CheckPeerCert() function for a Client

```
1  conf := &tls.Config{
2      InsecureSkipVerify: true,
3  }
4
5  conn, err := tls.Dial("tcp", "[1::2]:443", conf)
6  defer conn.Close()
7  fingerprint, err := hex.DecodeString(os.Args[1])
8  if err != nil {panic("could not parse server fingerprint given via command line")}
9  b := tpm.CheckPeerCert(conn, fingerprint)
10 if (b) {...}
11 else {...}
```

Lines 1 to 3: A TLS configuration is set. The verification of the server certificate is deactivated as PKP is used to verify the partner.

Line 5 and 6: The connection is initialized with the given configuration. After this point there is an active connection between the two. The closing of the connection is ensured through the defer statement.

Line 7: The first command line argument is read and decoded as hex string. Using hex values is the standard way for encoding SHA256 hashes.

Line 9: The CheckPeerCert function is called to compare the parsed fingerprint with the hash of the public key in the active TLS connection.

4.4. Evaluation, Applicability and Further Extensions

The exhibit consists of a LAN of two devices. Through Syncthing they build on the BEP to synchronize data, as described in the concept. TPMs where added to the exhibit. This part relies on a TPM device attached to the system using the TPM2 tools.

In Section 4.1, requirements where gathered. They are summarized and evaluated in Table 4.2.

In the way the exhibit is build, it is easy to add other devices. For time reasons this has not been realized in this thesis. The additional benefit would not be large if there is only a third device added. However, the devices added could take other roles, e.g. the role of Discovery Servers or Relay Servers, which are also specified and implemented by Syncthing. Using such entities would reduce the grade of Decentrality. For an industrial context this might be a necessary flaw, as assuming a single LAN is not realistic for an Industry 4.0 scenario: As layed out in Chapter 2 cross factory communication is important. For traversing NAT, Raly Servers can be used [10]. Assuming the possibility of end-to-end connection over IPv6 is not affected by this. These issues will be discussed in more detail in Chapter 5.

The exhibit should show the implementation of technical concepts for an industry context. Therefore it aimed to implement high security. The impact of a compromise of a peer can be reduced by preventing the private keys to be stored on the devices. Also, exploiting memory leaks is mitigated. An approach for ensuring this has been taken but not fully implemented. The corresponding part of the exhibit is a TPM integration into the crypto library of Go.

The application in Syncthing has not been realized due to time issues. However, the implementation is a generic one, using the Go TLS software stack. Thus an integration into

Requirement	Evaluation
Confidentiality	Through TLS Confidentiality is realized. TLS is a standard way for securing network traffic through the Internet. As mentioned, standardized technologies have several advantages.
Authenticity	The communication partners verify themselves to each other using PKP. This implies that the respective public key fingerprints where exchanged in advance. The membership of the Informational Group is thus satisfied implicitly: for a device knowing the fingerprint of a partner implies that the presumed Identity management process has taken care of this problem. The role of the Management node is given implicitly: an External peer (in the Logical Connections model) fulfills the task if it is accordingly configured.
Integrity	Integrity during transfer is realized through the downstream protocols of the BEP, i.e., TLS and TCP. On application level the BEP ensures the Integrity through hash sums on block level of the files.
Availability	The exhibit consists of two devices. Thus this point is not directly satisfied, as if one device fails there is no communication anymore. However, the use of Syncthing implies Decentrality. Depending on the setup with more devices this point can be satisfied: With a higher meshing, increased Availability is given. If all devices build a chain with single connections between the devices, the chain is broken with one device failing. This point is best examined in the view of Logical Connections and is investigated shortly in Chapter 5.
Decentrality	As Syncthing uses a decentralized approach this is satisfied.
Group and Identity management	Through PKP this is satisfied as a process is assumed. The implementation supports a mechanism for a management as demanded. However, there remains a scaling problem. If a peer is excluded from an Informational Group all ex-peers need to be reconfigured. This is discussed in more detail in Chapter 5.
Delta Exchange	Syncthing uses blocks of fixed size together with hashes of these blocks to find differences in the data. As explained in Chapter 3 there exists Content Defined Slicing which can be more effective. Thus, this is only partly satisfied.
Further Hardening	In the implementation TPMs where used to provide a secure storage and handling of private keys. Due to lack of time this has not been integrated into Syncthing. It has been shown, that the approach works.
Data Consistence	Syncthing uses a versioning of blocks to guarantee a consistency in the data in order to reach a global model. Thus, this is satisfied.
Time	In the current implementation Syncthing scans regularly for changes and distributes data if any changes occurred. The software also supports using the inotify subsystem of the Linux kernel to trigger distribution on change.

Table 4.2.: Summary of the requirements for the exhibit

any software written in Go, which builds on TLS for ensuring a secure communication, should be easily be possible. This includes Syncthing. The applicability is not limited to Syncthing, though. The authentication in the implementation is realized by using PKP. The server sets the option `InsecureSkipVerify`, which disables the signature check of Certificates as they are self signed. It would be possible to add a modification of the `New()` function which simply loads a CA signed certificate , thus offering the applicability for a PKI. The private key can then still be stored in the TPM.

As this module does not implement any data exchange, but only security for a TLS connection, for almost all further investigations an integration into other software like in Syncthing is required. In order to test the scalability, e.g., in an emulated enterprise network there is also the possibility of virtualized TPMs: IBM makes a virtualized TPM software available which implements the TCG TPM specifications [23]. It also offers a compatible TPM software stack [24] which should make it easy to use virtualized TPMs in the context of the exhibit implementation.

Currently, there is no way to store generated keys and certificates after a restart of the devices. The certificate is saved in a file in the working directory. One could implement a function which reads the file and includes it in a TLS server configuration. This would only make sense if the TPM keys would not get invalidated after a restart of the device. This can be realized using the TPM2 tools function `TPM2_EvictControl` to make the storage primary key persistent [34]. For an application outside of an exhibit this is necessary, as the keys must be persistent on the devices. Otherwise, the keys on every device have to be regenerated after each restart and the fingerprint of the public key has to be re-transfered.

A risk the current application holds is that files in the working directory may be altered by other processes. One could implement locks to ensure that the files are not touched until the application has terminated.

It would also be possible not to use the TPM2 tools binaries but communicate directly over the sockets provided by the TPM software stack.

4.5. Summary

The requirements for the Concept and the exhibit where gathered in Table 4.1. The Concept shows, how data distribution for a Master-Slave model can be realized, considering how data duplication can be prevented. It builds on models between partners, whose differences are exchanged.

For the exhibit there where Infineon TPMs together with RPis used. As software there was Syncthing chosen, which is written in Go and builds on the BEP. For securing the setup, interfaces of the Go crypto library where implemented to incorporate TPMs in a TLS connection. The TPM acts as a required device for creating x509 certificates and for establishing connections using them. PKP is added as a way to secure the Authenticity. It is realized by using SHA256 sums of the public keys.

The evaluation is gathered in Table 4.2. Almost all requirements have been satisfied or at least it would be not too hard to realize them. There are ways to improve, though. The implemenation of the TPM library is a generic one and it is easily possible to use the code for other Go projects which use the crypto library. It also has weaknesses like the working directory.

5. Discussion

In this Chapter different parts of the thesis are revised and discussed. Firstly, the requirements for an industrial context are examined under the different viewpoints. Secondly, some general considerations for the exhibit and the concept are undertaken.

5.1. Discussion of the Requirements

In Section 2.1 three models where proposed for analyzing the problem of data distribution. The requirements from Chapter 2 are now discussed in the context of these models. If any requirement from Section 2 is not discussed, it is not seen as relevant in the respective model.

Informational Groups. Taking this viewpoint, one is mainly concerned with matters of information asset management in a company. The central requirement emerging out of this model is a need for Group and Identity management, as the view emphasizes exactly the pattern of distrinct groups. If one is looking at data distribution for Industry 4.0 contexts an simplified interconnectedness does fall too short; especially if the goals for this development include data sharing along whole value chains with multiple enterprises along it.

Confidentiality as a requirement follows directly from this analysis as it means exactly to avoid disclosure to outgroups. Throughout the thesis Integrity fell a bit short. For data distribution it is an implicit goal. Assuming that data is not useful if the Integrity is not ensured, this requirement plays a natural role. What exactly the conditions for Integrity in the view of Informational Groups are, depends on the individual case.

Availability as a requirement is also of interest in this view. How important the access to information for every group is can be best analyzed in this view. The results from this analysis can then be used for the model of Logical Connections to ensure the required Availability.

Authenticity has to be satisfied for group membership. Any node has to have the possibility to ensure the group membership of a communication partner.

Logical Connections. This view has been partly excluded from the scope of the thesis. It is still relevant regarding end-to-end connections and in terms of Availability. Securing end-to-end connections has been addressed in the exhibit through securing the private keys for TLS connections. This feature can be seen as relevant for the view of Logical Connections. Thus, the exhibit also shows how a need for an Identity management can be located in this view and how it is correlated to Authenticity.

The other relevant requirement for this view is Availability, which can be realized through a sense-full network layout. Decentrality is directly connected with this. Assuming a centralized network, for a network of n peers there can be $n!$ connections. This is the case in the left part of Figure 5.1. If any peer fails, the Availability is still fully assured. On the other extreme, every peer is only connected with one or two other peers forming a chain. The chain is broken

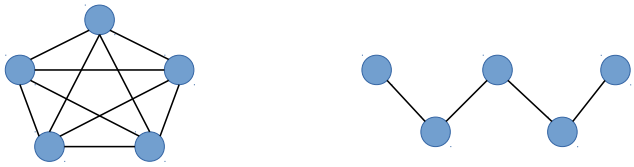

Figure 5.1.: **Two Possibilities of Interconnectedness**

if any middle peer fails, cutting the possibility of transfer. The same holds for a centralized server.

By using, e.g., PKP, as in the exhibit, the Scalability is of great concern in this view. Scalability has been included in the requirements only partly in the context of Decentrality. Thus the thesis falls short in the discussion of such issues. How to design such networks with a good tradeoff of Availabiliy and Scalability in mind is a complex matter, which is outside the scope.

Data Flow. A management of groups is implicitly required in this model as a Master is always the Master for a group. A management of identities is better called a *role management*. The identities of the single nodes is not of question and is addressed through the other views. This view requires a assignment of a Master. It also has to be assured that there is always a Master. As already proposed there might be a protocol for choosing a new one if the old one drops out. This ensures Availability.

Assuming a group management process, Confidentiality is not of concern for this view as the members of a group should already be in the same Informational Group. The same holds for the requirements of Integrity and Authenticity.

5.2. Discussion of the Exhibit

At the end of Chapter 4 the results where gathered in Table 4.1. In this Section a more general inspection of the exhibit is conducted. As one can see in the Table the requirement of Group and Identity management is realized in the exhibit through PKP. In Section 5.1, in the discussion of the view of Logical Connections, it has already been stated that there is a trade off between reliability of such a setup and the number of connections. In the solution of the exhibit every connection between every node has to be configured manually. This scales bad for a large number of members. The approach does, however, open the possibility for granular control of group membership as well as connections through a direct Identity management. The configuration work scales with the number of connections in the network. If a node leaves an Informational group, all its ex-peers need to be reconfigured to not trust this agent anymore.

The exhibit aimed to show how a subset of the Concept from Chapter 4 can be realized. It has been put in place through the BEP implemented by Syncthing. The program does not include roles like Masters or Slaves. Every peer is equal. It rather implements the situation shown in Figure 4.4. For the special case of distributing files from one Master to several Slaves this can be realized through a feature of Syncthing: by allowing only one node to write files. The other peers only synchronize their local files with the Master peer. The case of accumulating information at a Master is not implemented. Rather the data is accumulated

on every peer, not a central one. It can be argued, that all connected peers act as a Master, as long as they do not write data (in the accumulation case, only the Master does not add data).

Considering the management overhead for the need of configuration of all single points in a network, it is obvious to introduce a centralized Identity management structure. An example for this is the RADIUS protocol. However, it would be as well easy to have a central server which takes care of configuring the single devices remotely. This is possible through Syncthings REST API, which allows for remote configuration. Thus, the Decentrality of the informational network is preserved, only the configuration is centralized.

By using TPMs for private key storage in TLS connections it can be guaranteed that an attacker getting unauthorized access to a device, does not get hold of the private key. The corrupted device itself still would act as a regular device in the group and there is no obvious way for a communication partner to identify a malicious device. Thus, the Confidentiality of the data is corrupted too.

The main security improvement lies in the fact that the key never leaves the TPM and is subsequently never stored in the RAM. An attacker exploiting vulnerabilities and aiming to get access to information on the device would not be able to steal the private key. Thus, attacks like Heartbleed are excluded which allow to gain access to sensitive data through a out-of-bounds-read [43]. This makes it much harder for an attacker to impersonate devices on the network. Another advantage is given if a device gets corrupted and is sanitized afterwards. There is no need for reconfiguration of the peered nodes. This would be the case if the private key was stolen, as the derived public key is used for configuration of the peering.

During the work on the thesis a flaw in the RSA key generation procedure of the used Infineon TPMs was discovered. An attacker can use the vulnerability to compute the private key from the public key [19]. Thus, it would be possible to impersonate the attacked device, decrypt traffic and forge the signatures. Subsequently, the Authenticity and Confidentiality is threatened. The implementation is not build on the specific model so it would easily be possible to use another TPM device or change the used methods to Elliptic Curve Cryptography (ECC).

5.3. Summary

The discussion of the three models from Chapter 2 made clear, how the different requirements can emerge from the different viewpoints. The model of Informational Groups is the main viewpoint from which the requirements are infered from. The other two are more concerned with *how* to realize the requirements.

The exhibit has met several requirements but is not applicable to every scenario. The use of Syncthing allows for dynamic configuartion of partners and groups. However, the drawback is the configuration overhead. Through the usage of TPMs there has been assured that exploits reading memory can not access to the private key which enhances Authenticity greatly in defense against impersonation. Through a possible exploit in the used TPM models the security is strongly reduced in the present setup. There is need for either changing the hardware or the used cryptography to ECC.

6. Conclusion and Future Work

This thesis was set in the context of Industry 4.0. It aimed to find requirements for data distribution with a focus on Decentrality. For this, it defined three views or models one can take to approach this topic. In the single views the focus is shifted towards different problems and requirements.

For the IT-security perspective each of the models made other requirements stand out. The three classical security objectives, Confidentiality, Integrity and Availability are emerging out of the view of Informational groups, as this view emphasizes information on an abstract level. It underlines the separation of information against out-groups, the usability of the information and the accessibility of required information, if it is needed. The other two views are more technical. The three objectives are not directly derived from these. For them it is more of concern, how the objectives can be achieved.

Authenticity has the two dimensions of group membership and communication partner Authenticity for a single connection. Closely connected with the first point is the requirement of an appropriate Group and Identity management. This has been realized in the demonstrator through configuration of the single communication partners by defining all other partners through IDs. The Authenticity of communication partners has been implemented using TLS in combination with PKP. For securing the Authenticity, an implementation for using TPMs has been undertaken. Thus it becomes harder to disrupt the Authenticity of the partners by stealing their identities, i.e. their private keys.

As stated in Chapter 5 this end-to-end configuration scales bad for large infrastructures. Future work could try to find a more efficient approach. The most obvious one is to introduce a kind of identity broker comparable to approaches like the RADIUS protocol which, however, is a centralized solution. A hybrid version would be an interesting subject. As already proposed in Chapter 5, building on the REST API of Syncthing, it would also be helpful to have a central management point which configures the network as needed.

If one sticks to the implemented approach it would be feasible to develop a procedure for efficient connection topologies. This topic has also shortly been touched in Chapter 5 in the discussion of Logical Connections. How a meshing of a network is efficiently realized, which finds a tradeoff between Availability, Scalability and network load, is subject to another field of research.

Industry 4.0 aims for autonomous processes whith as little as possible required human intervention. Management of identities is something which is hard to automatize as there are decisions required which are substantial for companies and their confidential information. Developing techniques for automatizing these processes could be an essential area to profit from the ideas of Industry 4.0.

Appendix

A. Preparation of the OS

In this Section the steps are described that where taken to make the TPM device available for the OS.

In a first step the Raspian OS was flashed to a SD card. The exact version of the OS is located at this url: `downloads.raspberrypi.org/raspbian/images/raspbian-2017-01-10/2017-01-11-raspbian-jessie.zip`

Using the tool `raspi-config`, SPI was configured to be loaded by default. From the modified OS the Linux kernel configuration was extracted using `modprobe config`. The config.gz file in the system directory /proc contained the current kernel configuration. This kernel configuration was used when patching the Linux kernel for using the Infineon TPM.

For configuring the kernel it was downloaded from `github.com/raspberrypi/linux.git-brpi-4.4.y`. This version was used because for this kernel Infineon already made a patch available which enabled TPM communication [33]. The diff file was applied to the downloaded Linux kernel. Now the driver for the TPM had to be activated. These two commands where used to enter the configuration menu in which the appropriate TPM Hardware Support was enabled.

```
make ARCH=arm CROSS_COMPILE=arm-Linux-gnueabihf- olddefconfig
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```

Afterwards it was crosscompiled using the kernel configuration extracted from the modified Raspian OS:

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j3 zImage modules dtbs[1]
```

After the crosscompiling did succeed the modified kernel was installed onto the SD card with the Raspian OS by issuing:

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules_install \\
    INSTALL_MOD_PATH=[/path/to/root/filesystem]/
```

As a last step the new kernel image and additional configuration files where copied to the boot partition on the SD card.

```
scripts/mkknlimg arch/arm/boot/zImage [/path/to/sd/card]/boot/$KERNEL.img
cp arch/arm/boot/dts/*.dtb [/path/to/sd/card]/boot/
```

[1] The switch -j3 makes the compiler using three cores.

```
cp arch/arm/boot/dts/overlays/*.dtb* [/path/to/sd/card]/boot/overlays/
```

Now the attached TPM was available in the running OS as a device at /dev/tpm0. For communicating with it the TCG made the code for the TPM2 Software Stack (TSS) available [36] which enables communication with the TPM, based on the specification for the TSS System Level API [53]. The TCG also provides the code for the TPM2 tools, which builds on the TSS and makes tools available to interface easily with the TPM device [35]. Both where downloaded from Github, compiled and installed. Subsequently, the TPM2 tools binaries where available on the systems. The TSS provides the resource manager which is required to be running in order to use the tools. The following line was added to /etc/rc.local to make it start at boot.

```
nohup sudo resourcemgr &
```

The resource manager is a system daemon with a TCP/IP interface over sockets. The TPM2 tools use this interface for communication with the TPM. The task of the tools is to simplify the interaction by marshaling bytestreams to the TPM and unmarshalling bytestream from device. It also multiplexes access to the TPM which enables different programs to use the it [34].

B. Sample Certificate

In the following a sample x509 certificate is shown generated by the TPM library.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 6890850733611472929 (0x5fa1383ba126d821)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = syncthing
        Validity
            Not Before: Oct 5 15:20:58 2017 GMT
            Not After : Dec 31 23:59:59 2049 GMT
        Subject: CN = syncthing
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:97:95:94:8f:45:ed:75:67:cf:95:df:90:c6:af:
                    ef:c5:f9:eb:d4:af:9b:60:58:c4:78:df:40:6c:a3:
                    b2:95:b0:ba:41:82:ae:e7:9e:40:0a:1c:d6:e6:e6:
                    a0:4d:be:5d:a9:6e:bc:1d:a0:d7:93:ad:ed:69:ec:
                    c6:71:e7:88:a3:5b:1f:db:48:b5:4c:2b:6a:ba:37:
                    4f:88:d4:d2:2e:7d:b7:03:65:d3:6f:5e:20:a4:cb:
                    50:b9:f5:a7:dd:ad:9a:6e:c4:45:2f:ec:c9:33:a5:
                    3a:46:9e:39:7b:c5:d8:7f:22:fd:91:55:0a:9e:f9:
                    5b:92:b3:e4:3d:f4:32:72:ff:94:19:4c:29:99:7d:
                    d6:af:de:06:c1:b8:5d:57:f6:2b:78:ee:60:e6:67:
                    18:06:31:da:ef:3e:2a:e5:17:05:20:08:c0:48:53:
                    1b:a9:ee:92:2e:fa:57:0b:62:35:9a:5e:78:48:93:
                    ca:24:8b:54:ba:98:62:79:3d:77:7d:de:c0:e6:52:
```

```
                96:c6:8f:65:05:c1:d8:a3:e6:71:b2:f9:6a:5d:79:
                0e:07:38:69:89:b0:cf:b2:c9:b0:eb:8b:8e:d4:83:
                a7:04:1e:13:44:cc:ec:56:f2:5b:01:29:e1:72:5c:
                4b:44:59:cc:70:6f:2a:81:1e:5f:79:47:b6:d7:1c:
                39:15
            Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Basic Constraints: critical
                CA:FALSE
    Signature Algorithm: sha256WithRSAEncryption
         8e:e1:10:86:e0:8b:0d:67:f0:19:8c:d9:7b:83:40:ff:bd:9b:
         bf:22:b7:d4:54:97:59:fe:e8:5f:94:e1:a5:c4:39:08:0d:07:
         a5:36:3c:a0:f0:7b:29:b5:ea:dd:84:65:15:a5:30:4d:13:1a:
         02:60:b1:4f:0c:42:7e:3b:5e:92:03:21:f8:3f:ac:91:50:7d:
         84:b3:85:d9:c4:11:80:f1:f0:22:ce:d8:c4:37:d8:5e:27:a3:
         5a:7d:fb:03:eb:3d:67:1a:40:bf:2c:b2:75:5e:e3:ff:ae:22:
         0a:bd:ae:6b:24:3d:5f:8f:6c:81:4b:dd:d9:a1:37:c7:ce:c4:
         19:3f:21:bc:d0:d8:66:c9:81:bf:1c:68:84:63:ee:8e:6f:51:
         ae:cb:1e:2e:80:2e:58:09:f7:80:c0:91:a7:17:41:63:6c:33:
         7d:9c:8b:86:ef:d1:d3:70:c5:dc:28:48:0d:0c:e7:7e:ce:79:
         5b:e7:e1:4f:f3:b9:bf:18:9f:9a:9e:43:42:96:20:46:c4:53:
         ba:fb:e6:a3:c8:f1:12:ed:3c:36:08:00:95:a3:03:66:b0:72:
         fd:2b:a2:1b:c9:06:af:8c:55:00:75:d9:3e:63:a0:0f:42:ce:
         e6:db:ba:09:79:eb:fa:b9:70:ea:38:13:db:09:97:ce:8f:b7:
         7f:14:73:66
```

Bibliography

[1] R. L. A. Buldas A. Kroonmaa. *Keyless Signatures' Infrastructure: How to Build Global Distributed Hash-Trees*. Tech. rep. Gollmann D. (eds) Secure IT Systems, 2013.

[2] U. G. C. S. Advise. *Distributed Ledger Technology: beyond block chain*. seen 23. November 2017. 2016. URL: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf.

[3] F. I. AISEC. *AP 4. Secure Environments. Konzepte zur sicheren Verteilung von Daten entlang der Wertschöpfungsketten*. Tech. rep. Fraunhofer Institut AISEC, 2017.

[4] C. R. et al. *RFC 2865*. seen 5. December 2017. 2000. URL: https://tools.ietf.org/html/rfc2865.

[5] E. A. et al. *RFC 4871*. seen 30. November 2017. 2007. URL: https://www.ietf.org/rfc/rfc4871.txt.

[6] J. R. et al. *RFC3489*. seen 23. November 2017. Mar. 2003. URL: https://tools.ietf.org/html/rfc3498.

[7] M. B. et al. "How Virtualization, Decentralization and Network Building Change the Manufacturing Landscape: An Industry 4.0 Perspective". In: *International journal of mechanical, aerospace, industrial and mechatronics engineering* 8.1 (2014), pp. 37–44.

[8] M. Y. et al. *practical robust communication in dhts tolerating a byzantine adversary*. Tech. rep. ieee 30th international conference on distributed computing systems, 2010.

[9] O. T. et al. *RFC 7157*. seen 24. November 2017. 2014. URL: https://tools.ietf.org/html/rfc7157.

[10] R. M. et al. *RFC 8155*. seen 19. January 2018. 2010. URL: https://tools.ietf.org/html/rfc5766.

[11] Y. K. et al. *Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors*. Tech. rep. Carnegie Mellon University, Intel Lab, 2014.

[12] K. K. B. Möller T. Duong. *Libtorrent library*. seen 17. November 2017. 2014. URL: https://www.openssl.org/~bodo/ssl-poodle.pdf.

[13] E. Bertino and N. Islam. "Botnets and Internet of Things Security". In: *Computer* 50.2 (Feb. 2017), pp. 76–79. DOI: 10.1109/MC.2017.62.

[14] BitTorrent. *utorrent*. seen 15. November 2017. 2017. URL: https://www.utorrent.com/.

[15] J. Borg. *Connection over UDP*. sen 4. December 2017. 2017. URL: https://forum.syncthing.net/t/connections-over-udp/9382.

[16] T. B. Collective. *Borg. Data structures and file formats*. seen 11. December 2017. 2015. URL: https://borgbackup.readthedocs.io/en/stable/internals/data-structures.html.

[17] M. G. R. P. F. S. D. Goldschlag. "Onion Routing for Anonymous and Private Internet Connections". In: *Communications of the ACM* 42.2 (1999).

[18] J. A. Donenfeld. *Tox Handshake Vulnerability*. seen 19. November 2017. 2017. URL: https://github.com/TokTok/c-toxcore/issues/426.

[19] C. F. U. o. V. Enigma Bridge Ltd. *ROCA: Vulnerable RSA generation (CVE-2017-15361)*. seen 3. January 2018. 2017. URL: https://crocs.fi.muni.cz/public/papers/rsa_ccs17.

[20] T. Flexible and S. E.-E. C. in Industry 4.0. *Design Principles for Industrie 4.0 Scenarios*. Tech. rep. IEEE 15th International Conference on Industrial Informatics INDIN'2017, 2017.

[21] L. Foundation. *Linux Foundation Unites Industry Leaders to Advance Blockchain Technology*. seen 22. November 2017. 2015. URL: https://www.linuxfoundation.org/press-release/linux-foundation-unites-industry-leaders-to-advance-blockchain-technology/#.WZ8FmCiG.

[22] S. Frei. *syncthing-inotify*. seen 21. November 2017. 2017. URL: https://github.com/syncthing/syncthing-inotify.

[23] K. Goldman. *IBM's Software TPM 2.0*. sen 4. December 2017. 2017. URL: https://sourceforge.net/projects/ibmswtpm2/.

[24] K. Goldman. *IBM's TPM 2.0 TSS*. sen 4. December 2017. 2017. URL: https://sourceforge.net/projects/ibmtpm20tss/.

[25] T. C. Group. *TCG Releases TPM 2.0 Specification for Improved Platform and Device Security Part 1: Architecture*. sen 23. November 2017. 2014. URL: https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-1-Architecture-01.38.pdf.

[26] T. C. Group. *TCG Releases TPM 2.0 Specification for Improved Platform and Device Security Part 2: Structures*. sen 30. November 2017. 2014. URL: https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-2-Structures-01.38.pdf.

[27] T. C. Group. *TCG Releases TPM 2.0 Specification for Improved Platform and Device Security Part 3: Commands*. sen 30. November 2017. 2014. URL: https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-3-Commands-01.38.pdf.

[28] T. C. Group. *TPM Main Specification*. seen 23. November 2017. 2011. URL: https://trustedcomputinggroup.org/tpm-main-specification/.

[29] T. C. Group. *Trusted Computing Group Releases TPM 2.0 Specification for Improved Platform and Device Security*. sen 23. November 2017. 2014. URL: https://trustedcomputinggroup.org/trusted-computing-group-releases-tpm-2-0-specification-improved-platform-device-security/.

[30] A. R. (Hrsg.) *Einführung und Umsetzung von Industrie 4.0*. 1. Springer, Apr. 2016.

[31] G. Inc. *Package crypto*. sen 4. December 2017. 2017. URL: https://golang.org/pkg/crypto/.

[32] Infineon. *IRIDIUM9670 TPM2.0 LINUX*. seen 27. November 2017. 2017. URL: https://www.infineon.com/cms/de/product/evaluation-boards/iridium9670-tpm2.0-linux/#support.

[33] Infineon. *SLB 9670 VQ1.2 FW6.40*. seen 27. November 2017. 2017. URL: https://www.infineon.com/dgdl/Infineon-%20SLB%209645_SLB%209670%20TPM%201.2%20-AN-v06_16-EN.zip?fileId=5546d46255a50e820155b535d44d754f.

[34] Infineon. *TPM Application Note*. Tech. rep. Infineon Technologies AG, 2017.

[35] Intel. *tpm2-tools*. seen 28. November 2017. 2017. URL: https://github.com/intel/tpm2-tools.

[36] Intel. *tpm2-tss*. seen 28. November 2017. 2017. URL: https://github.com/intel/tpm2-tss.

[37] *ISO/IEC 27000:2016*. 4th ed. International Organization for Standardization. ISO/IEC JTC 1/SC 27 IT Security techniques, Feb. 2016. URL: https://www.iso.org/standard/66435.html.

[38] A. B. u. a. J. Borg. *Syncthing Projekt Seite*. seen 14. November 2017. 2016. URL: https://syncthing.net/.

[39] S. B. J.A. Johnsen L.E. Karlsen. *Peer-to-peer networking with BitTorrent*. Tech. rep. Department of Telematics, NTNU, 2005.

[40] J. H. JP. Martin-Flatin S. Znaty. "A Survey of Distributed Enterprise Network and Systems Management Paradigms". In: *Journal of Network and Systems Management* 7.1 (Mar. 1999), pp. 9–26. DOI: 10.1023/A:1018761615354. URL: https://doi.org/10.1023/A:101876161535.

[41] B. L. M. Castro. *Practical Byzantine Fault Tolerance*. Tech. rep. Massachusetts Institute of Technology, 1999.

[42] B. O. M. Hermann T. Pentek. *Design Principles for Industrie 4.0 Scenarios*. Tech. rep. Hawaii International Conference on System Science, 2016.

[43] N. Mehta. *CVE-2014-0160*. seen 7. January 2018. 2014. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160.

[44] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. seen 23. November 2017. 2009. URL: https://bitcoin.org/bitcoin.pdf.

[45] A. Norberg. *Libtorrent library*. seen 17. November 2017. 2012. URL: http://libtorrent.org/reference.html.

[46] D. M. P. Maymounkov. *Kademlia: A Peer-to-peer Information System Based on the XOR Metric*. Tech. rep. In: Druschel P., Kaashoek F., Rowstron A. (eds) Peer-to-Peer Systems. IPTPS 2002, 2002.

[47] R. Pfitzer. *Trusted Computing*. 1. Verlag Recht und Wirtschaft, Apr. 2004.

[48] L. P. R. Fagin P. G. Kolaitis. "Data exchange: getting to the core". In: *ACM Transactions on Database Systems (TODS)* 30 (Mar. 2005), pp. 174–210.

[49] K. W. (R. Steinmez. *Peer-to-Peer Systemes and applications*. 1. Springer, 2005.

[50] K. R.Griesemer R. Pike. *The Go Programming Language*. seen 13. December 2017. 2017. URL: https://golang.org/.

[51] Syncthing. *Syncthing tlsutil.go*. seen 30. November 2017. 2017. URL: https://github.com/syncthing/syncthing/blob/master/lib/tlsutil/tlsutil.go.

[52] M. K. T. Kivinen. *RFC 3526*. seen 15. November 2017. 2003. URL: https://tools.ietf.org/html/rfc3526#section-8.

[53] TCG. *TSS System Level API and TPM Command Transmission Interface Specification*. seen 29. November 2017. 2017. URL: https://trustedcomputinggroup.org/tss-system-level-api-tpm-command-transmission-interface-specification/.

[54] S. Team. *Local Discovery Protocol*. seen 21. November 2017. 2017. URL: https://docs.syncthing.net/specs/.

[55] T. Team. *Tox FAQ*. seen 19. November 2017. 2017. URL: https://wiki.tox.chat/users/techfaq.

[56] T. Team. *Tox specification*. seen 18. November 2017. 2017. URL: https://toktok.ltd/spec.html.

[57] yubico team. *yubico website*. seen 11. December 2017. 2017. URL: https://www.yubico.com/.

[58] R. Uzgalis. *BUZ Hash*. seen 15. November 2017. 1995. URL: http://www.serve.net/buz/Notes.1st.year/HTML/C6/rand.012.html.

[59] P. M. V. Corvello. "Virtual forms for the organization of production: A comparative analysis". In: *International Journal of Production Economics* 110 (Feb. 2007), pp. 5–15.

[60] B. für Wirtschaft und Energie. *Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0)*. seen 7. November 2017. 2016. URL: https://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/rami40-eine-einfuehrung.pdf?__blob=publicationFile&v=7.

List of Figures

Acronyms

BEP Block Exchange Protocol

CA Certificate Authority

CDS Content Defined Slicing

CPS Cyber Physical System

DHT Distributed Hashtable

EK Endorsement Key

HSM Hardware Security Module

IoE Internet of Everything

IoT Internet of Things

IP Internet Protocol

IPv4 Internet Protocol Version 4

IPv6 Internet Protocol Version 6

LAN Local Area Network

MITM Man In The Middle

NAT Network Address Translation

NVM Non Volatile Memory

OS Operating System

P2P Peer-to-Peer

PKI Public Key Infrastructure

PKP Public Key Pinning

RAMI 4.0 Reference Architecture Model for Industrie 4.0

RPi Raspberry Pi

SPF Single Point of Failure

SPS Storage Primary Seed

SSH Secure Shell

TLS Transport Layer Security

TPM Trusted Platform Module

TTP Trusted Third Party

UC Ubiquitous Computing

XOR Exclusive Or